SNIPPET GENERATION USING LAIR (LOCAL ALIGNMENT FOR INFORMATION RETRIEVAL)

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERT ANIL YILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2023

Approval of the thesis:

SNIPPET GENERATION USING LAIR (LOCAL ALIGNMENT FOR INFORMATION RETRIEVAL)

submitted by **MERT ANIL YILMAZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Nihan Kesim Çiçekli
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Çiğdem Keyder Turhan
Software Engineering, Atılım University _____

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering, METU _____

Assist. Prof. Dr. Burçak Otlu Sarıtaş
Health Informatics, METU _____

Date:23.06.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Mert Anıl Yılmaz

Signature :

ABSTRACT

SNIPPET GENERATION USING LAIR (LOCAL ALIGNMENT FOR INFORMATION RETRIEVAL)

Yılmaz, Mert Anıl
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Nihan Kesim Çiçekli

June 2023, 69 pages

The main objective of this thesis is to show the extraction of sections relevant to a query from ranked documents in the form of a snippet (summary). To achieve this, we adapt the local sequence alignment method, commonly used in bioinformatics to identify similarities between gene sequences. By applying this method in information retrieval, we can spot the relevant sections within the ranked documents. To address the shortcomings of this approach in terms of data processing, including time and computational power, we reduce them to a negligible level by applying the method to only relevant documents. The quality of the snippets is assessed by employing various metrics and they are compared with Google snippets. In addition, the benefits of the local sequence alignment method are emphasized by adapting this method to query searches in YouTube videos.

Keywords: Information Retrieval, Bioinformatics, Local Sequence Alignment, Snippet Generation

ÖZ

LAIR (BİLGİ ALMA İÇİN YEREL HİZALAMA) KULLANARAK PASAJ OLUŞTURMA

Yılmaz, Mert Anıl
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Nihan Kesim Çiçekli

Haziran 2023 , 69 sayfa

Bu tezin temel amacı, sıralanmış belgelerin bir sorguyla ilgili bölümlerinin çıkarılmasını bir pasaj (özet) biçiminde göstermektir. Bunu başarmak için, gen dizileri arasındaki benzerlikleri belirlemek amacıyla biyoenformatikte yaygın olarak kullanılan yerel dizi hizalaması yöntemini uyarlıyoruz. Bu yöntemi bilgi almada uygulayarak, sıralanan belgelerde ilgili bölümleri tespit edebiliriz. Zaman ve hesaplama gücü dahil olmak üzere veri işleme açısından bu yaklaşımın eksikliklerini gidermek için, yöntemi yalnızca ilgili belgelere uygulayarak bunları ihmal edilebilir bir düzeye indiriyoruz. Pasajların kalitesi çeşitli metrikler kullanılarak değerlendirilmiştir ve Google pasajları ile karşılaştırılmıştır. Ayrıca bu yöntemin YouTube videolarındaki sorgulara uyarlanmasıyla yerel dizi hizalaması yönteminin faydaları vurgulanıyor.

Anahtar Kelimeler: Bilgi Alma, Biyoenformatik, Yerel Dizi Hizalaması, Pasaj Oluşturma

To Birdem,

My love, my life, my light.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Prof. Dr. Nihan Kesim Çiçekli for her support and guidance from the beginning to the end of this study. She was supportive when I was feeling pessimistic.

Secondly, Prof. Dr. Tolga Can, made me love bioinformatics and kept my curiosity alive for this field. Thanks to him, I was able to establish a connection between information retrieval and bioinformatics.

Finally, I would like to thank my family who has always supported me even if I could not spend time with them while working on my thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CSE	Custom Search Engine
CSV	Comma Separated Values
DNA	Deoxyribonucleic Acid
IDF	Inverse Document Frequency
IR	Information Retrieval
IWT	Instance-Weighted Training
JSON	JavaScript Object Notation
LAIR	Local Alignment for Information Retrieval
MSA	Multiple Sequence Alignment
NDVR	Near-Duplicate Video Retrieval
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
SERP	Search Engine Results Page
TF	Term Frequency

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Today, the Internet has become an essential aspect of people's lives. The act of obtaining desired information which was previously accomplished through encyclopedias has now been replaced by Internet searches. The main advantage of the Internet over the encyclopedias is time. The time of accessing information is significantly quicker with the Internet. However, this leads to another issue: the exponential growth of web content. Comparing the retrieved information is important in terms of reaching the relevant information, but it is becoming more difficult day by day for users to find relevant information efficiently due to the abundance of web pages and information resources.

The queries written by the users are crucial to reach the desired information on various web pages. However, these query words may not be comprehensive enough, and therefore unrelated web pages can appear at the top of the results. To overcome this problem, there are many studies[3][4] aiming to enrich the query. Yet, it is not enough because even if relevant web pages are obtained, users still need to search for the exact information in the contents of the web pages which is another time-consuming process. This is the point at which snippets become critical. The snippets give an idea of whether the retrieved pages provide the desired information for access without having to read all the content on the highest-ranked web pages. Thus, the snippets make it easy for the user to choose among relevant websites. If the information in the snippet is sufficient, users may or may not need to look at the websites. Apart from this, snippets are beneficial for mobile device users because these devices have

limited screen space and slower internet speeds because of Wireless connection [5].

There are two types of snippets. The first one is query-independent snippets which have static content for each website. The main purpose of these snippets is to summarize the key parts of the website content. Since they are independent of the query, the user may need to search for the query words within the content by reading it. For this reason, query-dependent snippets, the second type of snippets, are generally preferred today. The type of snippet focused on in this thesis is query-dependent snippets. It is aimed to improve the quality of snippets by integrating local sequence alignment, which is a frequently used approach in bioinformatics to identify similarities in DNA and gene sequences, into snippet generation.

Although written contents are very valuable as a source of information, video-based contents are also frequently preferred today to access information. YouTube is a video platform that is widely preferred and includes various types of content such as news, entertainment, education, etc. In the Copyright Transparency Report of YouTube [1], it is explained that over 2 billion logged-in users come to YouTube every month and more than 500 hours of video are uploaded every minute.

While videos are widely used as a source of information, it remains a challenge that snippet generation is not commonly employed in videos. Handling a large amount of video content may appear discouraging; however, snippet generation can offer users precise and concise summaries of the video content.

For videos, snippet generation may correspond to small parts of the video or suggestions to the user from which minute to watch. By extracting small segments, users can quickly grasp the essence of the video without having to watch it in its entirety. Additionally, providing recommendations on which specific minute to watch allows users to efficiently navigate through lengthy videos and access the most relevant information. In this thesis, after evaluating the quality of the snippets created by local sequence alignment, the method is integrated and applied to YouTube videos.

In this work, the use of local sequence alignment in information retrieval is abbreviated as LAIR: Local Alignment for Information Retrieval. At this point, it is possible

[1] `http://transparencyreport.google.com/`

to make an analogy. The definition of the word, lair[2] is a place where a wild animal lives, often underground and hidden, or a place where a person hides. Similarly, in information retrieval, a lair can refer to a set of information that is difficult to access and maybe a part of documents or videos. Just as animals cannot be easily spotted in a shelter, information may be hidden in long documents or videos. Accessing information is valuable and this thesis proposes the LAIR as a way to access information.

Overall, this thesis aims to provide an overview of current approaches to generating web page snippets and highlight the challenges and opportunities in this area of research, and then present a new idea, LAIR, to improve the snippet quality. By integrating this idea into videos, the thesis emphasizes the advantages of the method and its applicability to different areas.

1.2 Proposed Methods and Models

This study proposes the method LAIR, which is basically the adaptation of local sequence alignment to information retrieval for snippet generation. This adaptation is achieved through a series of logical steps. First of all, drawing parallels between the process of detecting similarity in DNA and gene sequences and the query search process, documents can be seen as analogous to target gene sequences, while queries can be seen as analogous to gene query sequences, documents are compared to target gene sequences, and queries are compared to gene query sequences. Once the similarity is established, using the positional index approach, local sequence alignment can be used in information retrieval, similar to its application in bioinformatics. By utilizing a basic scoring function, document-query matches can be scored, enabling the extraction of the starting and ending words of the matching segments.

In the next step, it is examined whether the basic local sequence alignment algorithm is suitable for query expansion, or not. This thesis proposes the use of a substitution matrix to make a basic local sequence alignment algorithm suitable for query expansion. Thus, it is aimed to increase the relevance of query-document matches by using an enriched query. By using the substitution matrix, the algorithm can be executed

[2] https://dictionary.cambridge.org/dictionary/english/lair

with the desired query expansion method. In this thesis, WordNet [6], which is a common query expansion method, is preferred, since the main purpose is to demonstrate the feasibility and to observe the outcomes of query expansion.

After that, it is proven that the TF-IDF approach can also be integrated into local sequence alignment, again using the substitution matrix. Note that, in this study, only IDF values are integrated and the results are evaluated, but TF values can be integrated in a similar fashion.

In the last step, the snippet is generated using the proposed method. During snippet generation, the most important point is the ability to obtain the starting and ending words of the snippet through local sequence alignment. In addition, local sequence alignment can evaluate the document as a whole when creating a snippet.

Snippet generation should not be limited to documents only. After evaluating the quality of snippets generated by LAIR in comparison to Google snippets, this method is also applied to extract snippets from YouTube videos. Auto-generated subtitles of the corresponding videos are treated as the source documents in this process. The text within the subtitles is used to extract snippets to provide users with concise summaries of the video content.

Note that LAIR can be used to rank documents as well. However, in the thesis, the ranking ability of this method is not employed due to its computational complexity. In the context of snippet generation, the computational cost is ignored because this method is applied only to a limited number of documents, rather than N documents where N is hypothetically infinite.

1.3 Contributions and Novelties

The primary contributions of the thesis are as follows:

- The proposed method, LAIR, highlights the similarities between information retrieval and bioinformatics by applying local sequence alignment to the snippet generation process.

- LAIR can be used in phrase query searches as well as proximity query searches. In proximity query search, the distance of two query terms is decided by the value of k. It is shown in the following:

$$| index\_of\_term\_2 - index\_of\_term\_1 | \leq k \tag{1.1}$$

  For instance, suppose that the proximity query is "computer science" and the document includes the sentence "He uses his computer to study science problems". In this example, if the value of k is set to 4, then there is a match because the distance between the query terms is 3. However, if k is specified as 1, then there is no match. On the other hand, LAIR gives flexibility by not predefining a specific k value. Instead, the optimal value of k is dynamically determined based on the scoring function. So if the sentence in the example is the best match in the document, LAIR gives the sentence as output without setting any limit value.

- The compatibility of LAIR with preprocessing techniques as well as its adaptability to query expansion and TF-IDF approaches can increase LAIR's compatibility with different information retrieval scenarios.

- LAIR has an advantage over other snippet generation algorithms. Since other snippet algorithms use ranking algorithms that decide whether documents are relevant or not, they also need to identify some parts of the document as potential snippets before choosing the best snippet. As a result, generating a snippet without looking at the entire document may limit the quality of the snippets. In contrast, LAIR takes a holistic approach by considering the document as a whole during the querying process, ensuring improved snippet quality.

- By adapting LAIR to YouTube videos, it becomes possible to generate snippets as a result of query searches conducted on YouTube. Since query-dependent snippets are created, the user can find the information they are looking for by watching only a small part of the video instead of having to watch the whole video.

1.4 The Outline of the Thesis

The rest of this thesis is organized as follows.

- Chapter 2 provides a summary of the literature that emphasizes the similarities between information retrieval and bioinformatics. It also reviews the related work regarding snippet generation and query searches on YouTube.
- Chapter 3 explains why local sequence alignment can be used in the information retrieval area. It further presents our proposed solution, LAIR, for snippet generation.
- Chapter 4 presents the experiments conducted and provides evaluations using various metrics: number of query terms in snippets, snippet length in terms of characters, number of highlighted query terms per snippet, number of non-readable characters, number of ellipsis in snippets, number of no-highlighted-snippet, and number of no-snippet web page. These metrics provide insights into the content and quality of the generated snippets, allowing for an assessment of their readability and informativeness.
- Chapter 5 emphasizes the adaptability of LAIR in different areas such as video searches on YouTube.

CHAPTER 2

RELATED WORK

The basis of this thesis relies on the similarities established in the fields of bioinformatics and information retrieval. Hence, first of all, it is necessary to examine the studies not only in the field of information retrieval but also in the field of bioinformatics in order to identify the similarities that have been determined between these two fields. Even though bioinformatics and information retrieval are two different subjects, their approaches can be used interchangeably. For instance, there is a study adapting inverted index and positional index into bioinformatics. Similarly, there are investigations within information retrieval that rely on sequence alignment as a foundational principle. Furthermore, as these concepts are explained, other closely related information retrieval concepts are also discussed. Afterwards, studies related to snippets are described in order to gain a better understanding of the snippet generation method used in the thesis. Moreover, prior to integrating snippet generation into video search, it is beneficial to examine existing studies in this domain to acquire foundational knowledge.

2.1 Sequence Alignment

2.1.1 What is Sequence Alignment?

Sequence alignment is a technique used to detect similarities between two or more sequences. It is often used in bioinformatics. Arranging two or more sequences may give information regarding their similarities, differences, and evolutionary relationships. For instance, by aligning sequences from related organisms, researchers can identify conserved regions and understand common ancestry. This technique can be

used to understand the genetic basis of diseases, identify disease-causing mutations, and develop personalized medicine approaches.

There are various possible alignments for two sequences, and the number increases exponentially with the length of the sequences. However, it is impossible to examine all possibilities to choose the best alignment. Thus, a dynamic programming approach is used in sequence alignment algorithms. This approach ensures optimum alignment by considering all possible alignments and calculating their scores. It enables avoiding unnecessary computations and efficient alignment calculations by dividing the alignment problem into smaller subproblems and storing the intermediate results.

A dynamic programming algorithm for a sequence alignment needs the general preprocessing steps:

- To determine the best alignment, there must be a scoring function that assigns scores to matches, mismatches, and gaps in the aligned sequences. There can be different functions depending on the purpose and usage area of the alignment, but a basic scoring function has the following scores:
  - Match score is a positive score for aligning identical residues, indicating a match. For instance, a match between two identical amino acids could be assigned a score of +1 or higher.
  - Mismatch score is a negative score for aligning different residues, indicating a mismatch. Mismatches typically receive negative scores to discourage aligning dissimilar residues.
  - Gap penalty is a negative score for opening a gap in one of the sequences or extending a gap. Gaps represent regions where residues are not aligned. The gap penalty encourages the alignment algorithm to minimize the number and length of gaps.

  If there is a more complex relation between residues than match and mismatch, the substitution matrix can be used when creating the scoring function. This matrix includes scoring for each binary combination of residues. Hence, it provides a systematic way to assign scores to different substitutions based on their observed frequencies or probabilities in known sequences.

- A scoring matrix is constructed. This is usually represented as a two-dimensional matrix, $((m+1)\text{x}(n+1))$ where m and n are the lengths of two sequences. The first row and column of the matrix are initialized by using scoring functions and substitution matrices. The results in the first row and column represent the base cases for dynamic programming recursion.

After the preprocessing, the scoring matrix is filled cell by cell, considering the scores of neighboring cells. Each cell represents the alignment score of two sub-sequences. In the equation below [7], $V(i,j)$ defines the current cell in the scoring matrix and δ represents the scoring function. Hence, the recurrence relation defines how the score in $V(i,j)$, depends on the scores of neighboring cells which are $V(i-1,j-1)$, $V(i-1,j)$, and $V(i,j-1)$.

$$V(i,j) = max \begin{cases} V(i-1,j-1) + \delta(S[i],T[j]) & \text{match/mismatch} \\ V(i-1,j) + \delta(S[i],\_) & \text{delete} \\ V(i,j-1) + \delta(\_,T[j]) & \text{insert} \end{cases} \quad where\ i,j > 0 \tag{2.1}$$

The score of the best alignment is $V(m,n)$ where m and n are the lengths of the sequences. This is the highest score in the filled matrix. Finally, by using the back-tracing technique, the best alignment is obtained.

There are basically two methods of sequence alignment in bioinformatics: global sequence alignment and local sequence alignment. Firstly, the Needleman-Wunsch Algorithm [8] is based on the dynamic programming approach and scores similarity between two character strings globally. It is called global sequence alignment. The other method, the Smith-Waterman algorithm [9] determines the similar parts between two character strings and calculates the similarity scores for these parts based on dynamic programming. This algorithm specifically focuses on local similarity between character strings. It is called local sequence alignment.

To decide which sequence alignment algorithm is more suitable for information retrieval, the working principle of global sequence alignment and local sequence alignment algorithms can be examined.

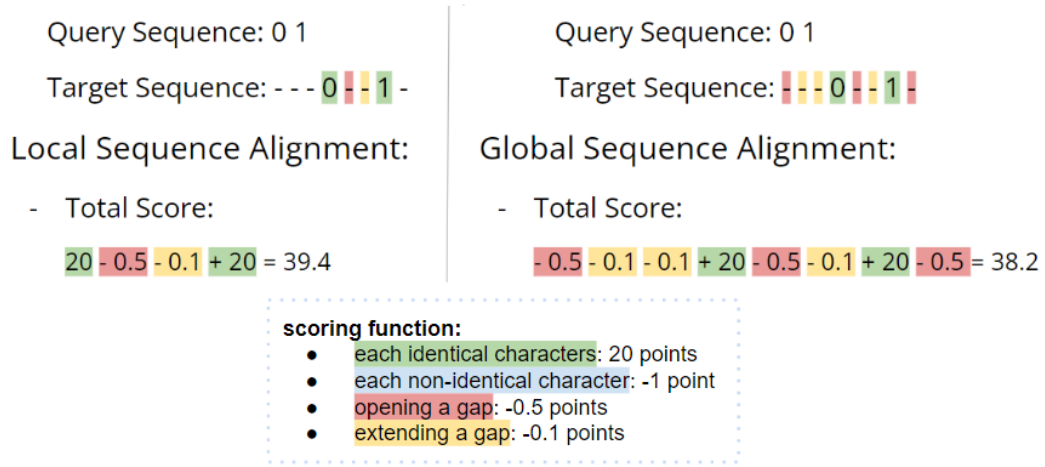

Figure 2.1: Local and global sequence alignment scoring examples

Figure 2.1 illustrates the behavior of global sequence alignment and local sequence alignment for long target sequences and much shorter query sequence inputs. In information retrieval, generally, documents have hundreds of words, but queries have a few words. Figure 2.1 demonstrates that the beginning and the end of the document are irrelevant for local sequence alignment. However, global alignment penalizes the beginning and the end of the document because there are violations that are opening a gap and extending it according to its working principle. It means that, for global sequence alignment, longer documents cause less score. Therefore, it produces misleading results.

2.1.2 Applications in IR

LAIR is not the first study to use sequence alignments in information retrieval. A previous study explored the similarities between musical notes and gene characters [10]. Savage and Atkinson introduce a method to automatically identify the tune family of a musical piece by adapting sequence alignment techniques into the domain of the music. In this study, each melody is modeled as a sequence constructed from an alphabet. Similar to how DNA is represented using a set of four nucleic acids (C, G, A, or T), or how proteins are represented using a set of 20 amino acids, melodies can be represented as a sequence using a set of 12 pitch classes that correspond to the 12 notes of the chromatic scale. Figure 2.2 shows how these are visualized using

standard piano keyboard notation in this study.

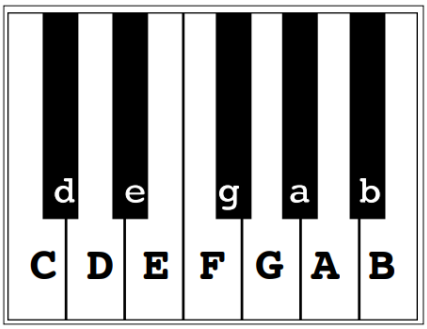

Figure 2.2: Character representations of notes

Finally, they propose a scoring scheme based on musical intervals and rhythmic patterns and employ dynamic programming algorithms for aligning musical sequences.

Another study [11] proposes a hybrid plagiarism detection method by using Levenshtein distance and a simplified Smith-Waterman algorithm. Levenshtein distance is a metric used to measure the edit distance between two strings. It calculates the minimum number of operations required to transform one string into another, where operations can be character insertions, deletions, or substitutions. Before using these algorithms, the lexical entities are represented using integers. The study explains the way through the example in Figure 2.3

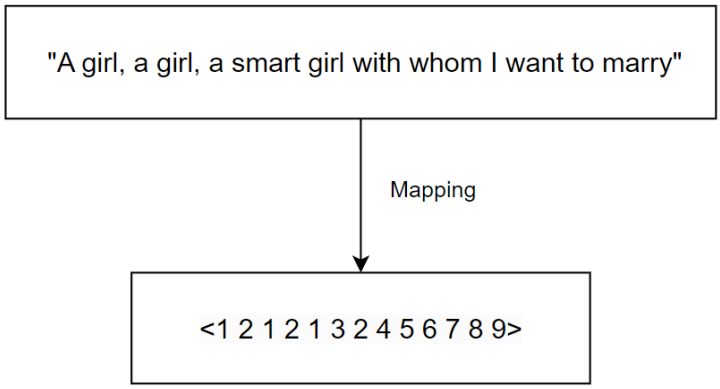

Figure 2.3: Integer representations of lexical entities

By comparing two texts in this way, it aims to find similarities between them and to detect plagiarism. However, the absence of text preprocessing in this study makes

the detection of similar word groups difficult. Also, since the score of the sequence alignment is directly related to the word order, even when the order of the words is changed, the plagiarism score will drop and it will be difficult to detect the plagiarism.

2.2 Basic IR Concepts

2.2.1 Inverted Index and Positional Index

Inverted index [12] is a widely used data structure in information retrieval and it enables efficient keyword-based searches. This data structure consists of the term and a list of document identifiers where the term occurs. Figure 2.4 can be given as an example of inverted index.

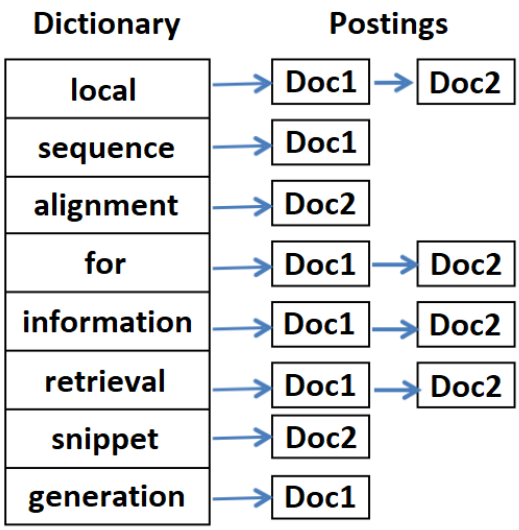

Figure 2.4: An example of inverted index

Although inverted indexes are highly effective in document retrieval, they do not provide information about the position of terms within the documents. Hence, this approach is not applicable to tasks that require precise term position information such as phrase and proximity searches. On the other hand, according to Manning, Raghavan, and Schütze [13], positional index is a suitable data structure for these searches. Positional index is an extension of inverted index. The main difference between inverted index and positional index lies in the additional position information stored within

the corresponding document shown in Figure 2.5.

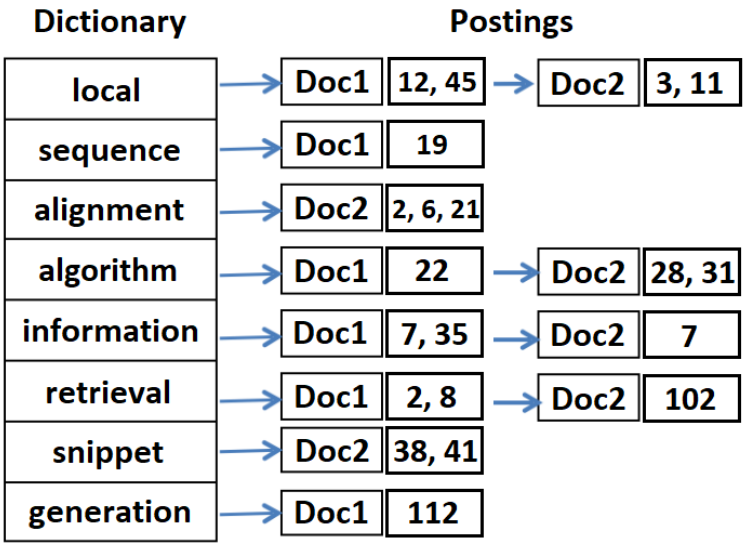

Figure 2.5: An example of positional index

2.2.2 Document and Query Preprocessing

The purpose of preprocessing operations is to obtain data structures such as inverted index and positional index. In this way, information can be retrieved in an efficient way. Preprocessing steps vary because the requirements of each project are different. However, some preprocessing techniques listed below are common in many projects.

- **Tokenization:** It is used to splits into words, phrases, symbols, or other meaningful elements.
- **Stop words removal:** Some of the words are common and highly repetitive for many documents. These words are called stop words, and they are different in each language. They are commonly removed from tokens of a document because their statistical significance is low for information retrieval. In fact, there is an advantage in that the system needs less memory and computational power after the elimination, which provides an advantage.
- **Stemming:** It is used to obtain the common forms [14] of words such as different suffixes, in other words, to group the words by determining a common

form. There are many stemming methods used to find this commonality between words, and they are usually divided into three groups which are truncating methods, statistical methods, and mixed methods [15]. Porter stemming [16] is one of the truncating methods and its simplicity compared to several other methods has made it popular. Furthermore, the lemmatization step, which is not in this pipeline but is used in many studies, is often compared to stemming. Stemming [17] is a procedure to reduce all words with the same stem to a common form whereas lemmatization removes inflectional endings and returns the base or dictionary form of a word. Although lemmatization is ahead in terms of retrieval performance [17], the stemming technique performed better in terms of computational speed [18] and this is the reason why stemming is generally preferred over lemmatization.

2.2.3 TF/IDF

For some studies, rare words may be more important. To decide the importance of a word, it is usually checked how often it is found in documents. TF [19] refers to how important if it occurs more frequently in a document. Therefore, the higher TF reflects the more estimated that the term is significant in respective documents. Additionally, IDF is calculated on how infrequent a word or term is in the documents.

2.2.4 BM25

The BM25 algorithm [20], which is frequently used for ranking documents today, has been developed based on the Probabilistic Relevance Framework. There are many algorithms [21] created based on this algorithm such as BM25+, BM25T, BM25-adpt, and BM25L, etc. Furthermore, it is possible to access many open-source implementations of this algorithm, and Okapi[1] is one of them.

[1] http://www.soi.city.ac.uk/

2.2.5 Query Expansion

Query expansion [22] is a technique for expanding the initial query by adding new related terms. The main purpose of query expansion is to increase the accuracy of document matches by enriching the query. Although, at first, synonyms [23] were used for the query expansion, studies in this field are still ongoing and there are many studies to determine query-related words such as query expansion using word embeddings [24]. In this thesis, query expansion is integrated into the proposed method by enriching queries with synonyms.

2.2.6 Applications in Bioinformatics

In addition to the integration of sequence alignment algorithms into information retrieval, in bioinformatics, information retrieval approaches are also seen. There is a study [1] which proposes a new DNA alignment method by using an inverted index and positional index. In this study, Liang and Kayong first split DNA sequences into segments, referring to their previous work [25]. This former research introduces an approach to segment or decode DNA sequences using a statistical language model. This model, for instance, segments a sequence in human genome as follows:

Sequence:
TGGGCGTGCGCTTGAAAAGAGCCTAAGAAGAGGGGGCGTCTG
GAAGGAACCGCAACGCCAAGGGAGGGTG

Segments:
TGGGCGTG/ C/ G/ CT/ TG/ AAAA/ G/ AGCCT/ AAGAA/
GAGGGGGCGTCTGGA/ AGGAA/ CC/ G/ CA/ A/ C/ GCCA/
AGGGAGGG/ TG/

By making this transformation, a DNA sequence is transformed to a document, and the words in this document are made up of segments of DNA called DNA words. Figure 2.6 from the study demonstrates this transformation with example documents.

Each known sequence that is attached to the database creates a document. After this mapping, inverted index and positional index can be created from DNA sequences.

Document 0, D0= “ATCG ATT ACC”;
Document 1 ,D1= “ATCG ACG AAA ACC”;
Document 2, D2 = “ATT ACG AAA ATTC ACC”;

Figure 2.6: Example DNA documents [1]

In this demonstration, there are three documents which are D0, D1, and D2 and they have 6 different words in total: "ATCG", "AAT", "ACC", "ACG", "AAA", and "ATTC". An inverted index is created by determining the documents in which these words appear, as shown in Figure 2.7.

word	Doc id
ATCG	D0,D1,
ATT	D0,D2
ACC	D0,D1,D2
ACG	D1,D2
AAA	D1,D2
ATTC	D2

Figure 2.7: Example DNA inverted index [1]

In addition, since the positions of words are known, the positional index of the documents can be created as shown in Figure 2.8. After this process, a new DNA sequence is processed and segmented, and its similarity is compared with existing documents (DNA sequences).

word	(Doc id, word position)
ATCG	(D0,0),(D1,0)
ATT	(D0,1),(D2,0)
ACC	(D0,2),(D1,3),(D2,4)
ACG	(D1,1),(D2,1)
AAA	(D1,2),(D2,2)
ATTC	(D2,3)

Figure 2.8: Example DNA positional index [1]

2.3 Snippet Generation

A snippet [13] is a small section of text that is displayed as a summary or a preview of a document, which gives an idea of whether the document contains the information desired to be accessed without reading the entire document. Snippets are often used in search engine results pages (SERP), where they serve as a preview or short description of the web page that matches a search query. Snippets help users decide to click and explore further by quickly understanding the content and relevance of the document. Moreover, sometimes users can find the information they are looking for from the snippet and do not need to enter the web page.

There are two types of snippets which are static and dynamic. Dynamic snippets are generated as query-dependent. Static snippets, on the other hand, are created as a summary of the information provided by the document, which makes them independent of the query. The title and metadata of the content can be used while creating this summary.

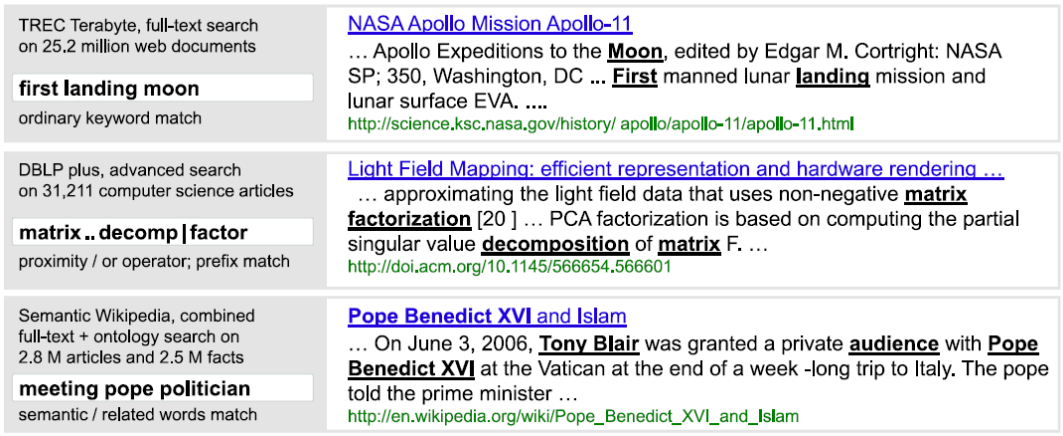

Figure 2.9: Snippet examples [2]

There are three examples of query-dependent snippets in Figure 2.9. The first example demonstrates a snippet, comprising two parts, generated for a typical keyword query. Because the query words are far from each other in the document, the snippet is shortened by using an ellipsis. The second case is an example of a combined proximity/or query snippet. The snippet can include only segments containing query words in close proximity. The third example presents a semantic query snippet that

involves nonliteral matches, such as matching "Tony Blair" with "politician." Nonliteral matches can be obtained by query expansion.

2.3.1 Approaches to Dynamic Snippet Generation

There are two approaches to create query-dependent snippets in terms of snippet quality: abstractive snippet generation and extractive snippet generation. Abstractive snippet generation creates brief summaries by rewriting sentences and phrases, while extractive snippet generation detects the relevant sentences from the documents and combines them into a summary.

2.3.1.1 Abstractive Snippet Generation

In this approach, major statements are rephrased using a distinct vocabulary from the original input. The basic approach is translating the content of the document statistically. A recent study [26] in neural network offers an alternative way to abstractive snippet generation. Instead of incorporating a neural network into a machine translation system, the model directly generates a translation from a source sentence. Furthermore, common methods for machine translation involve encoding the entire input sentence into a fixed-length vector and then decoding the translation from that vector. Conversely, the proposed approach extends the basic encoder-decoder model by allowing the model to search for input words or their annotations during the generation of each target word. This eliminates the need to encode the entire source sentence into a fixed-length vector and allows the model to focus only on relevant information to generate the next target word.

The main advantage of abstractive snippet generation is that it eliminates grammatical errors in the original sentences as it rewrites the sentences. Despite that most research focuses on extractive snippet generation because semantic representation, inference, and natural language generation are relatively harder than data-driven approaches [27].

2.3.1.2 Extractive Snippet Generation

Once the relevant documents are identified, the extractive snippet generation algorithm selects a portion of the document's content to be presented as the snippet. Since extractive snippet generation uses the original sentences, it guarantees that the user reaches the same sentences when entering the web page. However, grammatical errors in the original sentences directly affect the snippet quality. For the selection process, first of all, important parts of the document need to be determined, these important parts can be called candidate snippets. Each candidate snippet is scored and a summary is created from the highest-scoring candidates based on the required number of phrases. There are several ways to obtain candidate snippets:

- Key extraction employs supervised machine learning to identify core information. The main challenges in using supervised learning techniques are the need for training data and the potential bias towards the specific domain created for [28].
- Sentence compression ranks sentences based on importance, groups them by similarity, and selects the clusters with the highest score [29]. After ranking the documents by using a query and identifying relevant documents, the importance of sentences in a relevant document can be determined by searching the same query for each sentence as if it is a document [30].
- Parse-based approaches [31] split a document into parts which are semantically different from each other. Yet, this process is query-independent, and this may result in the candidate snippets not having all query words even if they are found in the document.

While these approaches to extractive snippet generation primarily emphasize the quality aspect of determining which snippets to display, there is a study [2] focusing on the efficiency aspect of computing high-quality snippets in the shortest possible time. This study makes two definitions, document-based snippet generation, and index-based snippet generation. In document-based approach, candidate snippets are created by using the positions of the matching query words. Due to the absence of positional information in candidate snippets, positional indexes for each document can

be created and this allows for quick retrieval of the positions of query words. However, this causes code duplication and it is time-consuming, error-prone, and harder to maintain [2].

In index-based approach, instead, there is a simplified and query-dependent positional index and it includes only the highest-ranked documents. Firstly, positional index is simplified by processing the query words from left to right and this is called operator inversion. Starting from the first and second query word, whenever an operator is applied to two posting lists, the resulting list will contain positions from the second posting list only. To illustrate, assume $L(q_i)$ as the posting list that includes all instances of the i^{th} query word in relevant documents. In the most basic form, this is the inverted index list of q_i. When applying the proximity search operator to q_i, $L(q_i)$ becomes a proximity inverted list of q_i, and so on. In addition to obtaining all valid positions of query words in a precomputed way before creating the snippet, this approach can be employed with various query operators, such as boolean search, phrase search, proximity search, prefix search, range search, fuzzy search, and semantic search [2].

2.3.2 Evaluation of Snippets

There are basically four approaches to evaluate snippets [32]:

- Eye tracking experiment aims to examine the behavior of the user by providing a user interface. In other words, this method needs a user interface and many volunteers or paid users to do this evaluation. Despite all this effort, the results may be open to interpretation. For example, if a snippet is of high quality, the user may be more likely to click on that site, but this is not certain. If the snippet quality is high and the user can get the information they are looking for from the snippet, there is no need to enter the website.
- Manual assessment needs lots of human resources. For this method, the whole corpus should be read for each query and the best snippet parts should be determined. Therefore, it requires long periods. Furthermore, this method may contain false evaluations that will be created by the human factor.

- A/B testing is similar to eye tracking, but the main difference is that it offers users two options and learn which one is better. The evaluation is done by changing variables one by one with this method.

- Automated quality measures are used to first determine the evaluation metrics and then evaluate the snippets automatically. There are two main advantages to this method. Firstly, this method is financially inexpensive, and secondly, the evaluation period is shorter compared to other approaches. However, it does not take user interaction into account and it is a drawback as user interaction is an important factor in determining snippet quality. Yet, automated quality measures are determined by leveraging findings from surveys that evaluate snippet quality based on user interactions. The general idea is that snippet quality depends on two primary factors: informativity and readability. Informativity refers to the snippet's ability to provide enough information about the entire document within the context of the query. Readability, on the other hand, pertains to how easily the snippet can be comprehended and understood. In this thesis, automated quality measures are used for snippet evaluation. The measures are chosen based on readability and informativity.

2.4 Video Search

Traditional video search consists of ranking videos by query words. With the increasing volume of video content available online, this simple video search is not enough for users to locate specific video segments, scenes, or moments of interest. A search result which gives a specific time range for a query can save time and effort compared to manually scanning through lengthy videos. Essentially, the specific time range or exact start time in video searches is a summary/snippet of a video. Similar to text snippets, the user experience improves as snippets become more common in videos.

There are many efforts to improve the user experience in video search results, generally by enriching the video descriptions and adding keywords/tags to the descriptions [33] [34]. The process involves making YouTube videos accessible within the annotation application, where metadata is contributed by the crowd. This information is

uploaded to YouTube in the form of captions and descriptions using YouTube Data API[2]. YouTube's speech recognition mechanisms handle the synchronization of the uploaded information by users. This feature significantly saves users' time and effort by eliminating the need to manually transcribe the spoken video content and determine the start and end times. Ultimately, the aim is to enrich video descriptions and tags and present a better user experience.

Although these works and the proposed method in this thesis use subtitles to improve user experience, they employ completely different approaches. The integration of LAIR into YouTube search aims to obtain snippets from videos using subtitles as documents and the BM25 algorithm for ranking. LAIR processes the subtitles and then generates a snippet video. LAIR processes the subtitles and generates snippet videos, without introducing any additional tags to the videos. The use of comments to add tags can have unintended consequences, as irrelevant comments may mislead ranking algorithms. Furthermore, this method introduces tags that are not query-dependent, which means that for certain queries, the inclusion of extra information may not yield desired results.

There is another study[3] that acquires a dataset by using YouTube Transcript/Subtitle API[4] and then develops models by applying NLP techniques. The study employs machine learning techniques, specifically natural language processing and topic analysis, to examine the current state of YouTube's recommendation system. Although the way of obtaining the dataset is similar to that of LAIR, the objectives of these two studies differ. LAIR generates snippet videos that provide a concise summary of whether a specific video is worth watching. On the other hand, this work suggests new videos to users that are related to the currently watched video.

For a query, several videos are ranked and recommended. However, as users watch videos from these recommendations, subsequent recommendations are influenced by the content of the viewed videos. In this study, each related video has a recommendation depth, and this is obtained by processing subtitles, applying NLP techniques, and employing topic modeling.

[2] https://developers.google.com/youtube/v3
[3] https://github.com/Alexander-Parker/youtube_nlp
[4] https://pypi.org/project/youtube-transcript-api/

Another study [35] obtains similar videos based on visual inputs, not verbal inputs. This paper focuses on Near-Duplicate Video Retrieval (NDVR). The proposed framework introduces a pre-processing step inspired by Multiple Sequence Alignment (MSA) of DNA sequences, where videos are represented as genomes, and MSA is employed to automatically cluster the video collection. According to experimental results, clustering-based approach is significantly faster than other state-of-the-art techniques that lack a preprocessing clustering step. There is an example shown in Figure 2.10. This approach can be used to recommend new videos based on the video currently being watched by a user.

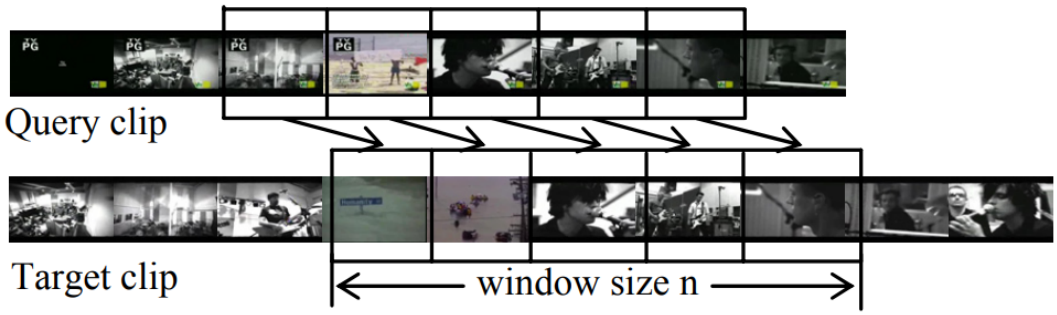

Figure 2.10: An example of MSA video retrieval

There is also another study [36] that searches within the video, such as LAIR, to improve the user experience. The study focuses on training a function that calculates the similarity between two components: text and video. Using this function, they assign rankings to videos or subtitles within the dataset based on their similarity to the query in text-to-video or video-to-text retrieval scenarios. The aim of the similarity function for video-subtitle pairs is to assign a high value when the video and subtitle are similar, and a low value when they are dissimilar. Achieving this requires precise representations for both the video and the subtitle.

In this study, word embeddings are created from subtitles using BERT (Bidirectional Encoder Representations from Transformers). In addition, video embeddings are created using methods such as feature and temporal information extraction. These embeddings are then used to compute similarity. In conclusion, this approach has the ability to process various features that are extracted from moments of a video.

While this work employs both video and text processing, LAIR primarily focuses on

text processing. Additionally, instead of using pre-made video-subtitle pairs, LAIR can take the entire video content and generate snippets of varying lengths depending on the query. This is achieved using local sequence alignment which is a new approach for snippet generation.

CHAPTER 3

LAIR - LOCAL ALIGNMENT FOR INFORMATION RETRIEVAL

This chapter presents the implementation details of LAIR, which is the method proposed in this thesis. First, the preparation steps required for using local sequence alignment in information retrieval are explained. Next, snippet generation is performed by using the outputs of a simple local sequence alignment integration. Thanks to the snippet, local sequence alignment can be visually evaluated in query search, and features that need improvement are identified. Finally, additional approaches such as query expansion and inverse document frequency are integrated into the local sequence alignment algorithm.

3.1 Sequence Alignment's Applicability to Information Retrieval

Sequence alignment algorithms are widely used in computer science, especially in bioinformatics. The main purpose of sequence alignment algorithms is to detect similar fragments in the two inputs. In bioinformatics, these two inputs are a query gene and target sequences. The query gene sequence is shorter than or equal to the query target sequence. As a result, using sequence alignment algorithms, fragments closest to the query sequence are found in the target sequence. Which sequence alignment method to use is also determined by the definition of the problem. For example, global sequence alignment can be preferred when one-to-one match detection is desired.

In information retrieval, there are basically two inputs: query and document. Again, as in bioinformatics, one of the search areas in information retrieval is to obtain the most relevant fragments by query in documents. As can be seen, there are two similar inputs in both bioinformatics and information retrieval, and the outputs of these inputs

are similar as well. While bioinformatics detects the fragments most similar to the query sequence in the target sequence, information retrieval identifies the fragments most relevant to the query within the document.

The most significant difference in problem definitions between these two areas is semantic relations. If sequence matching can be found in gene sequences, it can be said that these two genes are similar. However, in information retrieval, even if there is a match in word sequences, there may be differences in meaning due to factors such as words having multiple meanings or being used figuratively. Sequence alignment algorithms can be used in information retrieval, but they reduce the chance of making semantic inferences.

Another important point is that sequence alignments cannot be used for every problem in information retrieval. There are various query types in this field. Among them, the most suitable query types for sequence alignment algorithms are phrase queries and proximity queries because the query word order is important in these query types, and sequence alignment relies on the order to make matches. In other words, sequence alignments are not suitable for bag of words model. For instance, if there is a two-word query and the order of these query words is not important, all matches with the reverse order will be ignored, even though they could be considered good matches. It means that, if sequence alignment is desired for keyword queries, the algorithm must be run repeatedly for different combinations of query word order, which is far from being feasible. Therefore, in this thesis, local sequence alignment is used for snippet generation after relevant documents are obtained. By this way, snippets of different lengths are generated depending on the orders of the query words, while preserving the good matches on a document.

3.2 Adapting Local Sequence Alignment to Snippet Generation

When applying the local sequence alignment algorithm, if there is a match, then the score of the match is also calculated. Therefore, using local sequence alignment, document rankings can be made for phrase and proximity queries. Nevertheless, when the local sequence alignment algorithm is applied to all documents, it is not

applicable because it turns into a process that requires long computational times. The time complexities of information retrieval by using an inverted index(3.1) and local sequence alignment(3.2) is shown below:

$$O(|\mathbf{q}| * |\mathbf{L}|) \tag{3.1}$$

where |q| is the length of the query *and* |L| is the length of the average posting list

$$O(|\mathbf{q}| * \mathbf{D} * |\mathbf{D}|) \tag{3.2}$$

where D is the number of documents *and* |D| is the length of a document

For document ranking, a query search is done for all documents. On the other hand, snippet generation is done for only the documents which have high relevance to the query. Therefore, unlike document ranking, local sequence alignment may be a suitable method for snippet generation.

Note that there are different ways for snippet generation. The snippets created in this study are query-dependent because it is possible to attract the attention of users by including query words in the snippet as explained in detail in Chapter 4, and therefore query-dependent snippets are preferred today.

3.2.1 Document and Query Preprocessing Steps

In the preprocessing stage, the following steps are performed. Firstly, the stop words are removed. A word defined as a stop word has no contribution in terms of information gain. When the term frequency values are examined, it is generally observed that these words have high frequency in the document. Stop words vary across languages, and for this study, documents and queries are in English. For this reason, the English stop words in NLTK[1] is used. After this process, all words are converted to lowercase. All words must be either uppercase or lowercase, as both the positional index and local sequence alignment algorithm are case-sensitive. Finally, the stemming technique

[1] https://github.com/nltk/nltk

is applied. To obtain the base forms of words with different suffixes, this technique groups the words according to their base states. In this thesis, Porter stemming[2] is used for stemming.

Note that both the documents and the queries go through the same preprocessing steps because the slightest difference can lead to a lack of matches and to get the wrong result.

3.2.2 Local Sequence Alignment

In bioinformatics, genes are encoded with the initials of nucleotides. Since number of nucleotides is limited, it is highly likely that all these letters are present in both the target sequence and the query sequence. Nonetheless, this is not the case for documents and queries. For example, there may be three different words in the query, but there may be 100 different words in the document. Since the main purpose is to find these three words in the document, inputting all the irrelevant words to the local sequence alignment algorithm and looking at string equality for each word slows down the system. Nevertheless, with some optimization, local sequence alignment can process documents and query words by applying character mapping to them.

First, a unique character is assigned to each distinct query word. The query sequence is formed by replacing each query word with its corresponding character. After that, a similar process is done for the document with the help of the positional index, and finally, the document sequence is also created.

An example of this process can be seen in Figure 3.1. Suppose that the query obtained after some preprocessing is "allow sport activ". A character assignment is made for each different query word. A string is generated consisting of "-" signs with a length equal to the document's length. The positions of the query words in the document are determined by the positional index, and the corresponding indexes in the generated string are updated with the unique characters representing the query words. As a result of this process, both the document and query are transformed into a form similar to the query sequence and target sequence in bioinformatics.

[2] https://tartarus.org/martin/PorterStemmer/

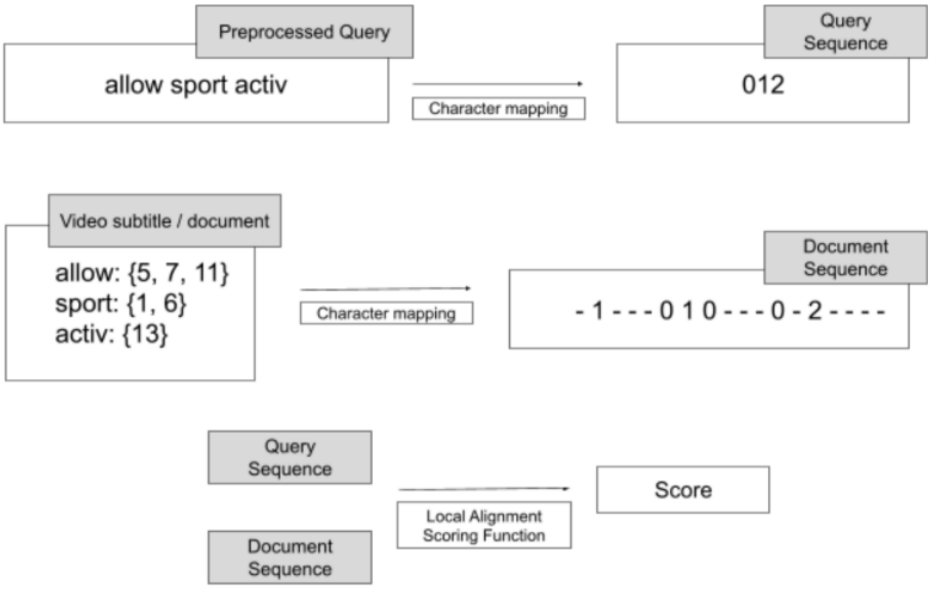

Figure 3.1: Character mappings

Changing documents and queries by character mappings has another benefit besides the reduction in computational time. Unlike documents and queries, the mapped documents and queries are compatible with the parameters of local sequence alignment functions in the Bio.pairwise2[3] module. Thus, in this thesis, built-in local sequence alignment functions from this module are used.

3.2.3 Integration of Basic Match Scores and Gap Penalties

Local sequence alignment is a dynamic programming algorithm and it chooses the best result by using a scoring function. There are four basic parameters in a scoring function. Two of these parameters are related to a matching strategy of the algorithm:

- match score (for identical characters)
- mismatch score (for non-identical characters)

Since local sequence alignment is a greedy algorithm, in some cases, it can sometimes result in matches with non-matching characters, which is known as a mismatch. This

[3] `https://biopython.org/docs/1.75/api/Bio.pairwise2.html`

mismatch is intentionally made to obtain a higher score for the subsequent match. Note that the score assigned to a mismatch is much smaller compared to the score assigned to a match.

The other two parameters are different types of gap penalties:

- opening a gap penalty
- extending a gap penalty

These penalties provide flexibility to the algorithm. Depending on the penalty, creating a gap without any matching may result in a higher score compared to matching with non-identical characters. However, it is not necessary for the penalties associated with opening a gap and extending a gap to be the same. Once a gap is opened, it can be extended until a match is found. During this extending process, the same character is still being tried to be matched. Therefore, instead of giving a penalty as if a new gap is being opened, a smaller penalty can be assigned for gap extension.

It is important to understand the problem while creating the scoring function and to set parameters accordingly. When matching query words with words in the document, matching non-identical characters implies that fewer query words appear in the resulting snippet. Consequently, the matching of non-identical characters is not promoted. Therefore, a high positive score is assigned to identical character matches, while a negative score is assigned to non-identical character matches. Although opening or extending a gap is not as bad as matching non-identical characters in the algorithm, the penalties associated with gaps should still be assigned as negative numbers. This ensures that gaps are not excessively favored and encourages the algorithm to prioritize matches over gaps. The specified parameter values are as follows:

Scoring function:

- match score: 20 points
- mismatch score: -1 point
- opening a gap penalty: -0.5 points
- extending a gap penalty: -0.1 points

The function suitable for these parameters in Bio.pairwise2 is align.localms function. In this function, the code "m" refers to “a match score is the score of identical chars, otherwise mismatch score” and “s” refers to “same open and extend gap penalties for both sequences”.

$$pairwise2.align.localms(doc\_seq, query\_seq, 20, -1, -0.5, -0.1) \tag{3.3}$$

At this point, visualization becomes necessary to track the improvements. Therefore, the snippet generation algorithm is implemented which uses the outputs of local sequence alignment.

3.2.4 Combining Local Sequence Alignment and Snippet Generation

Not only the alignment score but also the beginning and the end indexes are obtained as the outputs of pairwise2.align.localms. By using the beginning and the end indexes, the matching words and the number of repetitions of the matching words in this match are obtained. Then, this information is used to identify the starting and ending words in the document enabling us to generate a snippet from that section of the document.

At this point, there is a problem that all document’s words are preprocessed before this matching process. For instance, the word "activities" in the document is saved in the positional index as "activ" after preprocessing. If there is a word like "activity" in the query, it enters the matching algorithm as "activ" after preprocessing. As a result, if the word "activ" is the beginning or end index of the matching, the snippet cannot be generated because the corresponding word in the document is unknown. This problem is solved by preprocessing the document again.

Although the initial results show the integration of local sequence alignment into an information retrieval task, some shortcomings are noticeable. First of all, giving a snippet that consists of the first and the last match is poor in readability. For example, Figure 3.2 shows a snippet obtained from a website using the query "majesty retirement tennis".

The two words in the query (retire and tennis) are matched and even if it is a good

Figure 3.2: An example for a snippet with low readability

match, it cannot be a meaningful snippet because the matching is separated from its context. To address this issue, the snippet is generated by including the first word of the sentence where the match starts and the last word of the sentence where the match ends.

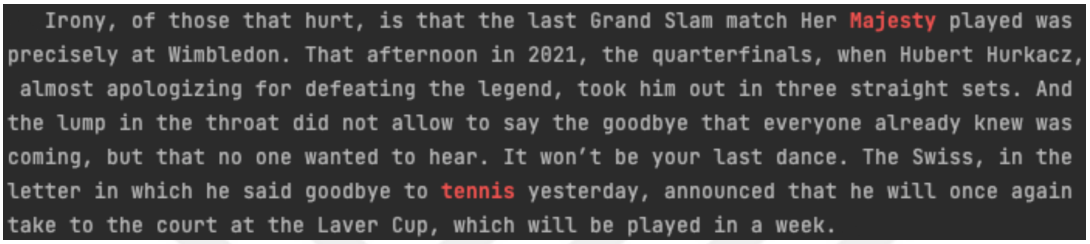

Figure 3.3: An example for a long snippet

Another problem is that, in some matches, there are too many words between the matching words in the snippet. In Google snippets, if there are too many words between two matching words, incomplete sentences are displayed, using an ellipsis to shorten snippets. In order to prevent long snippets, in this study, shorter snippets are generated by setting a maximum acceptable gap of 20 words between the matching words. To get a shorter match, the penalty for non-identical character matching can be set lower than the gap penalties. However, in this case, the algorithm can only find a match with the word "majesty", which is not the desired result. Nevertheless, with other improvements, such as matching with synonyms, and assigning a matching score based on the importance of the word, shorter and more meaningful snippets can be obtained for some query-document matching.

Finally, it is important to note that the algorithm may yield multiple matches with the same score. However, it does not necessarily mean that the same length snippet will always be generated for these matches. This is due to the removal of stop words during the preprocessing step. This situation is explained with the example below:

Case 1:

- **Sentence:** I wished my son could have been a good tennis player.
- **Tokens after removing stop words:** son, tennis, player

Case 2:

- **Sentence:** My son is a tennis player.
- **Tokens after removing stop words:** son, tennis, player

No matter which query is matched in the above example, the results will be the same for both cases, although the lengths of the original sentences are different. As a result of this observation, the document that has a shorter snippet is preferred if there is a tie between documents with the highest matching scores.

3.2.5 Integration of Synonyms Matching

The quality of the matching is not only dependent on the document or matching methods. If there are misspelled words in the query that are not detected during preprocessing, the incorrect word will be not found in the document resulting in a lower match score. The preprocessing step to prevent this situation is known as spelling correction. This process can be done by giving suggestions to the user or by ranking candidate words and choosing one of them automatically[37]. However, spelling correction is not used in this study because that is not the focus of the study. The queries are written in a way that does not require spelling correction so that the results of the study are not misleading.

Even if the queries are written correctly, the matching score may still be low unless query expansion is performed. Query expansion involves including similar words in the query to increase the possibility of matching. There are different ways for query expansion. In this study, query expansion is applied before the matching algorithm. This process is done automatically, without giving suggestions to the user. It means that the user is unaware of this process. For simplicity, only synonyms are added as part of the query expansion [38]. More complex query expansion methods can also

be developed, but the focus of this study is on how to integrate query expansion into local sequence alignment.

If only the relevance between the query and document were important, that is the order of query words did not matter, it would be sufficient to include the new words obtained with the query expansion into the query words and run the matching algorithm. However, in this study, a synonym for a word can only be used as a replacement for that word. Also, if the synonymous word has already matched, the original query word should no longer be included in the match.

This problem can also be solved by adopting an approach used in bioinformatics for information retrieval. A substitution matrix [39] is a collection of scores designed for aligning nucleotides or amino acids with each other. These scores generally represent the relative ease of one nucleotide or amino acid mutating or substituting for another, and they are used to measure similarity in sequence alignments.

Table 3.1 is an illustration of how the substitution matrix is used in this study. Suppose that A, B, C, and D are words in the original query. E is a synonym for A, and F is a synonym for C. No synonyms are found for B and D. The points to be obtained in the one-to-one matching of the words are the points in the diagonal. If a word matches its synonym, it gets 10 points. However, the match score of a word with other words and the synonyms of other words is equal to the non-identical matching score, that is, -1.

Table 3.1: A substitution matrix for synonyms

	A	B	C	D	E	F
A	+20					
B	-1	+20				
C	-1	-1	+20			
D	-1	-1	-1	+20		
E	+10	-1	-1	-1	+20	
F	-1	-1	+10	-1	-1	+20

The substitution matrix corresponds to the score_dict parameter in the Bio.pairwise2

module where matches are kept as keys and match scores as values. However, pairwise2.align.localms function[4] is not suitable with this parameter. Hence, it is replaced with a different function after substitution matrix integration as in the following:

$$pairwise2.align.localds(doc\_seq, query\_seq, score\_dict, -0.5, -0.1) \quad (3.4)$$

The function above has the code “d” instead of “m”. The code “d” refers to “a dictionary returns the score of any pair of characters”. Match and no-match scores are no longer parameters because the substitution matrix already has these scores.

```
  Yet it seems Verstappen could not just let it go. His name was trending
on Twitter ... when necessary. Indeed in recent years Hamilton returned a
... Hamilton, let the British driver pass at Monaco, because it was to
the benefit of the team.
```

Figure 3.4: An example for shortened snippet and synonym matching

The query used for the snippet seen in Figure 3.4 is “Verstappen did not let Perez pass”. Although the word "go" is not in the query, it is highlighted in the snippet. Note that, the two colors, blue and red, are used to easily see the beginning and end words of matches during the first trials. The first and last matches, which are the output of the local sequence alignment algorithm, are indicated in red. The in-between blue matches do not have to be the result of local sequence alignment. In this example, the words that local sequence alignment matches are “Verstappen”, one of the “let”s, and “pass”. The word “go” (+10) is obtained as a synonym for “pass” (+20). Since the word "go" has a much lower score than the word "pass", ending the matching with "pass" has a higher score. However, the query-related words between the first and the last word match are highlighted while creating the snippet. Also, in this example, the snippet is too long to match "pass", so the snippet has been shortened using an ellipsis.

[4] https://biopython.org/docs/1.75/api/Bio.pairwise2.html

3.2.6 Integration of IDF

As stated in Section 3.2.4, there can be more than one snippet with the same matching score for a document. One of the reasons for this is that there is no order of importance among the query words. Consequently, the matching score of each query word is +20 resulting in the same score even if different identical words are matched. However, some words may be more important than others. Therefore, IDF (Inverse Document Frequency) is integrated into the local sequence alignment algorithm. After this integration, the overall significance of a word across all documents is determined and the matching score for this word is specialized. The formula of IDF[5] is below:

$$idf(t, D) = log\left(\frac{N}{count(d \in D : t \in d)}\right) \tag{3.5}$$

IDF values are obtained by using TfidfVectorizer[6]. By normalizing the IDF values to 20, the final matching scores are obtained.

The same process is done for synonyms with a maximum value of 10. As a result, the scoring function has become more dynamic as the words in the query affect the values in the substitution matrix, and these differences are reflected in the scoring function. Suppose that there are three words in the query. There are two matches and suppose that the first and the second words match in the first match, while the first and the third words match in the other match and the gap penalties are the same in both matches. In this case, the IDF values of the words are essential. If the IDF of the second word is higher, it means that the matching score of the first match is higher.

Figure 3.5: An example of the effect of IDF

For instance, Figure 3.5 uses the same query and document as in Figure 3.4. What differentiates the results is the integration of IDF into local sequence alignment. Thanks to the IDF scores, the difference in scores between the words "go" and "pass"

[5] https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

[6] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

decreases, enabling the local sequence alignment algorithm to generate a shorter match.

CHAPTER 4

EXPERIMENTS AND EVALUATIONS

This chapter presents the evaluations of LAIR snippets and Google snippets in terms of snippet quality. It is explained how the experiment dataset is prepared before the experiments are carried out. In addition, the evaluation metrics used while making this comparison are explained in detail one by one. Finally, the results of these evaluation metrics are analyzed and evaluated.

4.1 Computational Resource

The creation of a positional index presents a bottleneck because it is a computationally intensive task in comparison to the other parts of the study. Yet, the overall process of the study is completed in a short time. Thus, the computer which is used in this study is sufficient and its features are as follows:

Processor: Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz 1.61 GHz RAM: 16.0 GB (15.8 GB usable)

Since the positional index of the corpus created from the 810 web pages can fit in the available RAM, subsequent calculations do not take long. If this work is done with a larger corpus, parallelization can be employed.

4.2 Limitations

The evaluation of this study is the most time-consuming part as it contains many limitations. The first limitation is finding a suitable dataset. Since local sequence align-

ment essentially performs a proximity query search, the dataset's query set should not consist of natural language queries. As a result, a suitable dataset for this work is not found, but there is a potential solution to create labeled datasets from scratch. Queries and their related documents can be obtained by using the approach proposed by H. Williams and J. Zobel [40], however this is not a reliable evaluation method. When the raw query is shortened, it is possible that better matches exist in other documents. In such cases, it becomes uncertain whether the document's low ranking is due to the query creation process or a failure in the algorithm.

Hence, the effect of local sequence alignment on the proximity query search cannot be evaluated due to the unavailability of a suitable dataset and queries. However, it is possible to evaluate snippets that are created using the results of local sequence alignment. For this purpose, it is decided to create a custom dataset. Various queries are prepared and executed on Google. The contents of the resulting top 10 websites together with their snippets are saved. The snippets for the same resulting web pages are generated using LAIR as well. Finally, a comparison is made between the Google results and the results of LAIR in terms of snippet quality.

The second and biggest limitation of snippet evaluation is that the quality of the snippet is subjective. At the beginning of the study, the results are interpreted as whether the query words are sequentially in the snippets and whether they are meaningful. However, in the evaluation stage of the work, a more objective point of view is needed. To get a set of rules for evaluation, several researches working on what matters in snippet generations are examined. As a result, an evaluation metric set is formed and this process is discussed in detail in Section 4.3.

The last issue is the way the queries are created. As mentioned in the first limitation, since there is no suitable proximity query list, the natural language query sets do not meet the needs of the study. Thus, by using Google Trends[1][2], the queries from trending news searches in the United States and the United Kingdom from 2017 to 2022 are used. There are 10 trending news topics for each year and for each country. That's why 120 trend news topics are collected in total. However, some searches are common for the United States and the United Kingdom, so there are 98 distinct

[1] https://trends.google.com/trends/yis/2022/US/
[2] https://trends.google.com/trends/yis/2022/GB/

queries in total. A sample of queries is shown in Table 4.1.

Table 4.1: Sample queries

Query	Year	Country
Queen Elizabeth passing	2022	UK & US
Kyle Rittenhouse Verdict	2021	US
Unemployment	2020	US
Equifax data breach settlement	2019	US
London Bridge attack	2017	UK

Note that, since there are a limited number of queries, queries are not pruned to obtain balanced data in terms of the word counts of queries. The average word count found in a query is 2.14. The reason why the average is close to 2 is that there are 60 two-word queries. The distribution of the queries according to the number of words they contain is shown in Figure 4.1.

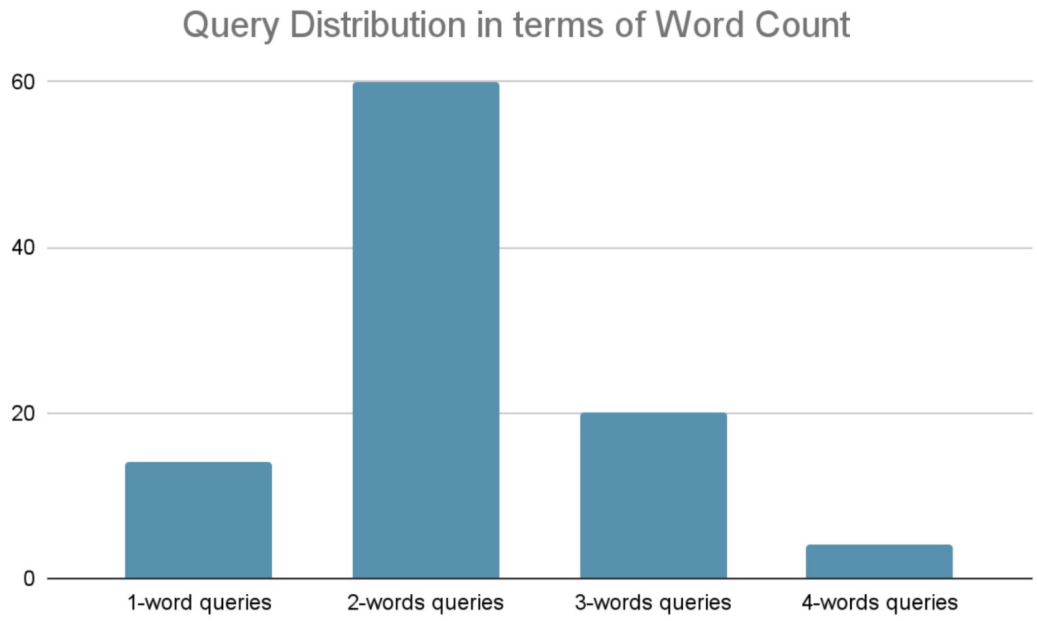

Figure 4.1: Query distribution in terms of word count

Google CSE[3] enables a bulk search on Google. Hence, after creating a list of queries,

[3] https://programmablesearchengine.google.com/

the ranked websites for each query are obtained. The sample result is shown in Table 4.2.

Table 4.2: A sample result from Google CSE search

Key	Value
Query	texas school shooting
URL	https://en.wikipedia.org/wiki/Robb_Elementary_School_shooting
Rank	2
Title	Robb Elementary School shooting - Wikipedia
Description	N/A
Snippet	On May 24, 2022, a mass shooting occurred at Robb Elementary School in Uvalde, Texas, United States, where 18-year-old Salvador Ramos, a former student at ...
HTML Snippet	On May 24, 2022, a mass shooting occurred at Robb Elementary School in Uvalde, Texas, United States, where 18-year-old Salvador Ramos, a former student at ...

Although the snippet and the HTML snippet are the same in content, the HTML snippet has an advantage. Thanks to HTML snippets, words that Google has marked as bold, referring to being highlighted, can be obtained. Highlighted words are important in evaluation metrics. On the other hand, the contents of the websites cannot be obtained, although snippets can be obtained from the websites ranked by Google for any query with Google CSE. Therefore, the contents are manually collected from all websites obtained with Google CSE, one by one, which is a time-consuming limitation of the study. Furthermore, the "title", "description", and "rank" columns are used for informational purposes. If the ranking scores of local sequence alignment are used to re-rank the documents, then the rank column is important for the comparison. This subject is discussed in detail in Chapter 6.

4.3 Evaluation Metrics

The text snippets presented in web search results provide users with a slice of page content that they can quickly scan to help inform their click decisions [41]. However, little is known about how these snippets are generated or how they relate to a user's search query. Similarly, there is not enough information about what metrics Google gives importance to when creating snippets.

There are many studies on which criteria are important for determining snippet quality, and generally, the eye tracking experiment, the manual assessment, the A/B testing, and the automated quality measures are used as evaluation tools [32]. Eye tracking is not used as an evaluation tool for this study because it is a laborious and costly way to evaluate the snippets. That approach requires a user interface and lots of participants which is not possible for this study. The manual assessment method would not be a realistic evaluation solution for this study either, because the whole corpus should be read for each query and the best snippet parts should be determined. This is a method requires long periods and may yield false evaluations as a result of the human factor. The third method, A/B testing is actually like eye tracking, and it would not be an appropriate evaluation method for the study either. As a result, automated quality measures are determined as the most appropriate method for this study since it is the fastest way to evaluate after deciding on evaluation metrics.

In the process of using automated quality measures, evaluation metrics are the most crucial aspects. Although there is no user interaction in this method, it makes great use of the user interactions of previous studies. Thus, the outputs of previous user interaction studies are used while deciding on the evaluation metrics.

Although there is not much information about Google's regular snippets, a study [42] has focused on identifying the important metrics for Google's "featured snippet". Google promotes this type of snippet for certain query searches. Figure 4.2 provides an example of a featured snippet. In Google's regular snippets, the order from top to bottom includes the web page URL, title, and snippet, while featured snippets are located above the URL.

A. Strzelecki and P. Rutecka [42] list the following features for featured snippets according to the result of their research:

- To ensure accuracy and reliability, Google selects direct response sources that meet its ranking criteria.
- Grammatically correct sentences have a better chance of creating featured snippets.
- Featured snippets are usually created from queries which are 2 to 4 words long.

Figure 4.2: An example of "featured snippet" for the query, "Verstappen did not let Perez pass"

- Snippet content is ideal for voice call responses and the snippet should be meaningful when it is read aloud.
- Keywords in the URL and response content are crucial to creating featured snippets, and more than half of the query words typically appear in both.

Some of these results give an idea of what metrics should be included in the snippet evaluation. Automated quality measures should be decided by focusing on informativeness and readability. First, not only the algorithm but also the length of the query affects the snippet quality. Long queries cause lower quality snippets in terms of informativeness and readability [32]. Since all the selected evaluation metrics look at the averages and the Google queries and the queries of this study are the same, the effects of this variable are constant for this study. Secondly, the fact that the titles are rich in terms of query words and that these words are highlighted is a critical factor affecting the snippet quality. However, the query words in the titles of the websites in the Google results are not highlighted, and if the titles had been included in the evaluation, the snippet quality of this study would have been higher than Google snippets. Nonetheless, since the main focus is evaluating the contribution of local sequence alignment for snippet generation, highlighted titles are not evaluated. In addition, if the average word length is too high, it may decrease readability. This explains the length of words that lie between words that local sequence alignment

provides a match for, rather than the impact of local sequence alignment on snippet quality. That is why the evaluation step does not include this metric. As a result of all these discussions, the metrics to be used in the evaluation step are finalized. The list of evaluation metrics is below:

- Number of query terms in snippets
- Snippet length in terms of characters
- Number of highlighted query terms per snippet
- Number of non-readable characters
- Number of ellipsis in snippets
- Number of no-highlighted-snippet
- Number of no-snippet web page

4.3.1 Number of Query Terms in Snippets

This metric will be one of the most important metrics because the more query terms included in a snippet, the more likely it is to be rich in informativeness. It means that after reading the snippet which is enriched by query terms, the user may not need to click the web page and read the whole document.

4.3.2 Snippet Length in terms of Characters

There is no consensus for snippet length, but there are common length assumptions. As a snippet gets longer, although the information gain does not increase after a certain point, users' interest in this snippet increases, and click rates increase [43]. Still, nowadays there is a hegemony of electronic devices with small screens. Therefore long snippets are not preferred because a single snippet taking up the entire screen affects the user experience negatively. It means that there is a trade-off between informativeness and readability here. As a result, it is argued that a snippet should be two lines [44] or 130-150 words long [43].

4.3.3 Number of Highlighted Query Terms per Snippet

It is important to differentiate the highlighted term and the query term. Even if a word is not related to the query term, it can be highlighted because there may be a relationship with the query. For instance, in this thesis, synonyms are used for increasing the chance of better matching. Even if the original word of the query does not appear in the snippet, the user may be satisfied by the snippet by finding the alternative word.

4.3.4 Number of Non-readable Characters

This metric is important for readability. If there are lots of non-readable characters in the snippet, users are less willing to read the snippet and they can perceive that the content of the corresponding web page has not the desired information. As a result, #, %, ^, @, *, <, >, +, -, =, ~, $, |, and \\ are selected as non-readable characters.

4.3.5 Number of Ellipsis in Snippets

If snippets contained truncated sentences or many fragmented sentences (text choppiness), users can perceive the quality of the results more negatively, regardless of length [45].

However, incomplete sentences are a way of shortening long snippets and making the sections containing query words attract the attention of the user.

The main reason for the formation of incomplete sentences is long queries. While creating the snippet, the more query words there are, the more relevance is tried to be established. Sometimes there can be too many words between the query words, but an algorithm can still consider a good match. Nonetheless, there can be another reason besides the large number of query words. Perhaps there are better and shorter matches, but the algorithm could not detect them. Therefore, when comparing the results of the two algorithms, there is a chance to see that one algorithm creates a better quality snippet than the other for the same query and document by looking at the number of incomplete sentences.

4.3.6 Number of No-highlighted-snippet

If there is no highlighted word in the snippet, it means that there is no match found with any query word. Even if this snippet has been created by providing a semantic and contextual match, this snippet may not be remarkable from the user's point of view as users cannot see any query or query-related words.

4.3.7 Number of No-snippet Web Page

Algorithms may not create a snippet if they cannot establish any relationship between the query words and the web contents. By comparing the two algorithms, it can be observed whether they can out-compete each other in this regard.

4.4 Experimental Results

The scoring function parameters are determined at the beginning of the study. The details are provided in Section 3.2.3. While determining the scoring function parameters, a hypothesis is used, and snippets are generated using a few custom queries, followed by an examination of snippet lengths. For the evaluation to be reliable, the scoring function parameters are determined without evaluation data. The results for all evaluation metrics are shown in Table 4.3.

Table 4.3: Evaluation metric results

	LAIR Result	Google Result
Number of query terms in snippets	1.511	1.405
Snippet length in terms of characters	143.637	152.821
Number of highlighted query terms per snippet	2.577	2.364
Number of non-readable characters	0.296	0.332
Number of ellipsis in snippets	1.004	2.381
Number of no-highlighted-snippet	0	35
Number of no-snippet web page	20	0

There is an interesting result for the "number of no-highlighted-snippet" metric. It can be seen that Google has created a snippet even though it doesn't match. In this evaluation, it is not considered whether these snippets that Google generates match the query in terms of meaning. However, after looking at the titles of some of the related snippets, we realized that a possible reason for this situation might be the matches in the titles.

The "number of no-highlighted snippet" metric on its own is not meaningful because there is uncertainty if LAIR can produce snippets containing highlighted words for these websites. At this point, examining the results of the "number of no-snippet web page" metric gives an idea about this. When looking at the result, LAIR can produce snippets for 790 of 810 websites, 790 snippets contain highlighted words. For 20 websites, it does not produce a snippet because it does not find any matches. Google has produced snippets for 810 of 810 websites, but 35 of them do not have highlighted words. When examining the situation manually, there can be seen that 20 websites are common to both Google and LAIR. However, even though there are no highlighted words on Google for 15 websites, LAIR does. At this point, it can be said that LAIR is more successful to obtain matching.

Note that snippet lengths can be shortened independently of the local sequence alignment algorithm as explained in Section 3.2.4. However, for the purpose of this evaluation stage, snippets are not shortened by using ellipsis because the length and content of the snippet generated with local sequence alignment are the focus of the evaluation. It has already been seen that there is no need for any extra snippet abbreviation due to the evaluation metric, "snippet length in terms of characters". Without the shortening process, LAIR snippets have an average length of 143.637 which is in the range of 130-150 words [43] which is a promising result. In addition, Google's snippets slightly exceed this range.

Although average snippet lengths in LAIR are shorter than Google snippets, by looking at the metric result of "number of query terms in snippets", it can be seen that the query terms dominate snippets. Note that the results would be considered misleading if we used proportions instead of amounts.

In short, this result shows that snippets produced with LAIR can contain more query

terms even though they are shorter. LAIR claims to obtain the optimum match for the selected parameters. In addition, while generating the snippet with LAIR, the first matching word's sentence is added to the snippet without shortening. In the same way, the sentence of the last matching word is also in the snippet. On the contrary, although Google's snippets are usually shortened from the beginning and the end, these snippets are longer and contain fewer query terms.

Like query terms in snippets, the number of highlighted terms is crucial, rather than the proportion. Before doing the evaluation, the highlighted snippets obtained with Google CSE are edited because the HTML brackets used for making words bold are common to the consecutive highlighted words. To make the calculation easier, after a preprocessing step, it is ensured that each word has separate bold brackets. Finally, LAIR showed better results in this evaluation metric as well. The number of highlighted query terms per snippet in LAIR is 2.577 and the result of Google for the same metric is 2.364. Comparing the results of this metric with a former metric result reveals another conclusion.

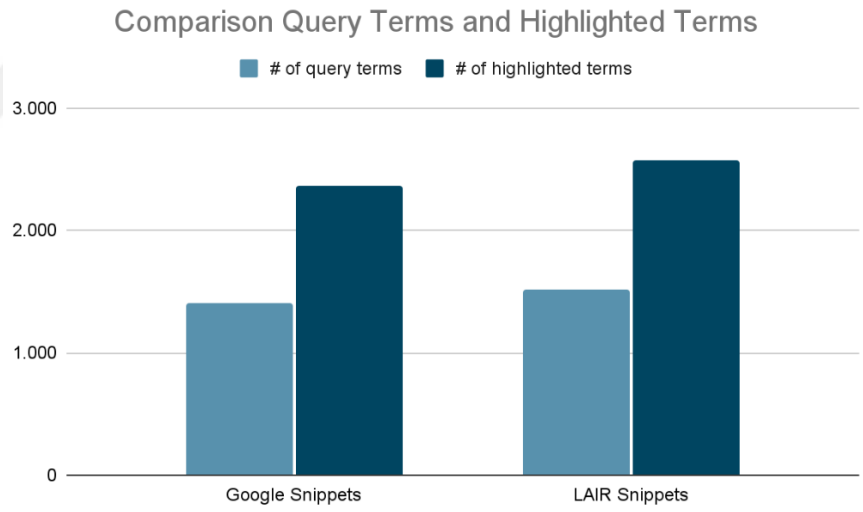

Figure 4.3: Comparison query terms and highlighted terms

As seen in Figure 4.3, the number of highlighted terms and the number of query terms are not equal in both Google and Lair snippets. This shows that both snippets also emphasize words other than query terms. The result shows that synonyms dominate some matches in LAIR although the coefficient of synonyms in the algorithm is low compared to the query words.

The main purpose of adding synonyms was to improve matches. Besides, we notice another benefit in the evaluation. Synonyms can also increase the user's interest in the snippet as it increases the number of highlighted words. If the synonyms were not integrated into the algorithm, Google's results would be better compared to highlighted words.

The metric, number of non-readable characters, indirectly demonstrates the success of the local sequence alignment algorithm. Non-readable characters are less likely as the query can find shorter matches without reducing the word count. However, the coincidence factor cannot be ignored for this metric. LAIR does not have the feature of detecting non-readable words during matching and reducing the score of the match. So in short matches, luckily there could be fewer non-readable characters, which would make LAIR's result better.

When the results of both algorithms are evaluated in terms of the number of incomplete sentences and fragments, it is seen that the result of LAIR is better. The minimum value is needed to be 1 because if there is no ellipsis, the snippet consists of a single fragment, which is a good result. As explained at the beginning of this chapter, since short snippets can be obtained with local sequence alignment, no extra shortening was made in the snippets during the evaluation, and therefore there is no ellipsis added by the algorithm. The reason why LAIR's result is 1.004, not 1, is because the website content naturally contains an ellipsis.

CHAPTER 5

ADAPTATION OF LAIR FOR YOUTUBE SEARCH

5.1 Problem and Motivation

Social media has recently become one of the most important communication tools. It has even begun to be preferred over some traditional communication tools. YouTube is an example of this case. In the past, people used to look at the broadcast flow of channels on television and developed watching habits accordingly. However, these habits no longer have to be shaped by the limited content because new video content has been produced fast. Millions of users generate vast amounts of data on social media every second. However, the credibility of its contents is doubtful due to the independence of freedom of expression, which differs from traditional news sources like newspapers and news channels [46].

One way of trying to get the right information out of a lot of unreliable information is to compare different resources and give importance to critical thinking [47]. However, since it is impossible to completely examine and compare an almost unlimited number of sources, people are reluctant to make comparisons and generally tend to believe in a source they read or watch. Furthermore, not only people who want to access information but also content producers do have problems because of having too many resources. As the number of resources increases, content producers also have problems with their content being visible. For instance, if a YouTube content producer is not conscious and does not feed his informationally competent video with tags and a meaningful title and description, the video is not displayed in YouTube searches. This is always an undesirable situation for the producer as well as for users who are willing to access this information.

In some cases, although the video content is too wide, it is desired to be summarized, but not all content can be summarized and certain parts of the content are promoted. However, if the content that the user wants to reach is not promoted, they may have some problems in reaching out to this content. For instance, if the video is a content of a news channel, the news channel can give the most impressive news in its video title, but there may be other news that the user is curious about.

Another problem is misleading information by resource producers to promote their content. Some content creators try to attract user attention by using misleading titles, descriptions, or cover photos. Similar to this, this situation also exists in web page searches, but it is easier to decide whether the website is useful or not by skimming or scanning techniques after entering the website, than by deciding whether it is a relevant place in the video.

Deciding that a video is search-related is important, and this saves users time. However, it can still be time-consuming for users to watch the whole video. For this reason, presenting the search-related portion to the user not only saves time but also provides an opportunity to review multiple video contents.

Finally, obtaining snippets from video content can also be useful in the entertainment field. It has become common and interesting today to create a meaningful new video from fragments obtained by clipping from many videos. For example, a popular song can be parodied by combining the words sung by different people obtained from many different videos. If a content producer, advancing with such a purpose, reaches the relevant parts of the relevant videos for each word group instantly with a query search, it can produce more content in a shorter time.

5.2 Creating the Corpus

The first step is to determine the topics. Some of the video contents have little or no conversation and the subtitles of such videos may not be sufficient. Therefore, the topics with more conversations are chosen. As a result, 668 video subtitles are obtained from 6 different topics which are listed in Table 5.1.

Table 5.1: Topics for the crawler

Topic ID	Topic
1	sports discussion
2	tonight show
3	movie review
4	tutorial
5	game review
6	news

First, a crawler is implemented. It obtains the titles and ids of the videos which are located in the URL, right after the "v=" URL parameter. For this process, the crawler uses the VideosSearch function from the youtube-search-python library[1]. Then the crawler gets the English subtitles in JSON format using the library, YouTube Transcript/Subtitle API[2]. After formatting the data, all subtitles are kept in one file in CSV format with 4 columns: "name", "topic", "id", and "subtitle" respectively. The "id" column holds the ids of YouTube videos and this information is crucial when making video suggestions. In the "topic" column, there can be one of the topics listed in Table 5.1. A excerpt from the subtitle data is shown in Table 5.2.

Table 5.2: A excerpt from the subtitle data

name	Parasite (2019)	Movie Review
topic	movie review
id	t_R-BIFl46Y
subtitle	["text": "hey everyone it's Mariana and I finally", "start": 0.03, "duration": 6.03, "text": "got the chance to see parasite the film", "start": 3.149, "duration": 5.431, "text": "a lot of people are calling the best of", "start": 6.06, "duration": 5.52, "text": "2019 I am here to tell you that the hype", "start": 8.58, "duration": 6.479, ... "text": "haven't already and I hope you're having", "start": 490.979, "duration": 4.771, "text": "an amazing day I will see you very soon", "start": 493.62, "duration": 5.419, "text": "in my next one bye", "start": 495.75, "duration": 3.289]

[1] https://pypi.org/project/youtube-search-python/
[2] https://pypi.org/project/youtube-transcript-api/

The YouTube Transcript/Subtitle API provides different ways to acquire subtitles. If there is a manually generated subtitle, it can be obtained. If it is not present, this API can produce an auto-generated subtitle. In addition to this, there can be many subtitle options whatever subtitles are available in different languages. Another feature is that it can translate a subtitle obtained in one language into another language. Since the information retrieval model is language-dependent, in this thesis, only one language, English is used. Note that, the adaptation of LAIR can be extended by changing preprocessing steps in a way that is language responsive.

If a video does not have a manually generated subtitle, and the auto-generated subtitle is only in the original language, then, the auto-translation option can be used to get English subtitles. Note that, the translation process may cause problems such as differences in vocabulary and grammar, idiomatic expressions, cultural references, ambiguity, subjectivity, and limitations of machine translation. Therefore, it causes low matching scores. Despite that, the translation option gives the chance that all videos can be searched for any query from any language. Yet, in this thesis, videos from other languages are not used so there is no need for a translation option.

Figure 5.1 shows the process of creating a corpus. Unlike classical documents, video subtitles, obtained with the help of the YouTube Transcript/Subtitle API, contain time sequence information as well. Time sequence information can be compared to page numbers in books. Both the time sequence information and the page number of a book give extra location information about the words they contain. In this study, two different approaches are used while creating a corpus. The first method makes use of time sequence information: each time sequence of video content is considered as a document. The second method uses the same method as in the implementation of LAIR on websites: entire video content is considered as a document.

5.3 Ranking Videos

There is no need to rank documents in the first stage of this work, which involves comparing Google snippets and LAIR snippets. When the query searches are made in the first part, the websites are already ranked. However, the video content obtained

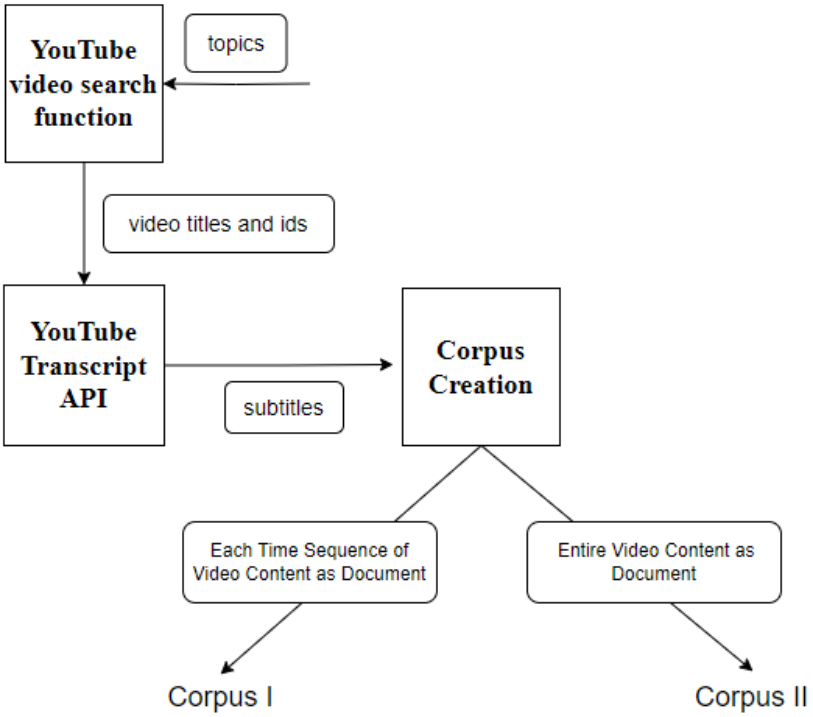

Figure 5.1: Corpus creation

through the YouTube API is based on topics rather than specific queries. For this reason, when conducting query searches, it is necessary to obtain relevant videos first. Then, LAIR can give results for a video.

There is no performance concern in getting relevant videos as the main goal here is to get to the state where LAIR can generate snippets. Thus, Okapi BM25 algorithm[3], which is an open-source implementation of BM25, is used to obtain relevant videos.

Corpus II was used to decide whether a video was relevant to a query. Before the relevant parts of the videos are obtained, a query is searched across all video subtitles. Afterward, LAIR is run for the first 5 video subtitles.

[3] http://www.soi.city.ac.uk/

5.4 Integration Local Sequence Alignment to Video Content

As seen in Figure 5.1, Corpus I and Corpus II are created from the same video subtitles. Therefore, the statistics in Table 5.3 are common to both corpuses. However, since every time sequence is converted to a document in Corpus I, the number of documents in Corpus I is 108979. In Corpus II, there are as many documents as subtitles, 668.

Table 5.3: Statistics for Corpus I and Corpus II

The number of subtitles	668
The number of tokens	17944
The number of words	694843
The ratio of word/subtitle	1040.2

Since two separate corpora are created with two different approaches, the use of the results obtained from the local sequence alignment algorithm also differs. Therefore, there are two different implementations for two separate corpora.

5.4.1 Using Corpus I: Each Time Sequence of Video Content as Document

When creating the corpus, for each subtitle, there are preprocessing steps. Each subtitle includes time sequences which include “text”, “start”, and “duration”. The number of documents created from a subtitle is equal to the number of time sequences in the subtitle, In addition, each document id is a combination of subtitle id, time sequence index, and start time. The combination of subtitle id and time sequence index makes the id unique. Furthermore, the contribution of "start" is to get the direct start moment of the matching document. These three parts are combined by putting the pipe character between them.

id: <subtitle_id | time_sequence_index | start>

In this method, duration is not used because duration is too short. For this reason, it may not be very meaningful if only the matching time sequence is shown. Hence,

it was preferred to show the match to the user by looking at the start time of the matching time sequences.

After obtaining the corpus, like in websites, there are preprocessing and positional index creation steps. After these steps, the local sequence alignment algorithm is run for each query. However, instead of generating snippets like in the case of websites, a URL along with the current timestamp for the relevant part corresponding to the query is provided here. The primary objective is to integrate local sequence alignment. Since each document consists of short time sequences, using the duration information of that time sequence is a convenient approach when creating a snippet.

Figure 5.2: A video matching with the query, "Donald Trump", for Corpus I

There is the matching video provided with the start time in URL[4] when the query is chosen as “Donald Trump”. A person who wants to watch the query-related part of the video can save time by watching the video from the moment shown in Figure 5.2. This video starts from 5 minutes and 10 seconds which is equal to the seconds in the URL.

[4] https://youtu.be/MjX1bY-EvS8?t=310

5.4.2 Using Corpus II: Entire Video Content as Document

Corpus II is very similar to corpus for websites except that there are new line characters in the corpus of websites. In Corpus II, on the other hand, the start and the duration information is displayed instead of the new line character.

For instance, after processing the subtitle in Table 5.2, its equivalent as a document in Corpus II is created as shown in Table 5.4. Since the text is too long, only the beginning and the end of the text are shown using the ellipsis.

Table 5.4: A row example for Corpus II

id	t_R-BIFl46Y
text	"hey everyone it's Mariana and I finally {0.03|6.03} got the chance to see parasite the film {3.149|5.431} a lot of people are calling the best of {6.06|5.52} 2019 I am here to tell you that the hype {8.58|6.479} ... haven't already and I hope you're having {490.979|4.771} an amazing day I will see you very soon {493.62|5.419} in my next one bye {495.75|3.289}"

In the preprocessing and positional index steps, start and duration pairs are ignored with the help of a regular expression. However, just like newlines are used when creating snippets on websites, this time the start and the duration information is used to obtain snippet videos. To demonstrate the process, a small part of a subtitle is shown below:

> "... parasite I feel very restored my faith {42.719\|7.381} in film has returned this is a wonderful {45.719\|7.02} movie and despite so many great films in {50.1\|4.709} his career already like memories of {52.739\|4.171} murder the host mother {54.809\|4.68} snowpiercer okhta this might actually be {56.91\|5.03} his best I thought this movie was {59.489\|5.731} virtually flawless I loved every minute {61.94\|5.499} of the film it's extremely funny in {65.22\|5.07} essence it's a social satire because {67.439\| 4.261} we're watching two very different {70.29\|3.06} families and how they perceive things {71.7\|3.57} the characters we follow who are sort of {73.35\|4.05} . . ."

Suppose that the query is "parasite movie character". Then the first matching word is parasite from the first time sequence and the last matching word is characters from

the last time sequence. Unlike creating snippets for web pages, creating snippets for videos is simple since there is information about the interval of the snippet. The snippet interval is calculated as:

Beginning of the snippet video:
first_matching_word_start_time

End of the snippet video:
last_matching_word_start_time + last_matching_word_duration_time

It means, for this example, the snippet starts from the second, 42.719 and the snippet ends in the second, 77.40 (73.35 + 4.05). Considering that, the numbers after the decimal are ignored when forming the URL. Furthermore, there is more than one match with the same score for a video. In that case, for simplicity, the implementation of the study forms a URL for only the first matching.

Note that, the output of this process is the video snippet's beginning and end time. Hence, the URL of the matched video must be changed. For instance, the video's URL[5] shown in Figure 5.3 has the parts which are "start=192" and "end=196". The video starts and ends according to this new URL and the figure shows the time, 3 minutes and 14 seconds which is the middle of the interval.

5.5 Discussion

There are some discussion threads regrading the results obtained using Corpus I and Corpus II. In both studies, the aim was to obtain the part of a video relevant to a query. Both studies look for the best matches between the query and the document using local sequence alignment. It should be noted that in this study, both Corpus I and Corpus II were exclusively utilized with the local sequence alignment algorithm. If these corpora were employed in conjunction with other ranking algorithms, a different scenario may arise.

- Corpus I: The document matching the query can be considered a snippet, as each video is created by obtaining multiple short documents using time se-

[5] https://www.youtube.com/embed/N8U52anaMlc?start=192&end=196

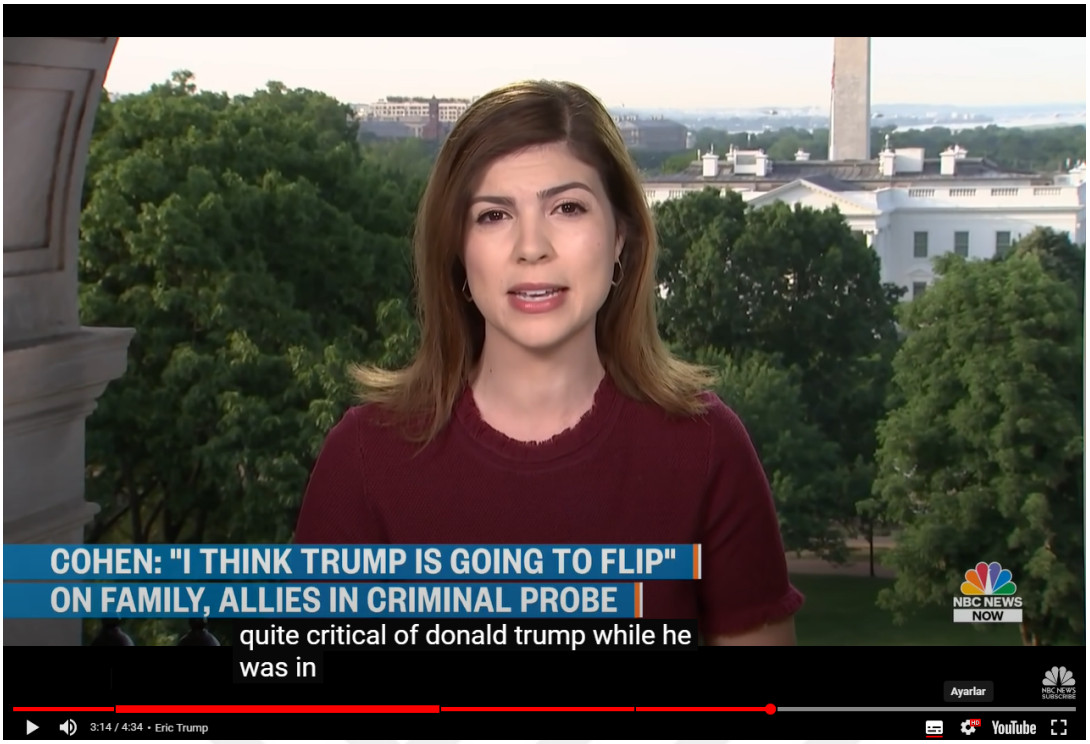

Figure 5.3: A video matching with the query, "Donald Trump", for Corpus II

quences from the subtitle. These small documents are very similar to candidate snippets used in snippet generation studies. While creating candidate snippets, the entire document is divided into small segments and a query search is performed on these segments. As a result, the relevance score is obtained for each segment and the most relevant snippet is selected from the candidate snippets. The size of these segments is an implementational choice. While creating Corpus I, video subtitles are segmented according to time sequences. The method of dividing the document into small segments used in snippet generation algorithms arises out of necessity because the other ranking algorithms can look at the whole document. Corpus I also has a structure that can be used with the other ranking algorithms at this point. It means, for example, if the Okapi BM25 algorithm is used instead of local sequence alignment, we may still obtain relevant time sequence information. Moreover, more than one sequence of a video can appear at the top of the rankings because they are considered different videos.

- Corpus II: If other relevant algorithms are used with this corpus, only informa-

tion regarding the relevance of a video to the query can be obtained. Therefore, other relevancy algorithms are not suitable for obtaining snippets using Corpus II.

Although Corpus I can be used with all ranking algorithms, local sequence alignment still offers many advantages of Corpus II:

- With local sequence alignment, a subtitle can be viewed as a whole and the relevant part can be obtained. Whereas, other ranking algorithms have to use Corpus I and, for each subtitle, all its segments must be ranked according to their relevance scores.
- In snippets obtained using Corpus I, the snippet size is equal to the segment length. The maximum snippet is determined before query searches and is equal to the longest segment. Whereas in Corpus II the snippet size is dynamic because it is query dependent. If the query-related part is long, the snippet is also long. However, a mathematical limit can be set for snippet sizes with the parameters specified in the scoring function of the local sequence alignment algorithm.
- Segmenting a subtitle while the query is not known yet can cause losing potential matches. This situation is explained through an example in Figure 5.4. In Corpus I, potential matches are lost by segmentation. Nonetheless, in Corpus II, this is not the case, since subtitles are considered as a whole. Note that in 1-word queries, there is no potential loss in Corpus I either because a query must contain at least 2 words for a snippet to be formed from the combination of at least two separate segments.

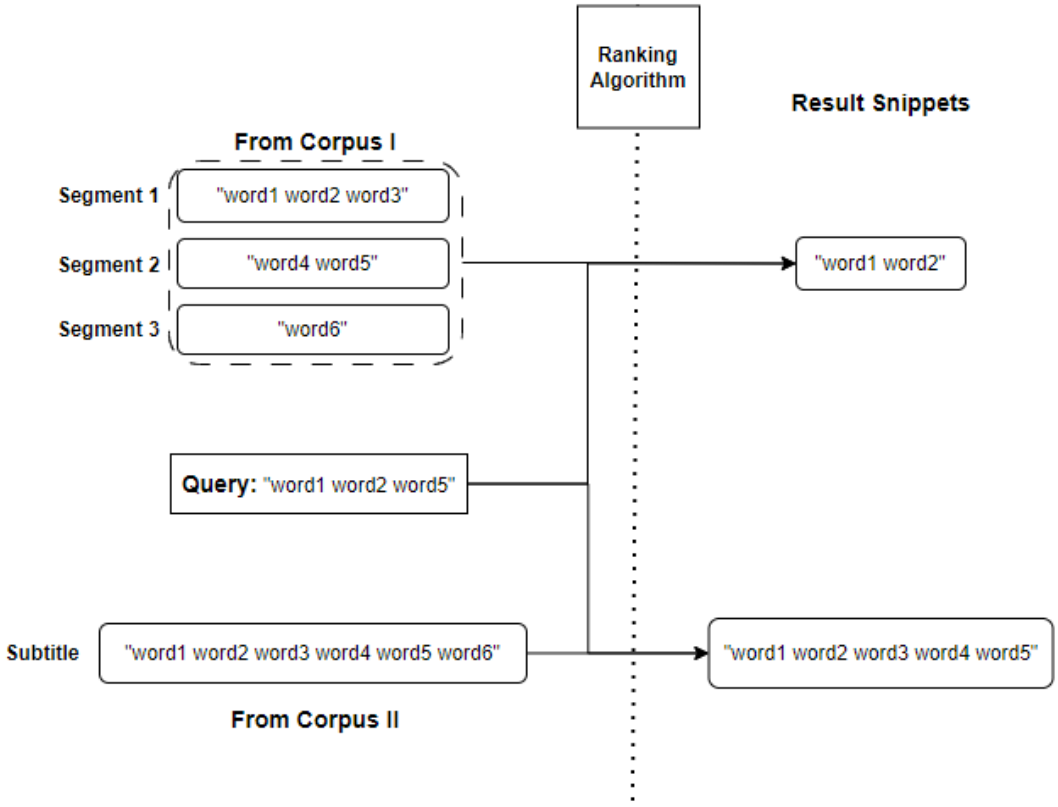

Figure 5.4: Effect of segmenting to snippet quality

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, the adaptation of the local sequence alignment algorithm to snippet generation is discussed. When the results were compared with Google snippets, LAIR was observed slightly better in the evaluation metrics.

Initially, a comparison was made between global sequence alignment, a similar algorithm, and local sequence alignment to determine the suitability of each for snippet generation. Then, the snippet generation process was carried out using the basic local sequence alignment algorithm. After that, query expansion and IDF approaches were integrated into the algorithm, respectively, and the results were examined manually.

Using local sequence alignment enables a more dynamic search compared to proximity query search. However, it was not possible to compare the results with the proximity query search in terms of relevancy since a suitable dataset could not be found for the proximity query search and there was not enough time to minimize human error while creating a new dataset. Therefore, snippets created using local sequence alignment were evaluated. To assess snippet quality, the choice of evaluation method was crucial. Therefore, previous works were reviewed. Due to the lack of resources, evaluation metrics based on user experience were employed. Google snippets and LAIR snippets were compared with the evaluation method called automated quality measures.

Furthermore, snippet generation can be incorporated not only in documents but also in YouTube videos. Basically, the algorithm still needs documents to work, and auto-generated subtitles of the videos are used for this purpose. Unlike documents, the output for video snippets differs from document snippets because the goal of video

snippets is actually to show a small part of the video. One of the key advantages of LAIR, which is the generation of snippets by searching for query words in the entire document, was emphasized over two different approaches in Chapter 5.

This thesis aims to both demonstrate the feasibility of local sequence alignment in snippet generation by highlighting its advantages and to increase the flexibility of the method by integrating commonly used approaches in information retrieval into local sequence alignment. However, the scores of the matching results from local sequence alignment were not used in this thesis. In future work, we intend to re-rank relevant documents using match scores and evaluate the results. Note that the order of query words is essential in local sequence alignment, and this factor may have a negative impact on the results. In other words, the re-ranking results could potentially be less favorable in terms of user experience.

As part of future work, other information retrieval approaches can be tried to be integrated into local sequence alignment or different methods can be tried for the approaches integrated in this study. For instance, this thesis demonstrates that query expansion can be done using local sequence alignment, but the method used in the thesis is to query expansion with synonyms. An alternative approach could involve BERT for query expansion, which could potentially yield improvements. BERT, developed by Google in 2018, is a pre-training method for natural language processing (NLP) [48]. Its applicability in different fields such as text classification, sentiment analysis, and question-answering also makes it popular. BERT is also used for query expansion [3]. Inspired by this study, we want to explore query expansion using BERT within the context of local sequence alignment.

REFERENCES

[1] L. Wang and K. Zhao, "A new dna alignment method based on inverted index," 06 2013.

[2] H. Bast and M. Celikik, "Efficient index-based snippet generation," *ACM Trans. Inf. Syst.*, vol. 32, apr 2014.

[3] Z. Zheng, K. Hui, B. He, X. Han, L. Sun, and A. Yates, "Bert-qe: contextualized query expansion for document re-ranking," *arXiv preprint arXiv:2009.07258*, 2020.

[4] H. K. Azad and A. Deepak, "A new approach for query expansion using wikipedia and wordnet," *Information Sciences*, vol. 492, pp. 147–163, 2019.

[5] B. Ballard, *Designing the mobile user experience*. John Wiley & Sons, 2007.

[6] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, p. 39–41, nov 1995.

[7] W.-K. Sung, *Algorithms in Bioinformatics*. Chapman and Hall/CRC, Nov. 2009.

[8] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[9] T. Smith and M. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[10] P. Savage and Q. Atkinson, "Automatic tune family identification by musical sequence alignment," 01 2015.

[11] Z. Su, B.-R. Ahn, K.-Y. Eom, M.-K. Kang, J.-P. Kim, and M.-K. Kim, "Plagiarism detection using the levenshtein distance and smith-waterman algorithm," pp. 569 – 569, 07 2008.

[12] S. Zhong, M. Shang, and Z. Deng, “A design of the inverted index based on web document comprehending,” *JCP*, vol. 6, pp. 664–670, 04 2011.

[13] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[14] V. Mohan, “Preprocessing techniques for text mining - an overview,” 02 2015.

[15] D. Sharma, “Article: Stemming algorithms: A comparative study and their analysis,” *International Journal of Applied Information Systems*, vol. 4, pp. 7–12, September 2012. Published by Foundation of Computer Science, New York, USA.

[16] M. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, pp. 130–137, Mar. 1980.

[17] V. Balakrishnan and L.-Y. Ethel, “Stemming and lemmatization: A comparison of retrieval performances,” *Lecture Notes on Software Engineering*, vol. 2, pp. 262–267, 01 2014.

[18] S. Pradha, M. N. Halgamuge, and N. T. Q. Vinh, “Effective text data preprocessing technique for sentiment analysis in social media data,” *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pp. 1–8, 2019.

[19] N. N. Amir Sjarif, N. F. Mohd Azmi, S. Chuprat, H. M. Sarkan, Y. Yahya, and S. M. Sam, “Sms spam message detection using term frequency-inverse document frequency and random forest algorithm,” *Procedia Computer Science*, vol. 161, pp. 509–515, 2019. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.

[20] S. Robertson and H. Zaragoza, “The probabilistic relevance framework: Bm25 and beyond,” *Found. Trends Inf. Retr.*, vol. 3, p. 333–389, apr 2009.

[21] A. Trotman, A. Puurula, and B. Burgess, “Improvements to bm25 and language models examined,” in *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS ’14, (New York, NY, USA), p. 58–65, Association for Computing Machinery, 2014.

[22] H. Azad and A. Deepak, "Query expansion techniques for information retrieval: A survey," *Information Processing Management*, vol. 56, pp. 1698–1735, 09 2019.

[23] E. M. Voorhees, "Query expansion using lexical-semantic relations," in *SIGIR '94* (B. W. Croft and C. J. van Rijsbergen, eds.), (London), pp. 61–69, Springer London, 1994.

[24] S. Kuzi, A. Shtok, and O. Kurland, "Query expansion using word embeddings," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, (New York, NY, USA), p. 1929–1932, Association for Computing Machinery, 2016.

[25] W. Liang, "Segmenting dna sequence into 'words'," 2013.

[26] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[27] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. Trippe, J. Gutiérrez, and K. Kochut, "Text summarization techniques: A brief survey," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, pp. 397–405, 07 2017.

[28] F. Boudin and E. Morin, "Keyphrase extraction for n-best reranking in multi-sentence compression," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Atlanta, Georgia), pp. 298–305, Association for Computational Linguistics, June 2013.

[29] K. Filippova, "Multi-sentence compression: Finding shortest paths in word graphs," in *Proceedings of the 23rd international conference on computational linguistics (Coling 2010)*, pp. 322–330, 2010.

[30] D. Pappas, G. Brokos, R. McDonald, and I. Androustopoulos, "Aueb at bioasq 7: Document and snippet retrieval," in *BioASQ*, 2019.

[31] R. Fernández, M. Frampton, J. Dowding, A. Adukuzhiyil, P. Ehlen, and S. Peters, "Identifying relevant phrases to summarize decisions in spoken meetings," pp. 78–81, 09 2008.

[32] D. Savenkov, P. Braslavski, and M. Lebedev, "Search snippet evaluation at yandex: Lessons learned and future directions," in *Multilingual and Multimodal Information Access Evaluation* (P. Forner, J. Gonzalo, J. Kekäläinen, M. Lalmas, and M. de Rijke, eds.), (Berlin, Heidelberg), pp. 14–25, Springer Berlin Heidelberg, 2011.

[33] R. Laiola Guimarães, P. Cesar, and D. Bulterman, ""let me comment on your video": Supporting personalized end-user comments within third-party online videos," pp. 253–260, 10 2012.

[34] J. P. Pinto and P. Viana, "Youtube timed metadata enrichment using a collaborative approach," in *Multimedia and Network Information Systems* (K. Choroś, M. Kopel, E. Kukla, and A. Siemiński, eds.), (Cham), pp. 131–141, Springer International Publishing, 2019.

[35] Y. Wang, M. Belkhatir, and B. Tahayna, "Near-duplicate video retrieval based on clustering by multiple sequence alignment," pp. 941–944, 10 2012.

[36] V. Gabeur, C. Sun, K. Alahari, and C. Schmid, "Multi-modal transformer for video retrieval," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 214–229, Springer International Publishing, 2020.

[37] D. Hládek, J. Staš, and M. Pleva, "Survey of automatic spelling correction," *Electronics*, vol. 9, no. 10, 2020.

[38] D. Pal, M. Mitra, and K. Datta, "Improving query expansion using wordnet," *J. Assoc. Inf. Sci. Technol.*, vol. 65, p. 2469–2478, dec 2014.

[39] S. Altschul, *Substitution Matrices*. 03 2008.

[40] H. Williams and J. Zobel, "Optimised phrase querying and browsing of large text databases," 07 2002.

[41] D. Hu, S. Jiang, R. E. Robertson, and C. Wilson, "Auditing the partisanship of google search snippets," in *The World Wide Web Conference*, WWW '19, (New York, NY, USA), p. 693–704, Association for Computing Machinery, 2019.

[42] A. Strzelecki and P. Rutecka, "Direct answers in google search results," *IEEE Access*, vol. 8, pp. 103642–103654, 06 2020.

[43] D. Maxwell, L. Azzopardi, and Y. Moshfeghi, "A study of snippet length and informativeness: Behaviour, performance and user experience," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, (New York, NY, USA), p. 135–144, Association for Computing Machinery, 2017.

[44] M. A. Hearst, *Search User Interfaces*. Cambridge University Press, Sept. 2009.

[45] D. E. Rose, D. Orr, and R. G. P. Kantamneni, "Summary attributes and perceived search quality," in *The Web Conference*, 2007.

[46] P. Meel and D. K. Vishwakarma, "Fake news, rumor, information pollution in social media and web: A contemporary survey of state-of-the-arts, challenges and opportunities," *Expert Systems with Applications*, vol. 153, p. 112986, 2020.

[47] A. L. M. Lemos, E. C. Bitencourt, and J. G. B. dos Santos, "Fake news as fake politics: the digital materialities of youtube misinformation videos about brazilian oil spill catastrophe," *Media, Culture & Society*, vol. 43, no. 5, pp. 886–905, 2021.

[48] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.