REDUCE++: UNSUPERVISED CONTENT-BASED APPROACH FOR DUPLICATE RESULT DETECTION IN SEARCH ENGINES

Zahraa Chreim[1], Hussein Hazimeh[1,2], Hassan Harb[1], Fouad Hannoun[2], Karl Daher[2], Elena Mugellini[2] and Omar Abou Khaled[2]

[1]Lebanese University, Faculty of Science, Beirut, Lebanon
[2]University of Applied Sciences of Western Switzerland, Fribourg, Switzerland

ABSTRACT

Search engines are among the most popular web services on the World Wide Web. They facilitate the process of finding information using a query-result mechanism. However, results returned by search engines contain many duplications. In this paper, we introduce a new content-type-based similarity computation method to address this problem. Our approach divides the webpage into different types of content, such as title, subtitles, body, etc. Then, we find for each type a suitable similarity measure. Next, we add the different calculated similarity scores to get the final similarity score between the two documents, using a weighted formula. Finally, we suggest a new graph-based algorithm to cluster search results according to their similarity. We empirically evaluated our results with the Agglomerative Clustering, and we achieved about 61% reduction in web pages, 0.2757 Silhouette coefficient, 0.1269 Davies Bouldin Score, and 85 Calinski Harabasz Score.

KEYWORDS

Information Retrieval, Websites Similarity, Graph Representation, Similarity Measures, Graph Kernel, Deduplication, Search Engines.

1. INTRODUCTION

Communication, Social Networking, News, Business, Ed-ucation, Online Shopping, and others, are some of the main reasons for the growing popularity of the Internet. With the Internet, there is almost no limit to what you can do online. If we take a deeper look, we find that an important common factor between these reasons is "access to information". Today, the Internet is clearly the fastest and easiest way to get any information we need and to answer any question we ask. Usually, people like to trust the information they get. In other words, when you generally search for something, you will automatically try to read many things about it first, and then get the final information you are satisfied with. This is exactly what the Internet does. It provides access to the World Wide Web, a service on top of it, which is a collection of content and information in the form of documents. So all you have to do is type one or more keywords, let us call it a search query, and through the search engines, you will get a bunch of results.

However, the Internet is one of the fastest-growing technologies in the world. As of January 2021, there have been 4.66 billion active internet users around the world. And the World Wide Web (WWW) is the fastest-growing part of the Internet, growing at a rate of 3,000 percent each year. Indeed, some sources have estimated that a new website is launched on the Internet every four seconds.

This means that with every search we do, we will get a large number of websites. Although these returned documents have different and varied content, they definitely have redundant information. Based on all this, today we pose the problem of duplication of documents in search engines, or in other words, the problem of high content similarity between them [1].

Reduce++ is based on our previous method REDUCE [2], which was based only on comparing the body (textual paragraphs) of web pages using only one similarity metric (cosine similarity).

2. PROBLEM DESCRIPTION

Similarity is a complex concept. It has been widely discussed in the linguistic, philosophical, engineering, information theory communities, etc. When it comes to defining it, in general, it is the quality or condition of having something in common. It is an aspect of correspondence, a feature or a point of resemblance, equivalence or likeness, in which things are similar.

Regarding documents similarity computation, it is one of the most important research topics in the field of information retrieval. The main goal of information retrieval systems is to provide users with the ”best possible information”, in which they are interested, for a given search query.

In recent years, many approaches and methods have been developed for calculating the similarity between two documents [3] [4]. But first, and in order to get the best results, it is necessary to ask how humans usually define how similar documents are. In general, most similarity measures assess the similarity of two documents based on the information content they share. This information content is represented by the text from which the document is formed. Therefore, the problem is referred to as text similarity detection.

Text similarity detection is a technique that determines how close two pieces of text are. It could be either a lexical similarity or a semantic similarity. The word level similarity (lexical similarity) does not take into account the meaning of the text; it only checks the common words. As documents are composed of words, two documents could appear very similar in case they have a great number of common words, regardless their order. On the other hand, semantic similarity focuses on the meaning of the whole sentence, paragraph or document, where they are treated as similar if they are semantically close and describe similar concepts. It is sensitive to sentence structure dependencies because differences in word order often go hand in hand with differences in meaning.

However, these techniques are insufficient, since treating a web document as a text-only document means that we are not actually working with the correct structure of the document, which is as already discussed, the first step that needs to be done.

Looking at current web pages, they include a title, banner, multiple navigation menus, a set of links, and main content [6]. This main content is divided into sections by headers (titles), subheadings (subtitles), and textual content (mainly paragraphs and lists) inside those sections.

In this work, we introduce an appropriate new model of document representation and similarity detection that takes into account the different content of the web document. As well as we benefit from multiple representations of texts, especially the graphical representation. Our model is called **REDUCE++:** REsult DUplication detection in Search Engines, which includes content-type-based similarity computation measures, in addition to a graph-based clustering algorithm. It

is composed of three main steps: Dataset Creation, Pairwise Similarity Computation, and Clustering.
The rest of this paper is organized as follows. In section III, we present the related work of different fields of the domain. We then define, in section IV, our model as Dataset, similarity measures used, and the clustering algorithm. After that, in section V, we discuss different experimental parameters and we show the final results. Finally, in section VI, we conclude this work and mention the future work.

3. RELATED WORK

A. Text Similarity

Calculating the similarity between the web documents is one of the mains tasks required by search engines in order to provide a better search experience. In the past two decades, a lot of research has been done to apply useful techniques for calculating similarities between text and others that determine a new display order for web pages.

According to [3], text similarity approaches can be divided into three categories: String-based, which operates on strings and characters, and measures the similarity according to the distance between them. Corpus-based, which determines the similarity between words according to information gained from large corpora. And Knowledge-based, which identifies the degree of similarity using information derived from semantic networks. In addition to some hybrid similarity methods that achieve the best performance.

To address the problem of information retrieval in big data environments, the authors of [7] proposed a semantic similarity measure using WordNet and a MapReduce algorithm. They index the query and compare it to the index of each document. They apply two MapReduce phases, to build document indexes from the words that compose it, and to calculate the similarity between these indexes and the query index.

Another work presented the basic aspects of evaluating the semantic similarity of texts [4]. They represent a document as a vector of features, which is a matrix of huge dimensionality. They reduce it using three methods: feature selection, feature extraction, and latent semantic indexing. Finally, they apply three similarity measures, word-to-word, document-to-document, and word-to-document, using the feature matrix and cosine similarity.

In order to detect redundant information in web crawlers and search engines, the authors of [8] proposes an innovative detection approach. Based on an ontology, they first summarize the text by providing the relevant sentences. Then, we calculate the semantic similarity from the obtained summary using WordNet. After that, a hash value is calculated using the winnowing algorithm. They therefore match this hash value with others using Dice Coefficient. Finally, a threshold is set to treat the documents either as similar or not.

Moreover, an idea introduced by [9] aims to find in a paragraph, the most similar match for each word of a first sentence, in a second sentence. They consider three similarity functions: string similarity, semantic word similarity, and common word order similarity to integrate syntactic information. The final text similarity is therefore derived by combining and normalizing all three functions to obtain a degree of similarity between 0 and 1.

B. Text to Graph

In order to model a text as a graph, there exist different methods and representations.

Unlike traditional Bag-of-Words models, and in the context of text categorization, the authors of [10] introduces a novel graph-based approach for document representation called Graph-of-Words (GoW). It encodes the relationships between the terms, questioning the term independence assumption. The nodes of the graph correspond to the terms of the document, and the edges capture co-occurrence relationships between them, within a sliding window of fixed size.

In addition, the authors of [11] described the standard methods for representing web documents using graphs, which are based on the adjacency of terms in a web document.

These representations are named: standard, simple, n-distance, n-simple distance, raw frequency, and normalized frequency.

According to the standard method, every word in the document, except stop words, becomes a node in the graph labeled with the term it represents. Then, if the word “A” immediately precedes word “b” somewhere in the section “S” of the document, a directed edge is added from “A” to “B” with edge label “S”. The sections are title, link, and text. Next, a stemming method is performed and the terms with the most frequent form are combined. Finally, the most infrequently occurring words are removed. Note that if a pair of terms appears together in more than one section, an edge is created for each section.

The simple representation is similar to the standard representation, except that it only looks at the text visible on the page (no title is checked). Therefore, it ignores the information about the “section” and does not label the edges between nodes.

Regarding the n-distance representation, a user-supplied parameter n is required. Unless words are separated by certain punctuation marks, it searches for n terms that follow a given term and connects them with an edge labeled with the distance between them. Here “distance” is related to the number of other terms that appear between the two terms in question.

Similar to n-distance, the n-simple distance representation does not label the edges, which means we only know that the “distance” between two connected terms is not more than n.
Regarding the raw frequency representation, it is similar to the simple representation but each node and edge is labeled with an additional frequency measure. The nodes indicate the number of times the associated term has appeared, and the edges indicate the number of times the two connected terms have appeared next to each other in the specified order. A problem with this representation is that large differences in document size can lead to skewed comparisons.

Finally, the normalized frequency representation, where instead of associating each node with the total number of times the corresponding term appears in the document, a normalized value is set in [0, 1] by dividing each node frequency value by the maximum node frequency value that occurs in the graph, as well as for the edges.

C. Graph Similarity

Graph similarity detection is a widely known field of research, where there exist several techniques.

In order to classify Web documents, the authors of [12] present them as graph data. These graphs are created from the content of the documents, preserving the inherent structure of the original documents and the structural relationships that exist between their terms. Regarding the similarity measure, they define a new distance metric based on the Maximum Common Sub-graph method, which outperforms the vector method in terms of execution time.

Other research explains that considering a document as a vase of words will prevent similarity measures from recognizing the contextual similarity of web documents [13]. Thus, they represent the documents as graphs using a composite model (Tag-Sensitive Graph and Context-Sensitive Graph), which preserves the advantages of the parent models but not the disadvantages. As for the distance measure, they use and optimize the Maximum Common Subgraph approach to take advantage of all the captured information.

In order to get the semantic relations between words, the authors of [14] propose a graph generation method that adapts algorithms of the theory of non-rigid 3D model analysis. These models can retrieve similar 3D topology models. Therefore, they create the graphs based on key-points, and semantically compare them based on the concepts of 3D invariant models.

Moreover, a graph-based semantic model is proposed in to represent the content of the documents, based on the DBpedia semantic network. The key of the approach is to combine a fine-grained relational vocabulary with measures of information theory, to identify degrees of correlation and levels of specificity between document concepts. After modeling the documents, they perform a matching problem based on Graph Edit Distance technology to find the degrees of similarity.

In order to better rank search query results, a new information retrieval system is developed [16], which represents a document as traditional bag of words and conceptual graph. The latter aims to integrate information about the concepts in the text and their relationships. Then, a similarity measure that combines conceptual and relational similarities is provided to measure the similarity between two documents.

4. METHODOLOGY

In this section, we introduce REDUCE++ in details. The full process is illustrated in (Algorithm 1) and (Fig. 1), in which we show how the query-search-reduce cycle flows.

```
Algorithm 1 REDUCE++
Require: Set of Search Query Results $S = (P_1, \dots, P_n)$
1: Initialize: Similarities = [ ]
2: for each $(P_i, P_j) \in S$ | i $\neq j$ do
3:     $Sim_{ij} = 0$
4:     for $Content\ C, Weight\ W_C, Similarity\_Measure\ M_C$ do
5:         $Sim_{ij} += W_C * M_C(P_i(C), P_j(C))$
6:     end for
7:     $Similarities \leftarrow Similarities \cup Sim_{ij}$
8: end for
9: return $Graph\_Cluster(Similarities)$
```

In our experiments, we generated our own dataset, where we carefully decided in what form we need it to be, what features to include, and how it should be pre-processed. In general, the choice of a dataset is critical since the evaluation and validation of any work depend on it. In case this dataset was not compatible with the implemented model and the objectives of the work, this may lead to incorrect results, and vice versa.

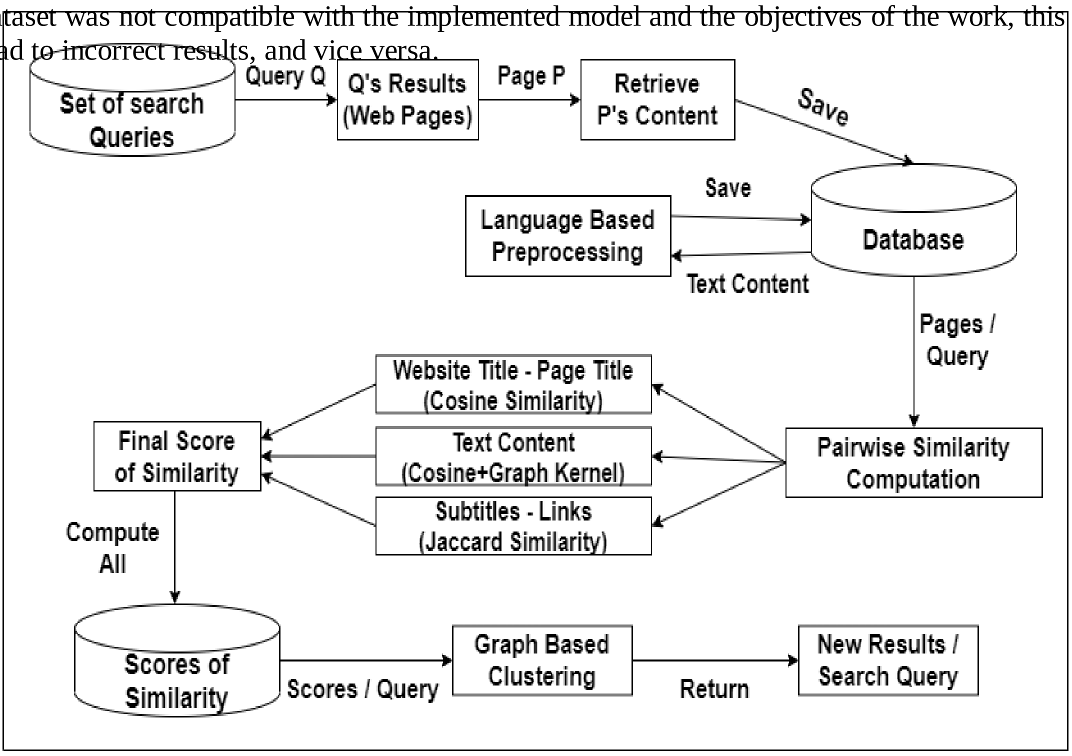
Set of search Queries
Query Q
Q's Results (Web Pages)
Page P
Retrieve P's Content
Save
Save
Language Based Preprocessing
Database
Text Content
Pages / Query
Website Title - Page Title (Cosine Similarity)
Final Score of Similarity
Text Content (Cosine+Graph Kernel)
Pairwise Similarity Computation
Subtitles - Links (Jaccard Similarity)
Compute All
Scores of Similarity
Scores / Query
Graph Based Clustering
Return
New Results / Search Query

Figure 1. Full system architecture

1) Data Collection: In this study, we collected the data from Google. It contains four different search query domains: healthcare, history, science, and general queries. In addition, we have included three different languages: English, French and Arabic. This diversity aims to avoid limiting our work to specific research topics or languages. On the contrary, our main goal is to demonstrate the effectiveness of this project by generalizing it, so that it can be used and applied in as many fields as possible.

In order to collect the data, we used Selenium WebDriver [17]. Selenium is a web framework that allows you to run tests across browsers, especially for testing web-based applications to verify that they are working as expected. It supports all major browsers like Chrome, Firefox, Safari, etc. In addition, using Selenium, you can choose which programming language you would like to use to create your test scripts.

Regarding the data, we have prepared 150 different queries, 102 in English, 24 in French, and 24 in Arabic. Then, for each query, we have extracted the first ten pages of its search results. This search result usually consists of 90 to 100 web pages. The dataset characteristics are described in (Table I).

2) Feature Engineering: From these web pages, and as we previously discussed the importance of taking into account the different content of a web page (Section. II), we extracted, in addition to the search domain, language and URL, the website title, web page title, subtitles and links, and the body text content, as shown in (Fig. 2). Next, we explain why we chose each feature.

i. Titles: Usually, the purpose of a title in a document is

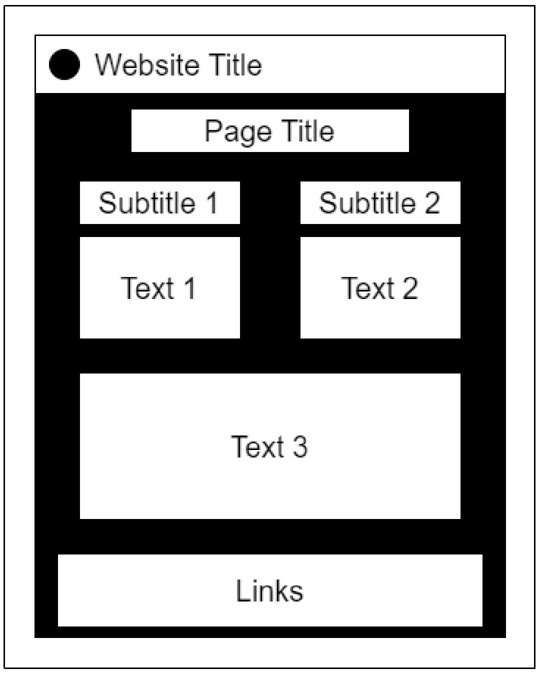

Figure 2. General Web Page Structure

simply to provide general information about the topic to be covered. Therefore, the similarity of the titles means that there is a high probability of high similarity between the two documents.

ii. Subtitles: Subtitles help to break down the text into

Table I Dataset Characteristics

Database Attributes (Features)								
Language	Domain	Query	URL	Body Text	Website Title	Page Title	List of Subtitles	List of URLs

Queries / Language			Queries / Domain			
English	French	Arabic	General	Healthcare	Science	History
102	24	24	52	30	39	29

Records / Language			Records / Domain			
English	French	Arabic	General	Healthcare	Science	History
7494	4276	2339	4903	2868	3620	2718

smaller chunks. Moreover, like any regular title, they provide a quick and easy guide to finding out what the content in that specific piece of text is about. Thus, when we find high similarity between two lists of subtitles of two different documents, this increases their total similarity score.

iii. Hyperlinks: Websites usually use hyperlinks as a way to navigate through online content and to point to other websites. The main goal of using hyperlinks is to provide recommendations to the users for a wealth of high-quality information related to what they are currently reading. This means that when two web documents link to same websites, they are probably sharing similar content.

iv. Body Text: Body text is simply the main content of the web document that needs to be compared, either syntactically or semantically. And this is actually the most important feature.

In total, we generated a dataset containing 14,109 rows of data.

3) Data Preprocessing: Data preprocessing is a crucial data mining technique that can directly affect the success or failure of any application. In fact, as we manage our code quality, data quality should follow similar principles, since data usually contain irrelevant, redundant, or unreliable information. For this reason, data pre-processing is used to convert the raw data into a useful and efficient format. This is done by cleaning the missing values, attribute values themselves, noisy patterns, and sometimes outliers and duplicate patterns, depending on every application.

In our work, we have implemented two different pre-processing processes, one syntactic and the other semantic. These two processes are only applied to the body text content of the web document, as this is the only content that needs to be cleaned well before being compared.

For syntactic pre-processing, it includes the usual pre-processing steps, namely: the elimination of upper case letters, non-ASCII characters, numbers, and punctuation. As for the semantic pre-processing, in addition to the above steps, stop words for each language are removed. Then, the text is enriched by semantic networks, based on the synonyms of words using dictionaries and lexical databases of the English, French and Arabic languages. This step is called lemmatization, where each word in the document is transformed into its root form. Using this process, we can analyze the inflected and different forms of a word as a single item.

B. Pairwise Similarity Computation

To calculate the similarity between two websites and to detect whether they are providing the same information or not, we have decided to use a specific similarity metric for each content in the web document, that we find suitable. This process is illustrated in (Fig. 3).

1) Similarity Measures:

i. **Website Title & Page Title**: Since the website and page titles are small sentences, the best metric we have found suitable for calculating their similarity between two websites is the Cosine Similarity. Cosine similarity is a popular NLP method to assess the similarity of two words or sentences [18] [19]. It is often used in the field of text analysis. Its main concept is to calculate the similarity between two vectors by measuring the cosine of the angle between them. The smaller the angle, the closer the cosine is to ”1” and the more the websites point in the same direction, so they are considered similar.

ii. **Links & Subtitles:** Each web page contains a list of links and a list of subtitles. In order to measure how similar two web pages are in terms of these lists, we used the Jaccard Similarity Coefficient [20] [21]. Also known as the Jaccard Index, this is a statistic widely used in computer science, ecology, genomics, and other sciences that aims to assess the similarity and diversity of sample sets. It is calculated by dividing the size of the intersection by the size of the union of the sample sets. The more elements the two sets have in common, the more similar they are considered.

iii. **Body Text:** When talking about high content similarity between documents and web pages, it means that we are directly dealing with the similarity of the content of the main body text. For this reason, we must first define how texts are represented.

a. **Bag-of-Words Representation of Text**: On the one hand, since documents are made up of words, most of the previously developed methods are text representations based on words to compute text similarity [22] [23]. The main idea is to represent documents as

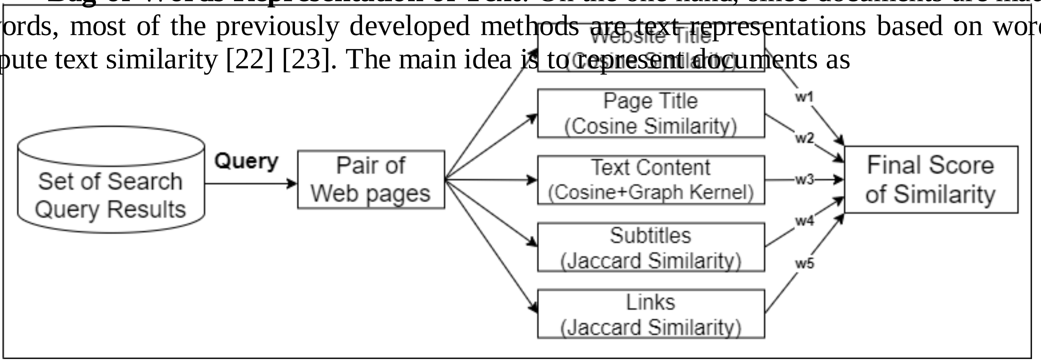
Set of Search Query Results
Query
Pair of Web pages
Website Title (Cosine Similarity)
Page Title (Cosine Similarity)
Text Content (Cosine+Graph Kernel)
Subtitles (Jaccard Similarity)
Links (Jaccard Similarity)
w1
w2
w3
w4
w5
Final Score of Similarity

Figure 3. Pairwise Similarity Computation

vectors of features, and compare them by measuring the distance or similarity between these features. This is known as the Vector Space Model [24]. Also called Term Vector Model, it is a straightforward algebraic model for representing text documents as vectors of identifiers. It is used in a wide variety of fields, and its first use was in the SMART Information Retrieval System.

A vector V can be expressed as a sum of elements such as: $V = a_{i1}v_{i1} + a_{i2}v_{i2} + ... + a_{in}v_{in}$, Where a_k are called scalars or weights and v_{in} as the components or elements. The weights usually represent the term frequency in the document.

Thus, the documents are a collection that can be viewed as a set of vectors in vector space, where each dimension of each document's vector corresponds to a single word and represents how often this word appears in the text. Now, we must decide which vector-based similarity measure to use. If we look at these vector in an axis model, we find the angel O represents the closeness of two vector documents. This value can be calculated by the cosine of the angel O. For this reason, we have used the Cosine Similarity Measure [18] [19] as a first method for calculating text similarity.

Cosine similarity is a metric often used to measure document similarity in text analysis. It is very useful for classifying data according to their similarity weight. As a definition, it is equal to the cosine of the angle between two non-zero vectors of an internal product space. The smaller the angle, the closer the cosine is to "1" and the more the documents are pointing in the same direction, so they are considered similar. The cosine similarity is advantageous because even though the two similar documents far apart by the Euclidean distance due to their size, they could still have a smaller angle between them, and therefore a higher similarity. In addition, it allows documents with partial match to be also identified.

However, Cosine Similarity treats the documents as bag of words. This causes the loss of the positional information about the terms. In other words, the Vector Space Model discards information such as the order in which the terms appear, where in the document the terms appear, how close the terms are to each other, and so on. For this reasons, other types of similarity metrics use different representations of text, such as Graph representation [25].

b. Graphical Representation of Text:

Several methods are used for representing web document content as graphs. For our graph-based similarity measure, we have created graph representations from the documents using the Graph-of-Words method implemented in [10]. First, we split the text into sentences. Then, we slide a window across each sentence. This window has a specific size, which means that the words within these windows have a strong relationship that must be taken into account in the representation. To do that, we add a directed edge to the graph, for each pair of directly and indirectly consecutive words, when sliding the windows. Finally, we end up with a graph that includes all the semantically related word pairs in the document. Note that for a given word, we create only a single node for it even if it appears more than once in the text. Thus, each node in the graph represents a unique word and is labelled with a unique label, which is the term it represents.

In order to measure the similarity between two graphs, we have explored many methods and approaches, such as Maximum Common Subgraph [13], Graph Edit Distance [26], Graph Matching [27], Graph Kernels [28], etc. Finally, we have decided to use Graph Kernel Methods.

A graph kernel is a function that computes the inner product of graphs. In other words, they are functions that measure the similarity of pairs of graphs. They are used in different real-life applications, such as bioinformatics, chemo-informatics, social network analysis, and others. Graph kernels allow kernelized learning algorithms such as support vector machines to work directly on graphs, without having to do feature extraction to transform them to fixed-length, real-valued feature vectors, which is what we exactly need.

To do that, we have used the already implemented Graphlet Sampling Kernel in the GraKel Python package [29]. According to the documentation, the graphlet sampling kernel decomposes graphs into graphlets (i.e. small subgraphs with k nodes where K ϵ 2 3; 4, …). Then, it counts the matching graphlets in the input graphs, which in our case are the two graph representations of two documents. Explicit feature maps compute the graphlet kernel, by first computing the representation of each graph in the feature space. The kernel value is then calculated by multiplying the two feature vectors. This kernel is a value between 0 and 1. The closer the value is to 1, the more similar the two input graphs are.

1) Final Weighted Formula: After computing the different scores of similarity for each content type of the webpage, we added them together using a weighted formula, to obtain the final similarity score

$$FinalScore = \sum_{ContentType\ C} Weight_C * Similarity_C \quad (1)$$

In order to set the weights for each content type, we consulted an expert to help us determine these values. These values are sensitive and have a significant impact on the results, as they determine the effect of each content type on the final similarity score. During the consultation, we tried to study and analyze these weights on the ground, so we tried to visually see which one really has a big impact, so it should have a higher value. Finally, we defined them as shown in (Table II).

Table II Weights Assignment

Website Title	Page Title	Subtitles	Links	Body Cosine	Body Graph
2.5 %	7.5 %	7.5 %	2.5 %	40 %	40 %

C. Clustering

In order to regroup the similar webpages together, we should cluster the data. Clustering is an unsupervised machine-learning algorithm [30] [31] [32], defined as the task of grouping a set of objects so that those in the same group, called a cluster, are more similar to each other than to those in other clusters. The main goal of clustering is therefore to discover a new set of classes, in which each group is of interest in itself.

Formally, the clustering structure is represented as a set of subsets
$C = C_1, \ldots, C_k$ of S, such that: $S = \bigcup_{i=1}^{k} C_i$ and $C_i \cap C_j = \emptyset$ for $i \neq j$
As a result, each element in *S* belongs to exactly one and only one subset.

In our work, after computing the similarity score for each pair of webpages for each query, we obtained a data in the form of a matrix, a pairwise similarity matrix, which is shown in (Eq. 2).

$$SimilarityMatrix = \begin{pmatrix} 1 & S_{12} & \ldots & S_{1n} \\ S_{21} & 1 & \ldots & S_{2n} \\ \vdots & & \ddots & \\ S_{n1} & S_{n2} & \ldots & 1 \end{pmatrix} \qquad (2)$$

To cluster the data of this matrix, and based on the importance of graph data structures previously discussed, we have implemented a graph-based clustering method.

The process steps are the following. First, we initialize an undirected graph G and a threshold t, above which we consider two websites as similar. Then, the method starts adding each pair of webpages as an edge to G, if their score of similarity is greater than or equal to t. Finally, we find all the connected components of G. In an undirected graph, a connected component is its maximum connected sub-graph, where each vertex belongs to single connected component, as does each edge. Each connected component represents a group of webpages that are all similar with a score of similarity t. Therefore, each one is considered as a separate cluster.

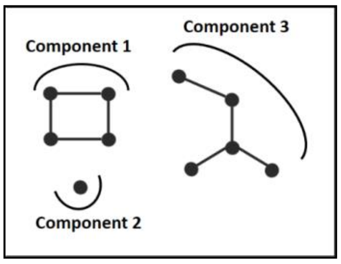

Figure 4. Graph's connected components scenario

A. Webpages Reduction

In this work, we introduced a model that aims to detect the duplicated web pages resulting from a search query, which give the user the same information. Webpages reduction is thus the main goal of this project. In order to evaluate and validate our results, we must show that the model is able to reduce the number of pages. In addition, since the clustering algorithm takes a similarity threshold as a parameter, it must also be evaluated. Varying a parameter in the same clustering algorithm is actually known as ”Relative Cluster Validation”, which is one of the cluster validation techniques. Usually, the number of clusters k is changed to determine the optimal one.

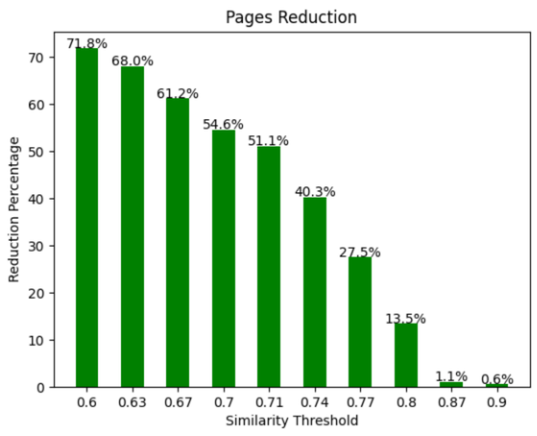

Figure 5. Page reduction results

In (Fig. 4), a clustering example is given, where the graph contains 10 vertices, divided into 3 connected components, therefore 3 clusters. This is done after calculating the similarity score of each pair of web pages of a specific query. As a result, the user will get 3 web pages instead of 10, so from each cluster only one page will be returned because it is similar to all other web pages in the same cluster. Note that single nodes are each considered a separate component. Thus, each represents a cluster with a single web page. This means that there was no other web page that is similar to it with a score of similarity t.

Mathematically, Algorithm 2 shows how the clustering process is applied to each set of webpages.

```
Algorithm 2 Graph Based Clustering
Require: Pairwise Similarity Matrix M (Eq. 2), Similarity
    Threshold t
Ensure: Final Set of Clusters:
    $Clusters = \{(P_1, \dots, P_i), (P_1, \dots, P_j), \dots, (P_1, \dots, P_k$
 1: Initialize: G = Undirected_Graph()
 2: for each similarity score $S_{ij} \in M$ do
 3:   if $S_{ij} \geq t$ then
 4:     Add edge $(i, j)$ to G
 5:   end if
 6: end for
 7: for each connected component $C_i \in G$ do
 8:   Clusters ← $Clusters \cup C_i$
 9: end for
10: return Clusters
```

As shown in (fig. 5), the percentage of reduction evolves when we vary the similarity threshold. Technically speaking, when the threshold decreases, the percentage of reduction increases. Our method is therefore flexible, and it allows us to choose the optimal threshold according to our

objectives. It just depends on how we actually define two webpages as similar. The more we need to detect extremely similar webpages, the higher the value should be, and vice versa.
Additionally, in (Table III), we compare the number of web pages initially returned by Google, with those returned by our method. We also test these results with different similarity thresholds. We can notice how the system can successfully identify similar websites. As described in the Clustering section, similar websites found are placed in a single cluster, and only one of them will be shown to the user.

5. EXPERIMENTS & RESULTS

In this section, we will present the validation metrics with which we have evaluated our work. First, in terms of percentage of reduction of webpages per search query, which is the main goal of this work. Second, in terms of cluster validation.

B. Cluster Validation

In order to ensure a good quality of the results of a clustering algorithm, certain validation procedures are necessary. In general, if the clustering algorithm separates dissimilar observations apart and similar observations together, then it has performed well. Practically speaking, a good clustering

Table III. Search Query Results: Before & After

Language	Query	Before		0.67		0.7		0.8
			After	Reduction	After	Reduction	After	Reduction
English	lasik eye surgery	97	7	92.8	10	89.7	67	30.9
	stomach pain	96	11	88.9	15	84.4	78	18.8
	symptoms of depression	96	13	86.5	15	84.4	57	40.6
	Headache	98	16	83.7	20	79.6	80	18.4
	History of Buddhism in India	89	17	80.9	20	77.5	53	40.4
	first person on the moon	93	18	80.6	19	79.6	56	39.8
	L Internet des objets	97	16	83.5	18	81.5	60	38.1
French	Symptomesˆ du diabete`	97	19	80.4	22	77.3	62	36.1
	qui est Maradona	97	20	79.4	23	76.3	57	41.2
	douleurs dentaires	100	22	78	27	73	94	6
	Causes de maux de teteˆ	98	23	76.5	24	75.5	74	24.5
	Bombe de Hiroshima	99	26	73.7	30	69.7	78	21.2
Arabic	عوارض مرض السكري	100	27	73	44	56	100	0
	موسم جمع الصنوبر	97	27	72.2	35	63.8	75	22.7
	اسباب الصداع	100	35	65	49	51	98	2
	علاج الم الاسنان	100	37	63	49	51	99	1
	الحرب الصينية الامريكية	96	37	61.5	50	47.9	94	2.1

algorithm is achieved when, within a cluster, the data points are most similar to each other, which is known as high intra-cluster similarity, and, between clusters, the points are the most dissimilar, which is the low inter-cluster similarity.

The purpose of clustering algorithm validation is to avoid finding patterns in random data, as well as to help compare two clustering algorithms. Validating clustering algorithms is more difficult than validating a supervised machine-learning system since the clustering process does not use ground truth labels. Moreover, since good scores do not necessarily translate into good efficiency in an application, other evaluation metrics may also be applied, such as processing time or other metrics, which depend on the application itself.

1) External Cluster Validation

Since we do not know the external true data labels, we chose the Agglomerative clustering algorithm as the external validation algorithm. Agglomerative clustering is the bottom-up type of hierarchical clustering [33] [34]. A very widespread technique that is easy to implement. It is also useful since there is no need to pre-specify the number of clusters for example. In addition, it can produce an order of the objects, which can be informative for display when needed. The algorithm begins with individual points as clusters, then merges the closest pair of clusters at each step until only one or k clusters remain.

To measure the distance between every two clusters and decide on the clustering rules, there are several methods, called Linkage Methods. However, since our data entry consists of pairs of web pages, each with the corresponding similarity score between them, it is exactly a pairwise similarity matrix. For this reason, and since there are no data features, we have provided the Agglomerative clustering algorithm with a pairwise distance matrix, taken from the pairwise similarity matrix 2, as data entry.

$$\text{DistanceMatrix} = 1 - \text{SimilarityMatrix} \qquad (3)$$

This way, no linkage method is required, since the distance between every two clusters is pre-computed. In fact, this is a very strong advantage in our case, as it is known that using different distance metrics to measure distances between clusters can produce different results. It is therefore recommended to carry out several experiments and then to compare the result to help the veracity of the real results, which in our case is not necessary.

In addition, the algorithm needs a distance threshold above which the clusters will not be merged. As we have already defined a similarity threshold to be taken into account when grouping data using the graphical approach, we can extract the distance threshold from the same similarity threshold.

$$\text{DistanceThreshold} = 1 \qquad \text{SimilarityThreshold} \qquad (4)$$

Finally, we compared the Agglomerative clustering results with those of the graphical method, using the following metrics:

a. Adjusted Rand Score, Rand Score
b. Adjusted Mutual Info Score, Normalized Mutual Info Score
c. Fowlkes Mallows Score
d. Homogeneity, Completeness, V-Measure Score

For two clustering results, homogeneity checks whether each cluster contains only members of a single class, and completeness checks whether all members of a given class are assigned to the same cluster. Their harmonic mean is called V-measure. All of the metrics mentioned above measure the similarity of two different assignments, ignoring permutations. In other words, they assess the similarity between several clustering results obtained after applying different clustering algorithms. After applying them, we get a value between 0 and 1, where the closer the value is to 1, the more identical the results are.

In our evaluation, these metrics provided a value equal to 1, as shown in (Fig. 6). This means that the Agglomerative clustering algorithm gave the same results obtained by the graph-based clustering method that we implemented, which is an index of stability as well as performance.

However, we still need additional evaluation metrics, to evaluate these results, which we will discuss in the next validation metric.

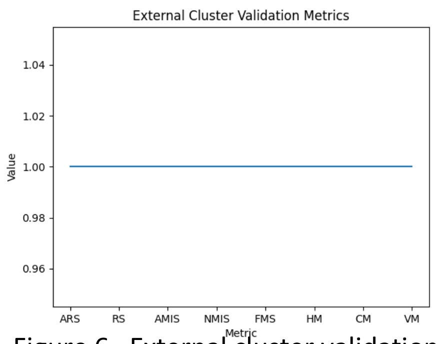

Figure 6. External cluster validation matrix

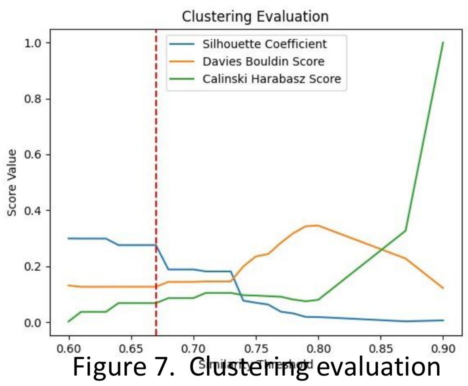

Figure 7. Clustering evaluation

2) Internal Cluster Validation: It uses internal information from the clustering process to assess its quality without reference to external information. Sometimes it is used to estimate the number of clusters or the better clustering algorithm without any external data. During our evaluation process, we applied three different internal validation metrics, with which, in addition to evaluating the clustering results, we were also able to choose the optimal similarity threshold.

i. **The Silhouette Coefficient**: This score is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. A high value indicates that the object is well suited to its own cluster, and poorly matched to neighboring clusters, which means that the higher the value, the better the performance is.

ii. **Calinski Harabasz Score**: Technically speaking, if on the line-plot of the values there appears that one solution gives a peak or at least an abrupt elbow, choose it.

iii. **Davies Bouldin Score:** It is defined as the average similarity measure of each cluster with its most similar cluster. Similarity is the ratio of the distance within clusters to the distance between clusters [37]. Consequently, clusters that are farther separated and less scattered produce better results. The lowest value is zero, with lower scores indicate stronger clustering.

To get the best results, we analyzed the values of the three scores, all together, and we chose the combination of values in which each value alone is considered a good value. As shown in (Fig. 7) the similarity threshold is equal 0.67, the Silhouette coefficient has almost the highest value 0.2575, the Calinski Harabasz at this point shows an abrupt elbow, and the Davies Bouldin Score has almost the lower value 0.1269. In this way, we evaluated our method with good scores of internal cluster validation metrics, as well as we found the optimal threshold of similarity.

6. CONCLUSION & FUTURE WORK

In this work, we introduced REDUCE++, a new model for REsult DUplication detection in Search Engine, which is based on a combination of various similarity measures for each type of content in web pages.

First, we created our own dataset and preprocessed it, both semantically and syntactically. Then, we applied the different similarity measures that are suitable for each type of content of the webpage. This is actually where our contribution lies. Then, we added the different calculated similarity scores to get the final similarity score between the two documents, using is a weighted formula. Finally, we suggested a new clustering algorithm to group search results according to their similarity.

We evaluated our method by comparing it with the Hi-erarchical Agglomerative Clustering, where we got 100% match between them. Furthermore, during the experiments, we collected data from Google, where we included 150 different query, with three languages and four search domains. As results, our model has led to a 61% reduction in web pages, 0.2757 Silhouette coefficient, and 0.1269 Davies Bouldin Score. REDUCE++ is able to detect the similarity between webpages while avoid reducing the quality of information. For our knowledge, this is the first approach that implements this process in order to compute webpages similarity and reduce the duplication.

The main implication of our research for science is redundancy reduction in search engine results. This is a serious problem in search engines that we managed to solve. In terms of practice, end-users can conduct high performance and high-quality search tasks.

Despite the advantages and interesting results of our approach, we still have some drawbacks and limitations in this work. On the one hand, we did not conduct our experiments on several search engines. On the other hand, we have not included the similarity between multimedia contents such as images, pdfs, etc.

As future work, we plan to include several search engines to evaluate and improve our approach, as well as to find the similarity between other web page contents such as images. In addition, we mainly focused on calculating text document similarity, so we will include pdf and docx files as new types of documents to be evaluated as well. Finally, we plan to implement a new weighting method, as it has a significant impact on the similarity score, to improve and obtain better results.

REFERENCES

[1] A. Kritikopoulos, M. Sideri, and I. Varlamis, “Wordrank: A method for ranking web pages based on content similarity,” 08 2007, pp. 92 – 100.

[2] H. Hazimeh, Z. Chreim, A. Noureldine, H. Harb, E. Mugellini, Abou Khaled, and F. Hannoun, “Reduce: a semi-supervised scal-able approach for result duplication detection in search engines.” in Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 25th International Conference KES-2021, 2021, pp. 893–902.

[3] W. Gomaa and A. Fahmy, “A survey of text similarity approaches,” international journal of Computer Applications, vol. 68, 04 2013.

[4] A. Rozeva and S. Zerkova, “Assessing semantic similarity of texts aˆC“ methods and algorithms,” vol. 1910, 12 2017, p. 060012.

[5] T. Gowda and C. Mattmann, “Clustering web pages based on structure and style similarity (application paper),” 07 2016, pp. 175–180.

[6] R. Levering and M. Cutler, “The portrait of a common html web page,” 01 2006, pp. 198–204.

[7] M. Erritali, A. beni hssane, M. Birjali, and Y. Madani, “An approach of semantic similarity measure between documents based on big data,” vol. 6, pp. 2454–2461, 01 2016.

[8] S. Kumar and K. Bhatia, “Semantic similarity and text summarization based novelty detection,” SN Applied Sciences, vol. 2, 03 2020.

[9] A. Islam and D. Inkpen, “Semantic text similarity using corpus-based word similarity and string similarity,” TKDD, vol. 2, 07 2008.

[10] F. Malliaros and K. Skianis, “Graph-based term weighting for text categorization,” 08 2015, pp. 1473–1479.

[11] A. Schenker, H. Bunke, M. Last, and A. Kandel, Clustering of Web Documents Using Graph Representations, 04 2007, vol. 52, pp. 247– 265.

[12] A. Schenker, M. Last, H. Bunke, and A. Kandel, “Classification of web documents using a graph model,” 09 2003, pp. 240– 244 vol.1.

[13] K. Phukon, “A composite graph model for web document and the mcs technique,” International Journal of Multimedia and Ubiquitous Engineering, vol. 7, pp. 45–52, 01 2012.

[14] L. Galdos, C. Lopez, and G. D´avila, “Document classification method based on graphs and concepts of non-rigid 3d models approach,” International Journal of Advanced Computer Science and Applications, vol. 11, 01 2020.

[15] M. Schuhmacher and S. Ponzetto, “Knowledge-based graph document modeling,” 02 2014, pp. 543–552.

[16] A. Lopez-Lopez and A. Gelbukh, “Information retrieval with conceptual graph matching,” 08 2002.

[17] “Selenium: https://www.selenium.dev/documentation/.” 2004.

[18] A. Lahitani, A. Permanasari, and N. A. Setiawan, “Cosine similarity to determine similarity measure: Study case in online essay assessment,” 04 2016, pp. 1–6.

[19] F. Rahutomo, T. Kitasuka, and M. Aritsugi, “Semantic cosine similarity,” 10 2012.

[20] J. Hancock, Jaccard Distance (Jaccard Index, Jaccard Similarity Coef-ficient), 10 2004.

[21] M. Chahal, “Information retrieval using jaccard similarity coefficient,” International Journal of Computer Trends and Technology, vol. 36, pp. 140–143, 06 2016.

[22] G. Lebanon, Y. Mao, and J. Dillon, “The locally weighted bag of words framework for document representation.” Journal of Machine Learning Research, vol. 8, pp. 2405–2441, 10 2007.

[23] Y. Zhang, R. Jin, and Z.-H. Zhou, “Understanding bag-of-words model: A statistical framework,” International Journal of Machine Learning and Cybernetics, vol. 1, pp. 43–52, 12 2010.

[24] G. Sidorov, Vector Space Model, 01 2019, pp. 5–10.

[25] C. Paul, A. Rettinger, A. Mogadala, C. Knoblock, and P. Szekely, “Efficient graph-based document similarity,” 05 2016, pp. 334–349.

[26] K. Riesen, Graph Edit Distance, 01 2015, pp. 29–44.

[27] T. Caetano, J. McAuley, L. Cheng, Q. Le, and A. Smola, “Learning graph matching,” IEEE transactions on pattern analysis and machine intelligence, vol. 31, pp. 1048–58, 07 2009.

[28] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and Vazirgiannis, “Grakel: A graph kernel library in python,” 01 2020.

[29] N. Sherashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borg-wardt, “Efficient graphlet kernels for large graph comparison,” 12th International Conference on Artificial Intelligence and Statistics (AIS-TATS), Society for Artificial Intelligence and Statistics, 488-495 (2009), vol. 5, 01 2009.

[31] A. Subasi, “Chapter 7 - clustering examples,” in Practical Machine Learning for Data Analysis Using Python, A. Subasi, Ed. Academic Press, 2020, pp. 465–511. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128213797000072

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander-plas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duch-esnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[33] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Van-derPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: experiences from the scikit-learn project,” in ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[34] A. Lukasov, “Hierarchical agglomerative clustering procedure,” Pattern Recognition, vol. 11, no. 5, pp. 365–381, 1979. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0031320379900499

[35] K. Chidananda Gowda and G. Krishna, “Agglomerative clustering using the concept of mutual nearest neighbourhood,” Pattern Recognition, vol. 10, no. 2, pp. 105–112, 1978. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0031320378900183

[36] R. Jujjuri and M. Rao, “Evaluation of enhanced subspace clustering va-lidity using silhouette coefficient internal measure,” Journal of Advanced Research in Dynamical and Control Systems, vol. 11, pp. 321–328, 01 2019.

[37] X. Wang and Y. Xu, “An improved index for clustering validation based on silhouette index and calinski-harabasz index,” IOP Conference Series: Materials Science and Engineering, vol. 569, p. 052024, 08 2019.

[38] A. K. Singh, S. Mittal, P. Malhotra, and Y. V. Srivastava, “Clustering evaluation by davies-bouldin index(dbi) in cereal data using k-means,” in 2020 Fourth International Conference on Computing

AUTHORS

Hussein Hazimeh: is a Lecturer of Data Science at the Lebanese University and MIT at Al-Maaref University. In addition, he is a supervisor of a number of PhD students in universities in France, and supervised more than 40 Master students in Lebanon, France, and Switzerland. He holds a PhD in Computer Science from the University of Fribourg, Switzerland, and a Master degree in Computer Science from the Lebanese University and the University of Applied Sciences of Western Switzerland. He worked for 3 years as a scientific collaborator at the school of architecture and engineering of Fribourg (EIA-FR). He published more than 15 research papers. His research interests include: social data analytics, smart education, machine learning, and knowledge graphs.

Hassan Harb: He received his Master's degree in Computer Science and Risks Management from the Lebanese University in 2013. In 2016, He received the PhD degree in Computer Science from the Lebanese University and the University of Franche-Comté (France). He is currently an associate professor at the Lebanese University, American University of Culture and Education (AUCE) and the Antonine University. His research interests are in WSN, IoT, Big Data, machine learning, e-health and modular robotic systems. Dr. Harb has published articles in important journals and conferences, and He is a reviewer in many international journals and conferences.

Karl Daher: is a collaborator at the HumanTech Institute at the University of Applied Sciences and Arts of Western Switzerland. He is co-founder of the research and development entity Empathic Labs. His research activities focus on the creation of empathic interfaces/systems and their use in everyday life. In addition, he is interested in data analysis and evaluation, as well as user experience on these empathic interfaces/systems. He is currently responsible for managing several research and development projects at the HumanTech Institute. Karl Daher holds a Master's degree in Computer Science and Telecom from the Faculty of Engineering at the Lebanese International University and will complete his PhD in Computer Science at the University of Fribourg in 2022.

Elena Mugellini: Telecommunications Engineering and a Ph.D. in Computer Sciences from University of Florence, Italy. Elena is the leader of the HumanTech Institute (Technology for Human well-being). The institute aims at improving the quality of life and well-being of human beings thanks to the ingenious use of new technologies, in order to strengthen their abilities as individuals, as well as members of an increasingly dynamic, nomadic and globalized society. Her research expertise lies in Human Computer Interaction (multimodal and natural interaction, smart spaces, serious game and gamification) and Intelligent Data Analysis (machine learning, multimedia content and knowledge management, semantic technologies). She teaches at graduate and undergraduate level a range of topics, including: project management, human-computer interaction and user experience design, machine learning.

Omar Abou Khaled: is Professor in computer science at HES-SO campus Fribourg (EIA-FR). He holds a PhD in Computer Science received from the Perception and Automatic Control Group of HEUDIASYC Laboratory of « Université de Technologie de Compiègne », and a Master in Computer Science from the same university. Since 1996 he has been working as research assistant in the MEDIA group of the Theoretical Computer Science Laboratory of EPFL (Ecole Polytechnique Fédérale de Lausanne) in the field of Educat