# Collecting, extracting and storing web research survey questionnaires data

Carina F. Dorneles, Gilney N. Mathias

Universidade Federal de Santa Catarina, Brazil
carina.dorneles@ufsc.br, gilney.salvo@gmail.com

**Abstract.** Companies or institutions can use survey questionnaires to evaluate items or products, analyze their employees/customers' satisfaction or collect any data they consider helpful. Furthermore, questionnaires can be used to collect data that can be used in research studies. Some problems in creating such questionnaires involve: deciding which questions to ask, how to ask them, and how to organize them. Many research communities, especially in the healthcare field, maintain repositories that are publicly accessible and include different questionnaires that help professionals and researchers analyze the results of questions, add new questions, or even point out nonsense questions. In this paper, we describe: (i) web crawler, which scans the Web searching for sites that possibly contain questionnaires; (ii) an extractor, which extracts the questionnaires from the list of pages collected by the crawler and saves them into a relational database; and (iii) the public dataset we have created to persist the questionnaires. The database created can then serve to analyze these data and/or as a centralized base of examples to prepare new questionnaires or reuse existing questions. The experiments we have conducted demonstrate that our crawler has achieved 94,47%, and the extractor has achieved a precision between 90% and 92%.

Categories and Subject Descriptors: H.3 [**Information Storage and Retrieval**]: Miscellaneous; I.7 [**Document and Text Processing**]: Miscellaneous

Keywords: HTML research questionnaires, dataset, crawler, data extraction

## 1. INTRODUCTION

The Web has made more accessible communication between people from all over the world and the search for information and knowledge. Such facts opened the doors to the use of online questionnaires, a more comprehensive and easier way for companies and researchers to collect data or profiles of people[1]. The great advantage of using such online questionnaires is the possibility of reaching a large number of people, with common and/or unique characteristics, quickly and cheaply [Wright 2017].

Survey questionnaires are powerful and essential tools for gathering opinions and profiles of people, and they are built for various purposes. With the popularization of the Web, it became common to make these questionnaires available online, so that people's participation is effective. Survey questionnaires are usually designed due to the need to obtain information for which data does not exist – or exists in insufficient quantity. Some problems encountered in the design phase of such questionnaires include: deciding which questions to ask, how best to express them, and how to arrange the questions to get the necessary information. Survey questionnaires can be used for various purposes, such as: in companies to evaluate products or ascertain employee satisfaction, to measure service satisfaction, among others; in teaching and research institutions to assess their faculty and students [da Silva 2012], for example; or by researchers to collect data that are later used in studies and research. Therefore,

[1]https://www.surveymonkey.com/

it can be beneficial to reuse questionnaires, or part of them, already created to carry out new data collections.

Some research communities, such as *Hirsh Health Sciences*[2], *ADAI Library*[3], *RAND Health*[4] and IHSN[5], maintain repositories of research questionnaires that are publicly accessible. They include different questionnaires that help professionals and researchers to analyze the results of questions, add new questions or even point out nonsense questions. These repositories are also useful for researchers who are looking for inventories of validated survey questionnaire instruments. In this sense, it is interesting to have a tool that searches for questionnaires without being a registered user.

Our work aims to collect survey questionnaires from the Web and extract questions to build a centralized database of collected information. The extracted data will serve as a knowledge base that can be used as a starting point for constructing new questionnaires or for the analysis of present characteristics to extract some kind of useful information. To the best of our knowledge, no work focuses on searching and extracting data from survey questionnaires on the Web. However, we have addressed two main problems: detecting questionnaires on any webpage; and extracting the data contained therein, in a generic way, to a database. The work was developed to carry out the experiments reported in [Souza and Dorneles 2019], whose proposal is a similarity metric to compare survey questionnaires and to provide a classification method based on variations in queries constructed by a user in search of questionnaires.

The big challenge in identifying and extracting online questionnaires is the considerable amount of different ways they are constructed using HTML [Laender et al. 2002]. Each website has its way of structuring HTML, styling page elements, and sending data from the client to the server. In the context of survey questionnaires, pages can have dynamic content, where all questions are loaded after the user answers one or more previous questions or clicks a 'next' button, redirecting the user to the next part or page of the questionnaire. Thus, it is essential to emphasize that this work identifies and extracts questionnaires containing static content pages. Figure 1 presents an example of a questionnaire's question, whose answer is must be given by choosing one of the provided alternatives.

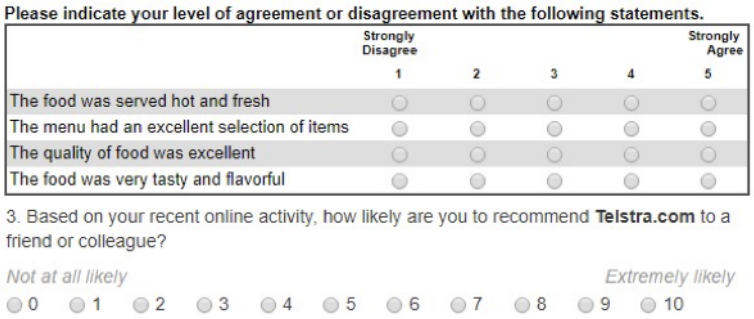


Fig. 1. Example of a question and its answer alternatives from a survey questionnaire

To deal with the lack of standard structure for HTML documents representing questionnaires, we have defined two sets of heuristics: one for questionnaire detection in an HTML page; and another for questions extraction present in these questionnaires. The heuristics defined for question extraction consider common words, clustering of HTML elements that indicate questions, proximity, and distance between nodes of the HTML tree, and characters used at the beginning and end of the questions. We

[2]https://researchguides.library.tufts.edu/c.php?g=249271&p=1659301
[3]http://lib.adai.washington.edu/instruments.htm
[4]https://www.rand.org/health/surveys_tools.html
[5]http://www.ihsn.org/health-modules

also developed a focused *Web Crawler* [Olston and Najork 2010], that scans the Web by downloading pages that have specific characteristics for the user [Liu 2007].

This article is organized as follow. Section 2 present some existing works that propose crawling Web Forms, or maintain repositories of questionnaire. In Section 3, we describe the questionnaires Finder and Extractor, presenting how they work, and the algorithms we have developed for each on. In Section 4, we detail the dataset we have constructed from extracted data and the ground truth designed to evaluate the proposal. Section 5 describes general results of the crawler and extractor. Finally, in Section 6, we describe conclusions and discuss some open questions that can be explored and suggest some directions that can be taken.

## 2. RELATED WORK

In this section, we describe some existing works that (i) propose crawling Web Forms, and (ii) questionnaire repositories, since to the best of our knowledge, there is no work on collecting data from questionnaires.

### 2.1 Web Forms crawling

Due to its similarity in the HTML structure when compared to questionnaires, we present works that make use of web forms in this section. Even though there are many different points when considering a web form and a questionnaire, such as common words like "survey" or "questionnaire", the interrogation character, question words, etc. (we discussed some issues in Section 3.3), we consider the work on web forms important related work.

Most proposals presenting some solution to crawl Web Forms aims at crawling the Deep Web [Kantorski et al. 2015]. Deep Web crawling refers to the problem of traversing the collection of pages in a deep Web site, which are dynamically generated in response to a particular query that is submitted using a search form [Hernández et al. 2019]. Two recent surveys have been published: [Hernández et al. 2019] present a survey where they propose a framework that analyses the main features of existing deep Web crawling-related techniques, including the most recent proposals, and provides an overall picture regarding deep Web crawling, including novel features that to the present day had not been analyzed by previous surveys; and [Madan K. 2021] that reports a survey of RL-based techniques applied in the domain of deep web crawling. The authors in [Madan K. 2021] reviewed the existing literature based on 31 articles from 77 articles published in various reputed journals, conferences, and workshops. As a results they describe challenges related to various crawling steps of deep web crawling are presented.

Recently, some other works have been proposed with solutions for deep web crawling. In [Murugudu and Reddy 2021] the authors propose a novel two-phase deep learning data crawler framework. The first phase initiates in gathering accurate and highly relevant links using the search engine, and the second phase explores fast and in-site relevant website links using adaptive site ranking. The approach focuses on drilling relevant site data and top-k ranking with different relations based on dynamic features with user preferences in single- and multi-query formation with adaptive weight features. The method promises to visualize improved results with efficient data exploration over the traditional approach concerning real-time defense and e-commerce related to web-based services. In [Ismailova et al. 2021] a model structure is proposed that provides navigation through the net as a whole. For logical expressions, a relativized evaluation map is given, which is parameterized by the indexing system, so it is allowed to use concepts that are generated using partial functions, the variables of which run over the range of possible individual information processes. A generalized operator providing virtualization is presented for individuals. The process is accompanied with the dynamic generation of Web-pages, which triggers their indexing. The proposal presented in [J. 2021] is an intelligent and secure solution for autofill. This includes fixing some security vulnerabilities related

to hidden fields in a web form, resulting in a secure auto fill.

### 2.2 Questionnaires repositories

Some research communities, such as *Hirsh Health Sciences*[6], *ADAI Library*[7], *RAND Health*[8] e IHSN[9], maintain repositories of survey questionnaires that are publicly accessible. They include different questionnaires that help professionals and researchers to analyze the results of questions, add new questions or even point out nonsense questions. These repositories are also helpful for researchers who are looking for already validated survey questionnaires. In this sense, it is interesting to have a tool that searches for questionnaires without being a registered user.

Other example of questionnaires repositories is that maintained by Innovations for Poverty Action (IPA)[10], which lists survey instruments that are being developed by IPA and others to learn more about responses to and effects of the COVID-19 pandemic. The goal is to make questionnaires available to a wide audience to allow researchers to harmonize language, item format, response options, and other features in order for make data more comparable across studies. To support this effort, IPA encourages research teams to use the RECOVR survey and the COVID-19 Economic Impact Survey instruments. Another useful questionnaire repository is provided by European Center for Disease Prevention and Control, an agency of the European Union[11]. In their page, the user can find a repository of questions to be used for designing a questionnaire and a guidance to best use this repository. The repository and the guidance are available in all 24 official EU languages.

## 3. QUESTIONNAIRES FINDER AND EXTRACTOR - QFEX

This section introduces how qFEx[12] (*Questionnaires Finder and Extractor*) works. Initially, we present an overview of the work and then we detail the main concepts, the developed components, the algorithms, and the tools used in its development.

### 3.1 Overview

Figure 3 shows the process of collecting and extracting questionnaires, which has two main components: the crawler and the extractor. Both receive a file with the database settings, the level log, the crawler library, the seeds for search, and the parameters' values. In general, the process works traditionally: the crawler component scans the Web, based on the seeds provided in the configuration file, looking for questionnaires that have specific features and saves the links in a file; the extractor component, in turn, uses the crawler's links, and the configuration file, and extracts data from the questionnaires persisting them in relational database. Both components, crawler and extractor, have rules to follow. The crawler rules are implemented through heuristic rules (described later in the Section 3.3), and the extractor rules are implemented through extraction patterns (presented in Section 3.4).

### 3.2 Elements ID and distance calculation

We have used the Dewey numeration [Tatarinov et al. 2002] to provide an unique identifier to each HTML element, giving the node position. Generally speaking, the root node receives the identifier

[6]https://researchguides.library.tufts.edu/c.php?g=249271&p=1659301
[7]http://lib.adai.washington.edu/instruments.htm
[8]https://www.rand.org/health/surveys_tools.html
[9]http://www.ihsn.org/health-modules
[10]https://www.poverty-action.org/recovr/questionnaire-repository
[11]https://www.ecdc.europa.eu/en/publications-data/outbreak-investigation-questionnaire-repository-questions-tool-5a
[12]https://github.com/nogenem/TCC_UFSC

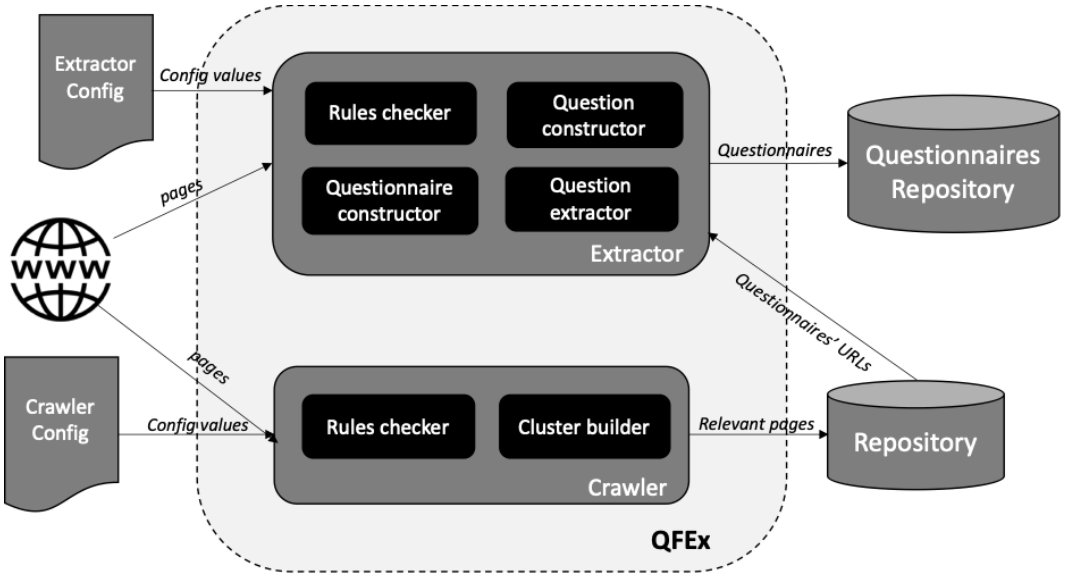


Fig. 2. QFEx general overview

"1," and then each of its child nodes receives an ID composed of the ID of its parent node joined with a number representing the position that this child node appears in the tree. For example: given a root node, the first child node will receive the ID "1.1", the second "1.2", and so on.

We also used this enumeration system to calculate the distance between elements, as proposed in [Leonardo Bres dos Santos 2012], where the Dewey numeration is extended to represent different distances between nodes. Basically, an element ID is composed of four parts, each one containing three digits:

(1) Height: the first three digits indicate the distance of the tree height;
(2) Max Height; the next three digits indicates the max height of a given node until the root;
(3) Width: the next three digits indicates the width, which can be used to verify how depth the nodes are from each other;
(4) Common Prefix: the last three digits are used to indicate if the nodes have the same sequence of relatives, and the value is obtained by sweeping the IDs from left to right and copying their equal values until the first different value is found.

Other changes we have made in relation to the Dewey Numbering: (i) some HTML tags are considered noise and are ignored in the ID definition, such as comment nodes, empty tags, or without the `href` attribute, in addition to `BR`, `DIV`, `SPAN`, `P`, `TH`, and `A` tags.; (ii) text nodes separated by line breaks, `BR`, are joined into a single node. This operation helps in some aspects of the implementation, such as discovering matrix headers where the text has been broken using the *tag* BR for a better visual representation.; and (iii) IDs are represented with padding up to two zeros, for example: ID 1.10.100 is represented as 001.010.100. This representation facilitates the distance calculation. For example, in Figure 3.3(a) the distance between node "Name", ID 0001.0002.0001.0001, and node "input[type=text]", 0001.0002.0002.0001, ID is 001.000.

## 3.3 Questionnaire detection

One of the challenges in collecting and detecting survey questionnaires is their differentiation from WebForms [Leonardo Bres dos Santos 2012], as the structure in the HTML language is very simi-

lar. Therefore, five specific heuristics were defined so that it is possible to identify certain specific components of a questionnaire.

**H1. Common words**. Questionnaires are usually constructed with some common words in their title and/or body. Some words, such as "questionnaire" and "survey" are quite common and can be used as an initial filter to check whether or not a given web page has a survey. This word list can be configurable.

**H2. Clusters of HTML components**. Questionnaires, in general, have a number of HTML forms elements in sequence, we call clusters. Sometimes, the questions are very next to each other, and we can use it to identify a questionnaire. Figure 3.3 presents an example of three clusters.

```
<1>Cluster:
        001.001.002.001.004.001.001 - Q.1
        001.001.002.001.004.002.001.001 - Would you recommend Shopify?
        001.001.002.001.004.002.002.001.001.001 - input[type=radio]
        001.001.002.001.004.002.002.001.001.002.001 - Yes!
        001.001.002.001.004.002.002.001.002.001 - input[type=radio]
        001.001.002.001.004.002.002.001.002.002.001 - No, not currently

<2>Cluster:
        001.001.002.001.005.001.001 - Q.2
        001.001.002.001.005.002.001.001 - Does Shopify work well in your country?
        001.001.002.001.005.002.002.001.001.001 - input[type=radio]
        001.001.002.001.005.002.002.001.001.002.001 - Yes, it works well
        001.001.002.001.005.002.002.001.002.001 - input[type=radio]
        001.001.002.001.005.002.002.001.002.002.001 - No, there are issues

<3>Cluster:
        001.001.002.001.006.001.001 - Q.3
        001.001.002.001.006.002.001.001 - How did you hear about Shopify?
        001.001.002.001.006.002.002.001.001.001 - select
        001.001.002.001.006.002.002.001.001.001.001 - option
        001.001.002.001.006.002.002.001.001.001.002 - option
```

Fig. 3. Example of Question Clusters containing Form Components

**H3. Start/end of question**. A question description usually starts with a number, question words, or a word beginning with a capital letter, having at least four characters, space, and ending with the character ':', '?' or '.'. Figure 4(b) presents an example of questions pattern, such as the numeration (Q1, Q2), some question words, punctuation, and so on.

**H4. Elements' proximity**. Questions usually have their descriptions and form components next to each other, such as the elements of INPUT, TEXTAREA, SELECT, and the like. Figure 3.3(a) presents an example, where we can see that the distances between the questions' descriptions and their form components are 1, for both fields (Name and E-mail).

**H5. Exclusion by distance**. It is possible to eliminate specific forms/fields that do not belong to a questionnaire, checking the distance between the components and certain words/phrases usually found in them. This heuristic is mainly used to eliminate loose components on web pages, such as search fields or even elements for login, registration, and similar forms. Figure 3.3(c) shows some of the words/phrases that are commonly found in this login forms, such as "Keep me logged in", "Don't have an account?", "Login", and so on.

### 3.4 Extraction patterns

Survey questionnaires, built with HTML, can be structured using the most diverse tags. In this work, we performed a previous study and delimited some HTML tags, which were the most commonly found. The tags impose a certain standard in the definition of the questions used by the extractor to extract the data corresponding to the questions, answer alternatives, and children questions.

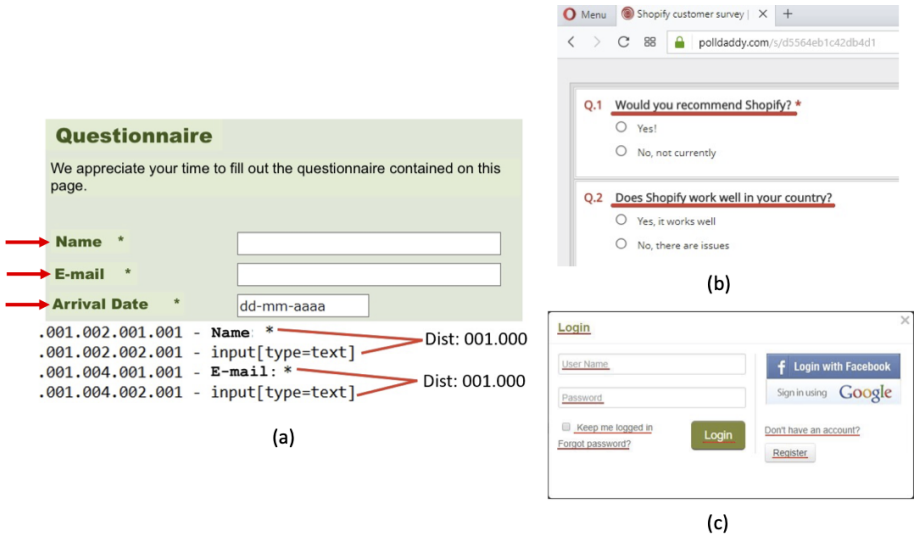


Fig. 4. (a) Example of the distance between question descriptions and their components; (b) Examples of questions patterns; (c) Common words in login forms

—**Identification data**: they correspond to commonly used attributes, mainly at the beginning of questionnaires, mostly used to get users' identification information or registration. Generally, they are in tags such as `TEXT, NUMBER, EMAIL, DATE, TEL, TIME` and `URL INPUTs` ;

—**Free answers**: it is a commonly used pattern for open questions, which is represented using the tag `TEXTAREA`;

—**Select answers**: it is commonly used for questions having alternatives, which is structured using the tag `SELECT` that can have many `option` tags;

—**Alternative answer**: they indicate another pattern of questions with alternatives of answers, which can be text, images, or even matrix, structured using the tags `RADIO` and `CHECKBOX INPUTs`;

—**Rating answers**: it is an answer where user must provide a rate between 0 and n;

—**Question having many components**: they represent questions with several components in sequence, such as questions that ask for more than one answer from the user.

—**Matrix question**: they are questions structured as table, where line is a question and each column corresponds to the answer.

—**Question having subquestions**: it represents questions with another question as child, which can have any structure presented before.

### 3.5 Algorithms

In this section, we present four algorithms we have created to find questionnaires on the Web and extract data from them.

Algorithms 1 and 2 are used to detect questionnaires in an Web page. The basic idea of Algorithm 1 is to determine whether or not the input URL should be saved in the database. To do this, it checks the title and body of the page and groups the text, image and form components nodes and calls Algorithm 2 to check the presence or not of a questionnaire on the HTML page. Algorithm 1's initial check looks for specific words/phrases common in online questionnaires, such as: questionnaire, survey, questionnaire, etc (see Heuristic H1 - Section 3.3). Node groupings are initially done by proximity and then using specific heuristics, such as joining input clusters in sequence and text clusters followed by component clusters.

**Algorithm 1:** Focused crawler - verifying HTML page

```
 1: boolean shouldSave(html: HTML)
 2:    root <- get body of html;
 3:    if ((not text of page has certain words) AND
 4:       (not title of page has certain words)) then
 5:       return false;
 6:    end if
 7:    nodes <- find nodes of root;
 8:    clusters <- group nearby nodes;
 9:    clusters <- group clusters by heuristics;
10:    if hasQuestionnaire( clusters ) then
11:       return true;
12:    else
13:       return false;
14:    end if
15: end shouldSave
```

To determine if a page has a questionnaire, Algorithm 2 runs through the clusters passed to it, checking the number of form components each has and counting the number of clusters with at least one component. If the number of components in a cluster is greater than or equal to a particular value or if a certain amount of clusters with at least one component is found, then the algorithm returns true, indicating the possible presence of a questionnaire on this page. Line 5 implements Heuristic H2 (Section 3.3).

Verifying the number of components in a cluster is necessary since some questionnaires are made so simple that the entire structure is grouped by proximity in the same cluster. Line 6 checks if the questionnaire being analyzed has reached the end. Such verification considers the distance between the clusters and the number of clusters that have only text and/or images between clusters that have form components (Lines 9 to 17 in Algorithm 2). It implements Heuristic H2, Heuristic H3, and Heuristic H4, described in Section 3.3.

**Algorithm 2:** Focused crawler - verifying clusters

```
 1: boolean hasQuestionnaire(clusters: List)
 2:    cCout <- 0.0; # componentCout
 3:    qCount <- 0; # questionCount
 4:    for each cluster in clusters do
 5:       cCout <- count the number of components in cluster;
 6:       if (is starting a new questionnaire) then
 7:          qCount = 0;
 8:       end if
 9:       if (cCout >= 1.0) then
10:          if (cCout >= MIN_COMPS_IN_ONE_CLUSTER) then
11:             return true;
12:          else
13:             qCount = qCount + 1;
14:          end if
15:          if (qCout == MIN_CLUSTERS_WITH_COMP) then
16:              return true;
17:          end if
18:       end if
19:    end for
20:    return false;
21: end hasQuestionnaire
```

Algorithms 3 and 4 are used to detect extract questions, answers and children questions. The basic idea of Algorithm 3 is to group text and image nodes that are close to each other until a component node is found where, at this point, the algorithm tries to extract the question referring to this component. Line 15 takes into account the distance between the clusters and the number of text/image clusters between clusters with components. Lines 16 and 28 are used to eliminate the forms and fields dropped by the page (see Heuristic H5, described in Section 3.3). It checks that the questionnaire has a minimum number of questions and analyzes its subject and description of

questions.

**Algorithm 3:** Questionnaire extractor

```
List buildQuestionnaires(html : HTML)
    root <- get body of html;
    nodes <- find nodes of root;
    stack <- create new Stack;
    cluster <- create new Cluster;
    questionnaire <- create new Questionnaire;
    questionnaires <- create new List of questionnaires;

    for each node in nodes do
       if (node is a text OR node is an image) then
          if (should create a new cluster) then
             stack.add( cluster );
             cluster <- create new cluster;
          end if
          if (is starting a new questionnaire) then
             if (is a valid questionnaire) then
                questionnaires <- questionnaire;
             end if
             questionnaire <- create new questionnaire;
          end if
          cluster.add( node );
       else if (node is a component) then
          stack.add( cluster );
          questionnaire.addQuestion( buildQuestion( ... ) );
          cluster <- create new cluster;
       end if
    end for
    if (is a valid questionnaire) then
       questionnaires <- questionnaire;
    end if
    return questionnaires;
end buildQuestionnaires
```

Algorithm 4 extracts the questions from the questionnaire and, in addition, it also finds other essential elements, such as the subject of the questionnaire, images related to it or the current question being extracted, the descriptions of the question groups.

**Algorithm 4:** Questions extractor

```
Question buildQuestion(questionnaire : Questionnaire, nodes :
List, stack : Stack)
    question <- create new Question;
    if (have components in sequence) then
       do the extraction depending on the form of the question;
    else
       do the extraction depending on the type of component;
    end if
    if (question was extracted) then
       update the description of the question;
       check if it is a matrix;
       check if it is a question with subquestions;
       check if the question has an associated image;
       if (questionnaire doesn't have a subject) then
          check if last cluster is the description of a group;
          check if questionnaire has an associated image;
          find the subject of the questionnaire;
       else
          check if last cluster is the description of a group;
       end if
    end if
    return question;
end buildQuestion
```

### 3.6 Prototype implementation

qFEx was developed in JAVA from Oracle[13] together with the following external libraries: Crawler4j[14], which has all the functions a *crawler* should implement; Jsoup, to handle the HTML DOM tree; and Json, to handle Json format files. The work has three subprojects (*Common Lib*, *Crawler* and *Extractor*) in order to reduce code duplication.

## 4. DATASET AND GROUND TRUTH

Currently, the data collected comes from 2,262 questionnaire links found on 32 different websites[15], distributed both in Portuguese and in English. The questionnaires are classified into eight different research domains, which are presented in Table I.

From the 32 websites collected, manual analysis was performed to more precisely identify those that had questionnaires. We have decided to use a maximum of 25 questionnaires from the same website to homogenize the database, as some sites had hundreds of questionnaires while others only had one or two. This logic has been used to: (i) not create rules that would work better in portals with more questionnaires; (ii) and, identify a greater number of distinct structures. In addition, some questionnaires were considered useless because they represented web forms, questionnaires in different languages (many of them illegible), or even questionnaires with incomplete questions in a draft version. Thus, in the end, the experiments were carried out on a total of 510 questionnaires.

The collected questionnaires have 5,765 questions in total, with an average of 11 questions per questionnaire, with minor questionnaires having two questions and the most extensive one having 53 questions. Of this total of questions, 4,161 are closed questions, that is, questions that have fixed alternatives where the user has to choose exactly one of them; 1,351 are open questions, in which users are free to inform the answers they wish; and 254 are multiple-choice questions, where users can choose between one or more from the present alternatives.

The ground truth used for crawler and extractor evaluation was manually set. For the crawler, we checked whether the 32 websites actually contained questionnaires on the retrieved links. For the extractor, we have checked if questions, alternatives, and child questions have been correctly extracted.

Table I. Application's domains of the collected questionnaires

| ***Domain*** | ***nr. of Questionnaires*** | ***Total %*** |
|---|---|---|
| Products/Services Evaluation/Satisfaction, etc | 214 | 41,96% |
| Other subjects | 88 | 17,25% |
| Market, Business and Marketing Survey | 76 | 14,90% |
| Human Resources and Business Environment | 47 | 9,26% |
| Education and training | 30 | 5,88% |
| Health and Sports | 22 | 4,31% |
| Entertainment and Events | 22 | 4,31% |
| Comunidade e ONGs | 11 | 2,16% |

### 4.1 Database model

The extracted data was initially structured in JSON, and after stored in a relational database. The defined logical structure was designed so that the questionnaires could be reconstructed according to the originals. Figure 4.1 presents the database model where the extracted data is persisted - all entities

[13]https://www.oracle.com/java/index.html
[14]https://github.com/yasserg/crawler4j
[15]https://github.com/nogenem/TCC_UFSC/blob/master/tcc_forms_v03_all.backup

are tables in the database. Table `Questionnaire` keeps the link to the questionnaire as a way to keep its origin. Some important points should be considered: (i) table `Category` holds all the different ways to ask a question in a questionnaire, such as: `checkbox input`, `radio input matrix`, `text input group` and others; (ii) table `Figure` stores the information of images found in the questionnaire; such images can belong to the questionnaire, to a question or an alternative, fields `Owner` and `ownerID` are used to store it physically in the database; (iii) a question can have one or more child questions. This is used for cases like matrices and questions with sub-questions; and (iv) the type of a question can be `open`, `closed` or `multiple choice`, depending on its category. Finally, table `Group` stores data such as "Organization and Institutional Management", "Infrastructure", and so on. Since we extract data from Web and store in a database, the conceptual model elaborated has some restrictive cardinalities like it does not include identification of data sharing between different questionnaires or in the same questionnaire.

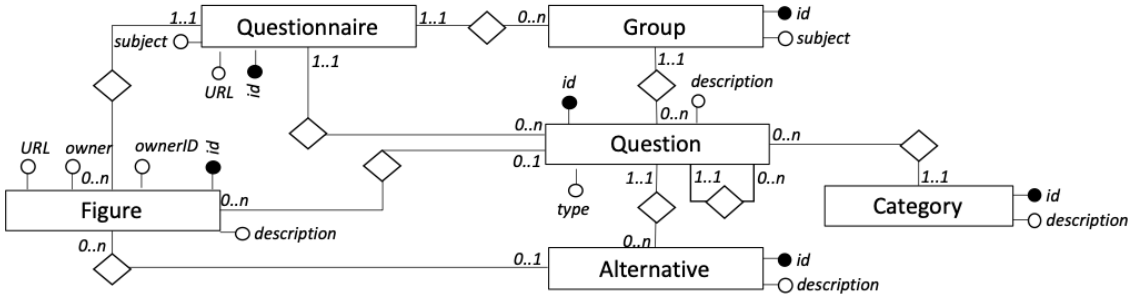


Fig. 5. Database model

## 5. EXPERIMENTS AND RESULTS

This section presents the recall, precision and f-value results we reached with the experiments performed with the crawler and the extractor. The objective of these experiments is to verify if the crawler correctly identified the HTML pages that had research questionnaires and if the extractor correctly got the questions, alternatives of each question, and child questions.

### 5.1 General Results

The crawler precision reached 94.47%, which is, 2,137 of the links found actually had one or more questionnaires in them. The remaining links ( 5.53%) did not have questionnaires, but extensive forms or multiple small forms, but close to each other, which confused the crawler questionnaire detection algorithm. We did not perform recall evaluation in this case since the harvesting was performed throughout the Web.

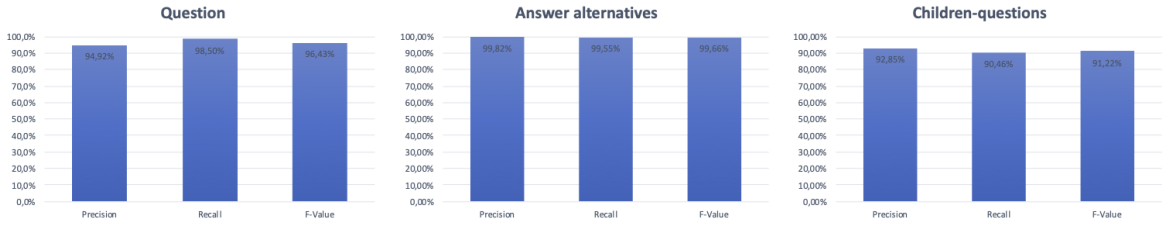


Fig. 6. Precision, Recall and F-1 (a) Question; (b) Answer; (c) Children-answers

Figure 6(a) contains the overall averages of the metrics used to assess question extraction. Results surpass 90% in all cases, which indicates that most of the question descriptions were extracted correctly. Figure 6(b) presents the general averages of the extraction of alternatives. In this case, notice that the results obtained were close to 100 %, indicating that the approach used to extract the alternatives was effective.

Figure 6(c) shows the results for the extraction of children questions, where it is possible to see that the precision reached a little more than 92%, the recall to 90%, and the F-1 to 91%. There are two main reasons for these reduced results compared to question, and alternative extraction: (i) some websites place child questions at the same level as a common question; and (ii) in some cases, the questionnaire's subject is considered the parent question of the first question in the questionnaire. It happens because the structure at the beginning of the questionnaire is identical to a question with children questions.

## 6. CONCLUSION AND FUTURE WORK

The main problems addressed in this work were the discovery and extraction of survey questionnaires found on the Web. To effectuate the process, a focused Web crawler capable of scanning the Web in search of online questionnaires and an extractor that can extract data from the questionnaires were developed. All extracted data were structured and stored in a relational database. As this is a problem not explored by the scientific community, many points remain open, enabling the development of various research options. Below, we discuss some open questions that can be explored and suggest some directions that can be taken:

**Metadata extraction:** in this version of our work, we focused on extracting attributes questionnaire's inherent. Adding new attributes referring to metadata can be considered (such as authorship, creation date, etc.). However, it would require a new version of the crawler to insert into the code the logic to detect such information in pages HTML. Such information is undoubtedly arranged in different regions of the questionnaires or questionnaire portals, and it is necessary to create new extraction algorithms and a new version of the extractor, which should foresee different possible structures to be considered.

**Dynamic websites**: scrapping dynamic websites is not a trivial task, since the data on such websites changes depending on an user answers or data input, and the task is reduced to frequent scraping of the static website. This task requires the crawler to plan the extraction based on the different response alternatives the user can provide, and a machine learning technique can be designed in the sense of learning the possible answers.

**Identification of identical questions in different questionnaires**: we analyzed the existence of similar questions in the different questionnaires. However, the issue is more complex than it might seem as semantically similar questions may have different syntactic structures and this requires further analysis using PLN techniques or word embeddings. The work developed in [Souza and Dorneles 2019] proposes an approach to search for similarity to questionnaires, which can be adapted to identify questionnaires that have questions that, semantically, may represent the same thing.

**Natural Language Processing Heuristics**: we have created a set of basic heuristics to find and extract questionnaires. However, it could be interesting to test some natural language processing for discovery questions in a more precise way [Zheng et al. 2017].

## REFERENCES

da Silva, J. M. *Collecta: um sistema computacional de coleta de dados e avaliação institucional para apoio à tomada de decisão na Universidade Federal de Santa Catarina.* M.S. thesis, Universidade Federal de Santa Catarina, Florianópolis, 2012.

Hernández, I., Rivero, C. R., and Ruiz, D. Deep web crawling: A survey. *World Wide Web* 22 (4): 1577–1610, July, 2019.

Ismailova, L., Wolfengagen, V., and Kosikov, S. A semantic model for indexing in the hidden web. *Procedia Computer Science* vol. 190, pp. 324–331, 2021. 2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society.

J., B. Intelligent and secure autofill system in web browsers. In *Proceedings of the 12th International Conference on Soft Computing and Pattern Recognition*, 2021.

Kantorski, G. Z., Moreira, V. P., and Heuser, C. A. Automatic filling of hidden web forms: A survey. *SIGMOD Rec.* 44 (1): 24–35, may, 2015.

Laender, A. H., Ribeiro-Neto, B., da Silva, A., and Teixeira, J. A Brief Survey of Web Data Extraction Tools. *Sigmod Record* 31 (2), 06, 2002.

Leonardo Bres dos Santos, Carina F. Dorneles, R. S. M. An approach for extracting web form labels based on distance analysis of html components. In *Proceedings IADIS International Conference WWW-Internet 2012*, 2012.

Liu, B. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data.* Data-Centric Systems and Applications. Springer, 2007.

Madan K., B. R. Reinforcement learning in deep web crawling: Survey. In *Proceedings of Second Doctoral Symposium on Computational Intelligence.*, 2021.

Murugudu, M. R. and Reddy, L. S. S. Efficiently harvesting deep web interfaces based on adaptive learning using two-phase data crawler framework. *Soft Computing*, 2021.

Olston, C. and Najork, M. Web Crawling. *Foundations and Trends in Information Retrieval* 4 (3): 175–246, 2010.

Souza, R. H. and Dorneles, C. F. Searching and ranking questionnaires: An approach to calculate similarity between questionnaires. In *Proceedings of the ACM Symposium on Document Engineering 2019*. DocEng '19. Association for Computing Machinery, New York, NY, USA, 2019.

Tatarinov, I., Viglas, S. D., Beyer, K., Shanmugasundaram, J., Shekita, E., and Zhang, C. Storing and querying ordered xml using a relational database system. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*. SIGMOD '02. Association for Computing Machinery, New York, NY, USA, pp. 204–215, 2002.

Wright, K. B. Researching Internet-Based Populations: Advantages and Disadvantages of Online Survey Research, Online Questionnaire Authoring Software Packages, and Web Survey Services. *Journal of Computer-Mediated Communication* 10 (3), 07, 2017. JCMC1034.

Zheng, W., Cheng, H., Zou, L., Yu, J. X., and Zhao, K. Natural language question/answering: Let users talk with the knowledge graph. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. Association for Computing Machinery, New York, NY, USA, pp. 217–226, 2017.