Distr.
GENERAL

WP.18
8 May 2012

ENGLISH ONLY

UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE (UNECE)
CONFERENCE OF EUROPEAN STATISTICIANS

EUROPEAN COMMISSION
STATISTICAL OFFICE OF THE EUROPEAN UNION (EUROSTAT)

ORGANISATION FOR ECONOMIC COOPERATION AND DEVELOPMENT (OECD)
STATISTICS DIRECTORATE

Meeting on the Management of Statistical Information Systems (MSIS 2012)
(Washington, DC, 21-23 May 2012)

Topic (iii): Innovation

The Web-based Data Collection in the Italian Population and Housing Census

Supporting Paper

Prepared by Leonardo Tininini and Antonino Virgillito, ISTAT, Italy

I. Introduction

1. One of the most innovative feature of the 15th Italian Population and Housing Census was an integrated web information system, supporting all the activities of the collection process.

2. The three main components of this integrated system are: (i) the Census management system, to be used by enumerators, operators, coordinators, and Istat personnel to manage the various activities, e.g. household assignment to enumerators, monitoring of the collection activity, overview of the main data summarizing the progress of the collection process, etc.; (ii) the online documentation systems available for operators; and (iii) the web data collection system (online questionnaire), mainly used by citizens for providing their Census data and also, for a few specific towns, by operators to perform the data entry of (paper) questionnaires collected by enumerators.

3. In this paper we focus on the main features of the web data collection system and the innovative solutions, on which it is based. Particularly, we illustrate the strong integration with the Census management system, the support for multilingual compilation, the metadata-driven methodology underlying it, the solutions developed to improve the quality of data inserted, while still ensuring good response times, even with large number of concurrent accesses.

4. Finally, we also illustrate a sophisticated mechanism developed for the assisted coding of textual responses, based on a dictionary, similarity string comparison and automatic ranking. The technique is used in the Census online questionnaire to support the web respondents to code the question on highest educational qualification, but can be generalized to any question requiring the automated coding of a textual response.

II. The Census Web-based Information System

5. The web system supporting the 15th Italian Population and Housing Census is composed by three applications integrated into a unique platform: the Census management system (SGR), the online questionnaire (QPOP), and the online documentation for operators (RETE).

6. SGR is the core of this integrated platform and is accessed by both Census office operators and Istat personnel. Its functionalities encompass the various phases of data collection and related monitoring. In particular, it allows Census office operators to manage the entire process, e.g. by assigning households to enumerators, monitoring the various collection activities and also to check the progress of activities by the visualization of some key indicators. Since the data collection process is multichannel, SGR enables the monitoring of questionnaires collected in the various possible ways (either online or from enumerators or by delivery to postal/municipality collection centers). SGR is also used by statisticians to collect the main variables to be disseminated in the Census provisional results. Finally, one further fundamental SGR module allows local administration offices to compare and re-align the data on citizens collected by the Census with those stored in Local Population Registries.

7. QPOP enables Italian citizens to return their questionnaire online. The interface assists users in following the correct compilation rules and checking errors before the final submission of the questionnaire. QPOP can also be used by Census office operators for performing data entry operations. When a questionnaire is completed in QPOP, all information provided are immediately available in SGR for further processing and checks. The online data collection has resulted in significant reduction of the on-field activity, with obvious economical and organizational advantages. Furthermore, the checks performed on data inserted by users before being saved greatly improved the quality of collected data.

8. The whole system architecture was entirely designed and developed by Istat. This was mainly motivated by the required tight integration and co-operation between the online questionnaire and the Census management system, a functional co-opeartion that would be practically impossible to implement, by reusing and extending the functionalities of existing tools for survey management and questionnaire compilation. The tight integration between the two applications allowed for real-time monitoring of the whole collection process, greatly reducing the gap between data collection and further related processing, in particular the fundamental process of re-alignment between Census and Local Population Registries.

9. The integrated system allowed Istat and the Population Census Network to cooperate more effectively. In this context, around 100,000 operators were involved in the Population Census operations and more than 33% of citizens returned their questionnaire through QPOP, showing that this channel was particularly appreciated.

III. The Online Questionnaire

10. The online Questionnaire of the Italian Population Census (**QPOP**) is a web application made available to citizens that could use it to fill up their census questionnaire in a way which was perfectly equivalent to fill up the printed form. An authorization code was printed in the front page of the paper questionnaire that, together with the respondent's fiscal code, could be used as an authentication credential for accessing the web application. The application reproduces all the three types of paper questionnaires, namely the two questionnaires for households (long and short form) and the questionnaire for collective dwellings.

11. At the end of the data collection phase around 8,500,000 questionnaires were returned through the QPOP application, corresponding to more than 33% of the total number of expected questionnaires. The average load in the first two months of Census operations was 100 questionnaires completed per minute, peaking to 300 in periods of maximum load.

12. QPOP is seamlessly integrated with the SGR Census management system: once a respondent completes a questionnaire in QPOP, the state of the questionnaire is immediately updated in the central database and

visible to operators through SGR. Operators could also use QPOP for performing data entry operations, thus updating the questionnaire "state" in SGR, accordingly.

13. Since the QPOP application was potentially available to the whole Italian population, it was of primary importance to carefully design it so that it could scale gracefully to a huge number of users, at the same time achieving a high robustness (i.e., no application errors presented to the user) and preserving data security despite all the possible security threats. Sophisticated technical solutions were adopted in the internal design of the application for (a) reducing the load to the database even in presence of a high number of users connecting to the application and (b) performing data validation at multiple points in the application for guaranteeing consistency of data sent by the user.

14. Furthermore, the design largely exploits metadata, avoiding the development of redundant source code for the three questionnaires. Everything that appears in the web user interface (including warning and error messages) is stored in specific metadata tables and resource files. In particular, the text of each single question constituting the questionnaire is stored in a (meta-)database table with the 3 different localized (Italian, German and Slovenian) versions. Also the single possible response modalities are stored in a (meta-)database table with the 3 different localized versions. Changes and corrections to the texts constituting the questionnaires could be (and indeed were) made at any moment, even after the final deployment of the application and its release to citizens, without affecting the source code of the application.

15. Texts are not the only part of the QPOP application, which are stored as metadata. A further fundamental feature of QPOP is that also part of the "behavior" of the application was stored in metadata. This is closely related to the concept of "questionnaire graph", which is a fundamental part of the technique, used to formally model the structure of the questionnaire and the correct set and sequence of questions to be filled in by the respondent. The questionnaire graph and its usage will be illustrated in more detail in Section IV.

IV. Application Design of the Online Questionnaire

16. QPOP is a web application implemented through Java2 Enterprise Edition platform and organized in a 5-layer architecture following the Model-View-Controller design pattern. The implementation exploits three widespread open source frameworks, namely Struts2, Spring and Hibernate. The layered application design is depicted in Figure 1.

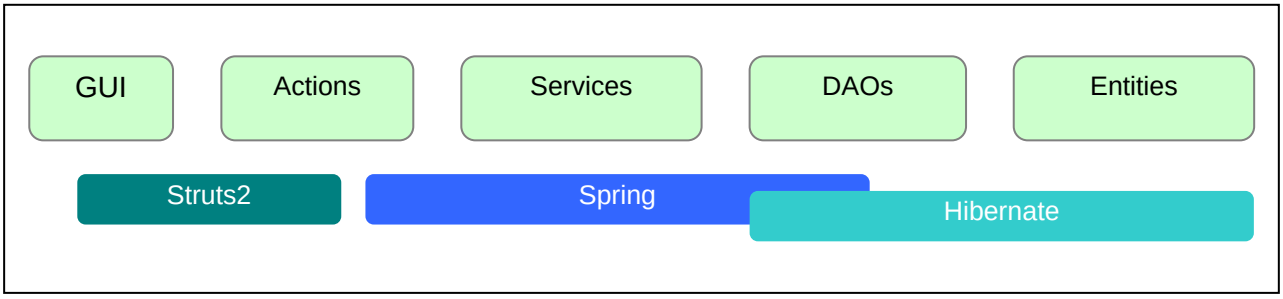

Figure 1: the 5-layer architecture of QPOP

17. The functionality of each layer is detailed in the following:

- **GUI**: JSP pages implementing the graphical user interface. They can be forms for sending data to the server, processed by an action, and/or results of an action execution;
- **Actions**: Java classes whose execution is triggered by a HTTP call, activated by a form submission on the GUI. They receive data from the HTTP request and execute some server-side processing by calling Services;
- **Services**: Java classes that implement database transactions, realized through sequences of calls to DAOs;

- **Data Access Objects** (DAOs): Java classes that implement so-called CRUD (Create-Read-Update-Delete) database operations related to one or more domain objects;
- **Entities**: Java classes representing records of one database table.

18. The implementation of each layer is supported by one or more Java *frameworks*. Frameworks are used to simplify development and to obtain more robust and standardized code. These are the frameworks used in the QPOP application and their role within the architecture:

- **Hibernate**: maps domain objects to database tables. Provides an API (used within the DAO objects) to access to CRUD operations that creates the SQL statements corresponding to each operation by exploiting the object-table mapping;
- **Spring**: handles database access configuration and transparently supports transactions. Service classes are managed by Spring, so that each method in a service class is surrounded by a transaction, which is automatically committed, if the method is successfully completed, or rollbacked, if some errors occur.
- **Struts2**: is a Model-View-Controller framework that handles the communication between the JSP pages that represent the application GUI and the server-side actions that handle the processing.

19. Upon application startup all the metadata for the three questionnaires, including the questions and the questionnaire graph, is loaded into main memory. The dictionaries used in some questions (list of provinces and municipalities, foreign countries and highest educational qualification) are also loaded. All this cached metadata is used when presenting pages to users and this largely limits the access to the database during user interaction, a fundamental design principle to achieve high scalability.

20. When the system shows a questionnaire page, it first reads all the metadata related to the questions in that page. Metadata for a question includes all the (localized) texts and the question types. There are more than 20 different question types in Census questionnaire and each question type is associated with a different template. A template specifies how a question type should be rendered in terms of HTML and is associated with specific Java *beans* to implement the corresponding data processing. By using templates together with metadata we achieved code optimization through massive template reuse, and faster response to changes in the specifications that occurred along the project.

21. When users saves data on a page, quality checks are applied. The idea is to allow saving data in the database only if it is satisfies all the quality constraints. The save operation largely exploits the features of the Struts2 and Hibernate frameworks, together with Java Reflection. Form data sent through HTTP is saved automatically by Struts2 in a server-side Java object (bean). Then, the Hibernate framework is responsible for mapping the bean to a database table, automatically writing the SQL queries for inserting/updating data. The Spring framework is used to manage the code needed for transaction management. The heavy use of frameworks had a significantly positive impact, producing cleaner code that was easy to write, test and maintain, resulting in a more robust application.

22. Before actually saving the data all the quality checks are repeated on the server. This step is fundamental for security reasons, because a malicious user could easily bypass the checks on the page by altering the data sent through the HTTP request. This could lead either to inconsistent data being sent to the server or even to security threats (e.g. injection of malicious SQL code that could reveal sensitive data from the database).

V. The Questionnaire Graph

23. As already mentioned above, an important feature of QPOP is that also part of the “behavior” of the application was stored as metadata. This is closely related to the concept of “questionnaire graph”, which is a fundamental part of the technique, used to formally model the structure of the questionnaire and the correct set and sequence of questions to be filled in by respondents. The idea of formally describing the information about question routing by using (directed) graphs is obviously not new and has been adopted by several other systems. However, as there are several possible variations to the basic idea, we will describe the specific technique used

in our project and how the formal description of the questionnaire graph has been exploited very efficiently in the QPOP application.

24. A Questionnaire Graph (QG) in QPOP is a Directed Acyclic Graph (DAG), such that:
   a) Nodes are in 1-1 correspondence with each questionnaire fragment (mainly questions, but also sections, instructions, as well as other modeling components that do not have a visible counterpart in the user interface);
   b) A (directed) labeled edge from node (question) N_i to node (question) N_j corresponds to the fact that the user has to respond to question N_j after having given a response to node N_i. if the condition expressed on the edge label is true. In general there may be more than one edge, each with a different label, exiting from the same node N_i
   c) The edge labels represent question routing (e.g. “If you have answered Yes to question X, then go to question Y, otherwise proceed”). The conditions on the edges exiting from any given node N_i have to be mutually exclusive (and one needs to be true) to univocally determine which is the actual “next” question after N_i.

25. In order to simplify their evaluation and usage on both client and server side, conditions on the QG edges have a fairly simple syntax constituted by a tuple of values. More complex conditions can be obtained by combining (in AND and OR) these elementary conditions. The most elementary condition is constituted by a pair (X,n), where X is a (closed) question ID and n an integer value. This is interpreted as follows “Is the answer (item number) given to question X equal to n?”.

26. If the condition has the form (X,n1,n2) the meaning is “Is the answer given to question X a value between n1 and n2?”. Depending on the type of question X, this may correspond to a range of values (integers, dates, etc.) or, in the case of closed questions, to a subset of the (finite) possible choices associated with it.

27. Most complex conditions enable the questionnaire modeler to express conditions like “Are there at least(/at most) n checked cells in the rectangular interval (4, 6) on rows and (2, 5) on columns?”. In practice this enables the modeler to express existential and universal quantification on the options checked by the user and to use the corresponding conditions to route the question compilation. Figure 2 shows an excerpt of the Census questionnaire and the corresponding fragment of QG.

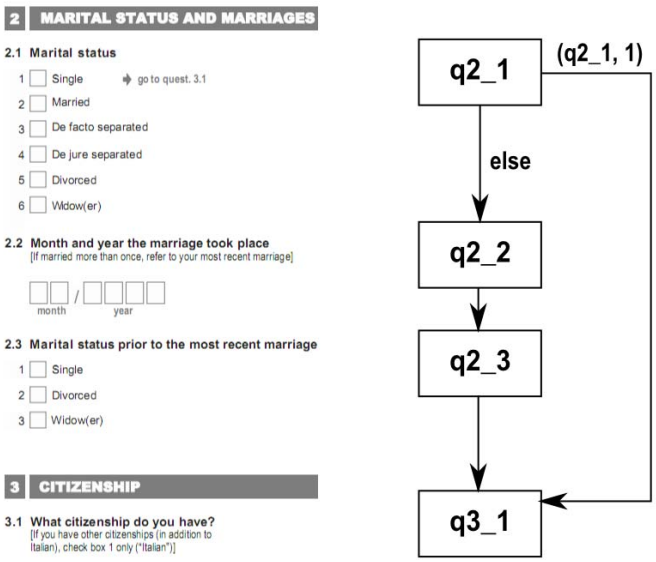

Figure 2: an excerpt of the Census questionnaire and the corresponding QG fragment

28. The collection of all data about nodes (questionnaire components, in particular questions) and edges (transitions from one question to another, with related labels/conditions) is stored in the QPOP (meta-)database and used by the application (both on client and server side) to visually enable and disable questions on the web

page and to validate the user's input before saving the user's answers in the microdata tables. The algorithm used on both client and server side is basically the same and corresponds to compute the "state" of each node in the GQ, depending on the structure of the graph and on the answers already given by the respondent.

29. During the compilation each node can be in one of the following states:
   a) already filled in by the user;
   b) unreachable by the compilation, according to the structure of the GQ and the answers given so far by the respondent;
   c) potentially (but not necessarily) reachable by the compilation, according to the structure of the GQ and the answers given so far by the respondent;
   d) necessarily to be reached by the compilation, independently of the answers still to be given.

30. The basic assumption underlying the client-side behavior of QPOP is that only questions in state (a) or (d) have to be made enabled by the user interface. Similarly, on the server side, only the data corresponding to questions in state (a) or (d) has to be permanently stored in the microdata DB tables. Figure 3 shows the states of the 4 nodes (questions) of Figure 2: (A) at the beginning of the compilation; (B) after the first item has been selected as the answer to question 2.1; or (C) after the third item has been selected as the answer to question 2.1. Note that in both cases (A) and (B) the questions 2.2 and 2.3 will be disabled in the user interface. In case (B) these questions are incompatible with the answer already given, while in case (A) they *may* produce incompatibilities, if an answer is given to either question 2.2 or 2.3 and the answer to question 2.1 is 1.

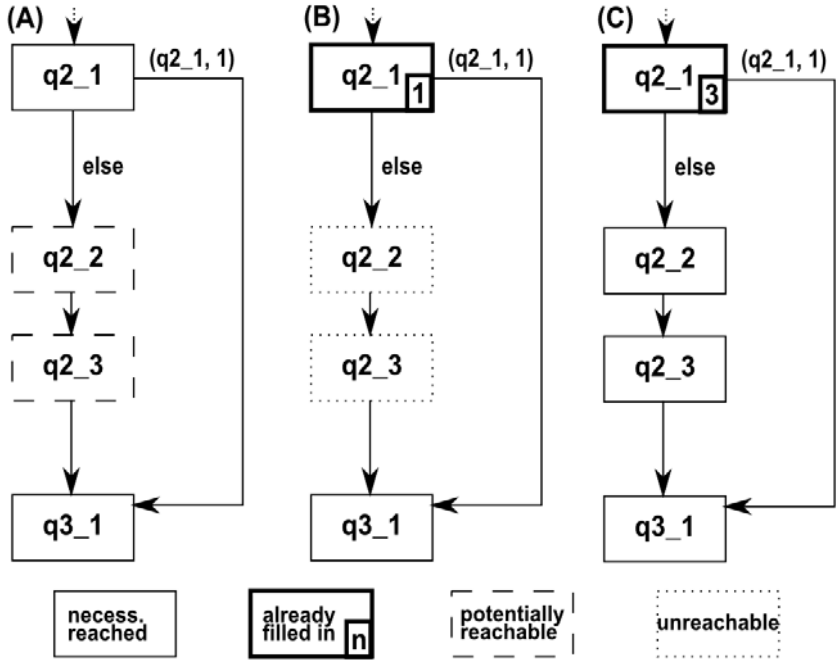

Figure 3: the update of GQ node states, depending on the answers given

31. Each time the user inserts a new answers or updates a previous one, the state of all nodes has to be updated. As a consequence the algorithm to recompute the node states needs to be extremely efficient, as it may be executed many times on web clients (PCs + browsers) with unknown, possibly low, performances, during the questionnaire compilation. Furthermore, no assumption can be made on the quality of the network connection between client and server and this makes all solutions based on the continuous interaction between client and server (e.g. the Ajax-based ones) totally unfeasible.

32. The implemented solution is based on the caching (on both client and server side) of all metadata describing the structure of the GQ. In other words, although the GQ structure is stored in the meta-DB, that DB is only accessed once at the start of the application and the retrieved data cached in main memory to avoid

further unnecessary and inefficient accesses. Similarly the GQ structure is passed to the client together with the HTML page and used repeatedly by a local routine to perform the update of the question states.

33. The algorithm for node state update has a very low complexity: thanks to the acyclicity property of the GQ the whole update can be performed by analyzing each edge of the graph only once, i.e. in linear time. Very briefly, the idea underlying the algorithm is that, given the answers already provided by the user, a question has to be enabled for compilation, if the user will necessarily reach it, independently of the responses still to be given. On the contrary, a question has to be disabled if she/he will certainly not reach it or there is at least one sequence of responses (for the questions still to be given) that will prevent her/him from reaching that question.

VI. The search engine for assisted coding of open (textual) questions

34. One of the requirements for QPOP was to support the respondent in the semi-automated coding of the question on highest educational qualification, which refers to a classification dictionary constituted by more than 6,000 distinct items. In the paper questionnaire this is an “open” question, i.e. the respondent has to write a free text describing her/his highest educational qualification, and the coding of this textual answer is deferred to the OCR-based data capture phase, where unmatched cases (texts not matching any of the dictionary items) are resolved by human intervention.

35. Conversely, it was required that QPOP would provide an already coded title (in other words QPOP could not store a free text to be coded later, either manually or semi-automatically). A selection based on a dropdown list was obviously unfeasible, due to the large number of possible items. Instead, the respondent had to be guided to self-encode her/his educational qualification, starting from its (possibly imprecise) textual description. To this aim a specialized search engine was implemented, totally integrated in the online questionnaire user interface, and using a sophisticated mechanism, based on dictionary pre-processing, similarity string comparison and automatic ranking.

36. Before they can be effectively used by the QPOP search engine, the single dictionary items need to be pre-processed by a specific procedure. This procedure is relatively heavy from the computational point of view and is consequently executed offline, starting from the final, stable version of the dictionary. The main steps of the procedure are the following:

a) *Character normalization*: accented letters are replaced with the corresponding unaccented version, uppercase letters with lowercase ones, other characters like punctuation marks are removed
b) *Stopword removal*: “useless” words are removed from the character-normalized version of the items, produced in the previous step. Both “general” (like conjunctions, articles, etc.) and “context-specific” stopwords (e.g. the word “degree”, when considering a list of academic degrees) are removed
c) *Search terms extraction and weighting*: the single terms (words) constituting the normalized items produced by the previous two steps are extracted and stored in the search engine DB tables. A weight is also assigned to each term, depending on its relative frequency inside the dictionary: less frequent terms will receive a higher weight, as they are more selective, i.e. more effective to filter out a large number of possible candidate items

37. The DB tables produced by the dictionary pre-processing described above are used by the search engine to efficiently produce a list of candidate items in response to a search string provided by the user. The input of the search engine is constituted by: (i) the response given by the user to a previous (closed) question on educational qualification, which has to be used by the search engine to apply the correct set of stopwords, as well as to pre-filter the set of candidate items; (ii) the search string provided by the user as a description of her/his highest educational qualification; and (iii) the pre-processed dictionary of qualifications, stored in the search engine DB tables, as described above.

38. The output of the search engine is a list of candidate educational qualifications, extracted from those available in the dictionary, filtered according to the response previously given by the respondent, and ranked according to their similarity with the search string provided. In general the ranking technique assigns higher scores if (i) one or more searched terms produce exact matches (terms are exactly equal and not simply very

similar to those in the dictionary item), (ii) the match is with one or more “rare”, highly selective terms, (iii) the dictionary item includes less extra-terms (i.e. terms that have no correspondence in the search string)

39. The main steps of the search engine procedure are illustrated in Figure 4. The first three are very similar to those illustrated for dictionary pre-processing, but are applied on the search string, instead of the whole dictionary, namely:

a) *Character normalization*: accented letters in the search string are replaced with the corresponding unaccented version, uppercase letters with lowercase ones, other characters like punctuation marks are removed
b) *Stopword removal*: “useless” words are removed from the character-normalized version of the string produced in the previous step; both “general” (like conjunctions, articles, etc.) and “context-specific” stopwords (e.g. the word “degree”, when dealing with academic degrees) are removed
c) *Search terms extraction*: the single terms (words) constituting the normalized search string produced by the previous two steps are extracted and used for the actual search

40. The next three steps start from a list of normalized terms to be searched and produce a ranked list of dictionary items. More specifically:

a) *Similarity search*: each (normalized) term to be searched is compared with those in the database; the terms that produce a similarity above a given (relatively high) threshold are passed to the following step
b) *Extraction of the dictionary items*: the dictionary items containing one or more terms obtained in the previous step are extracted from the DB. At the same time, for each item extracted, some values are either read or computed, which will be used in the following step to compute the score of each item and consequently produce a ranked list of results
c) *Dictionary item sorting*: by using the values extracted/computed in the previous step, the score of each item in the result set is computed and the list is sorted accordingly in descending order. This sorted list is proposed to the respondent, who will choose the one that best corresponds to her/his educational qualification (or try a new search with a different string, if the resulting list is unsatisfactory).

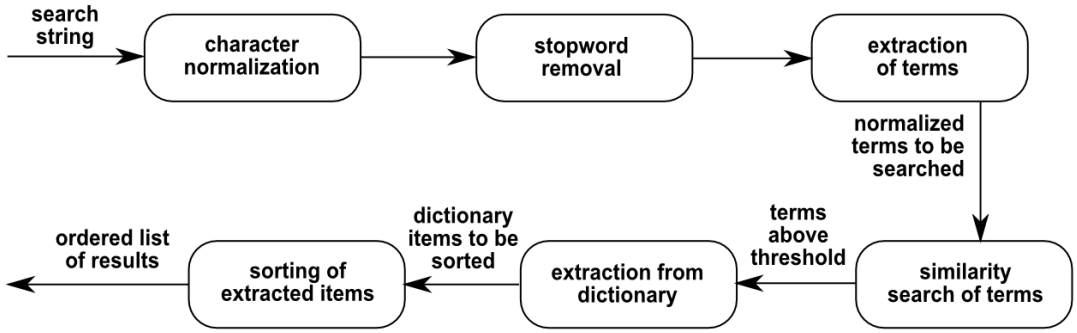

Figure 4: the main steps of the search engine for assisted encoding of open questions

41. The similarity threshold value which determines the inclusion (or not) in the list of matching terms can be freely modified and its correct calibration is very important. Excessively low threshold values can produce a large number of useless correspondences. In contrast, if the threshold value is too high (very close to 1) this can restrict the results only to exact matches. After some comparative experiments a threshold value of 0.95 was found to be a good trade-off. The adopted similarity algorithm is a normalized version of the well-known Jaro-Winkler distance measure, as implemented in the simMetrics Java open-source library.

42. The advantage of having a similarity comparison instead of an exact match is twofold. Firstly, it produces positive results even in the presence of little typing errors, like character transposition, single missing characters, etc. Secondly, it also produces the inclusion of closely related terms, e.g. a similarity match for the terms “psychologist” and “psychological” together with the exact searched term “psychology”. On the other hand, as stressed above, the similarity threshold must be sufficiently high (close to 1) to avoid a large number of false “similar terms”, i.e. matches that are not really related to what searched by the respondent.