InterviewStreamliner, a minimalist, free, open source, relational approach to computer-assisted qualitative data analysis software

Hans Pruijt
Erasmus Universiteit Rotterdam
pruijt@fsw.eur.nl

Preprint version
Accepted for publication in Social Science Computer Review, Volume 28, No. 2, Summer 2010

abstract:
InterviewStreamliner is a free, open source, minimalist alternative to complex computer-assisted qualitative data analysis packages. It builds on the flexibility of relational database management technology.

keywords: computer-assisted qualitative data analysis, CAQDAS, qualitative research

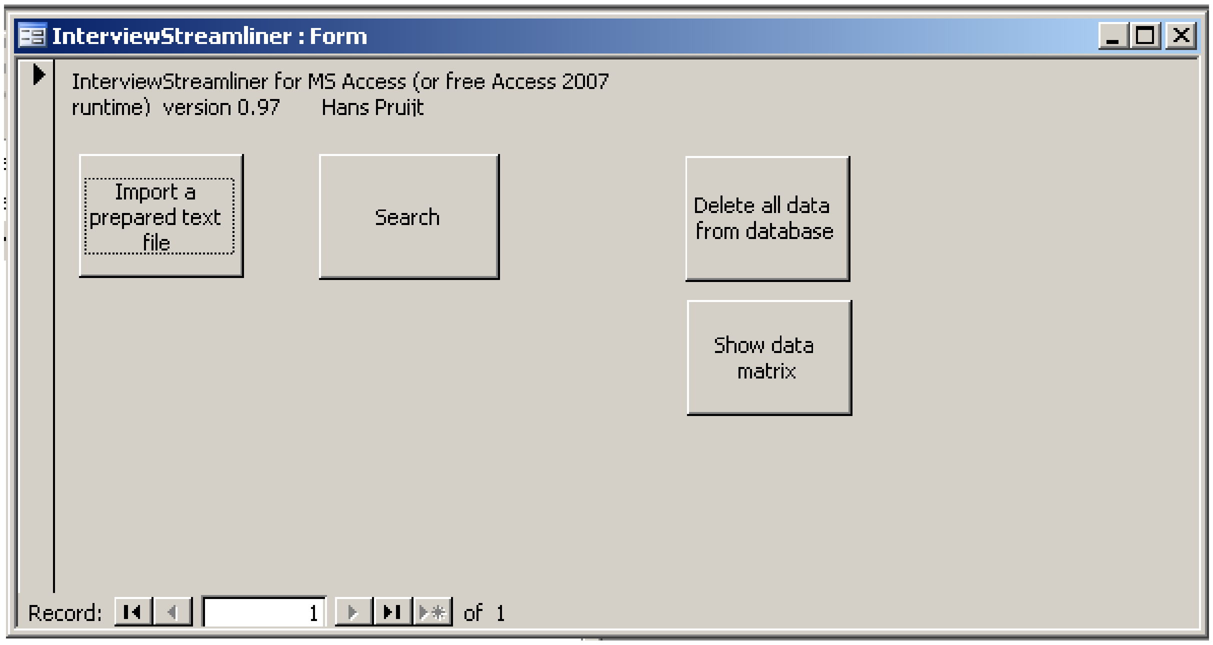
The currently fashionable computer-assisted qualitative data analysis software (CAQDAS) packages are complex, which results in considerable learning curves and costs. Nevertheless, their core functionality remains simple: coding, retrieving and recoding text fragments (Lewins 2007; 2009). This contradiction spurred the development of InterviewStreamliner. The goal was to create a free CAQDAS package that is as easy to use as possible and leverages existing skills. Also, an important wish was to eliminate the need for software installation, a hurdle to deployment in a centrally controlled computing environment. The strategy was to capitalize on the widespread availability of relational database management systems, such as MS Access. Relational database management systems are geared to the world of parts numbers and customer names, not so much to the textual data that is characteristic of qualitative research (Pruijt 1991; Schipper 1991). In this area, there has been little improvement over time. Additional programming is still required to obtain the desired functionality. InterviewStreamliner provides this. It was implemented as a MS Access application. On computers that already have MS Office Professional installed, no further software installation is necessary. Users who do not have MS Access can instead run InterviewStreamliner with the Access 2007 runtime, which is freely downloadable from the Microsoft website.

One design decision was to leave as much as practical to the word processing and operating system software that the users are already familiar with. Thus, initial coding is done in the word processor. This involves creating (a) text file(s) in this format:

```
@
_keyword A_ _keyword B_ _keyword C_
Text fragment ......................................................
.........................................................................................
............................text.....................................................
@
_keyword B_ _keyword C_
Text fragment ......................................................
.........................................................................................
............................text.....................................................
@
```

The @-sign separates fragments or scenes, keywords are marked by enclosing them in underscores (_).

InterviewStreamliner's minimalism is apparent when it starts (figure 1).


InterviewStreamliner : Form

InterviewStreamliner for MS Access (or free Access 2007 runtime) version 0.97 Hans Pruijt

Figure 1 Startup form

After importing one or more input files, fragments can be retrieved (figures 2 and 3).

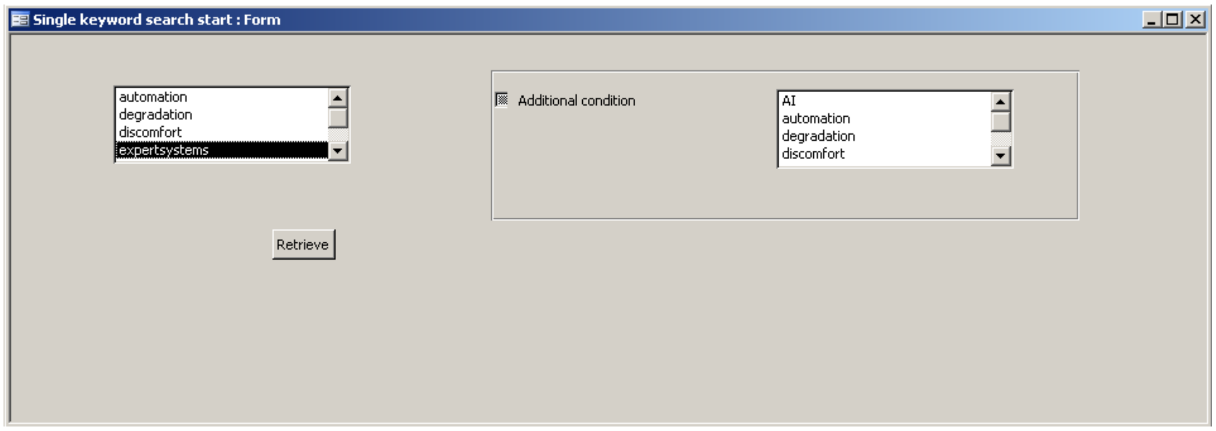
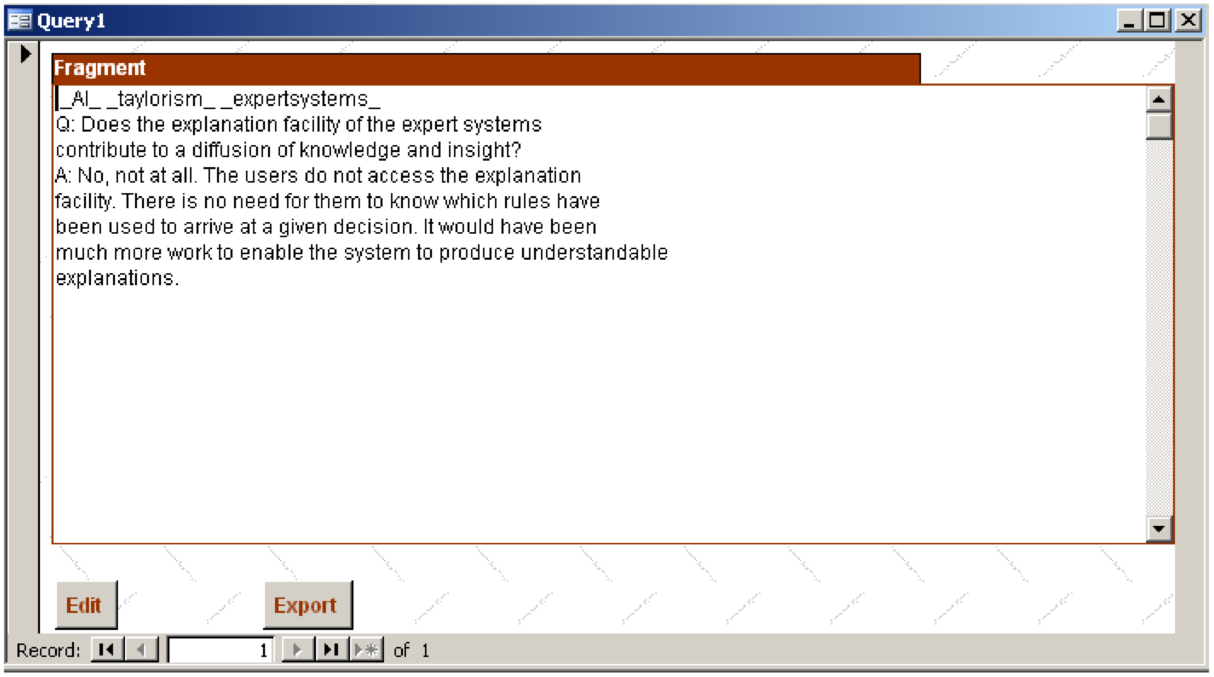


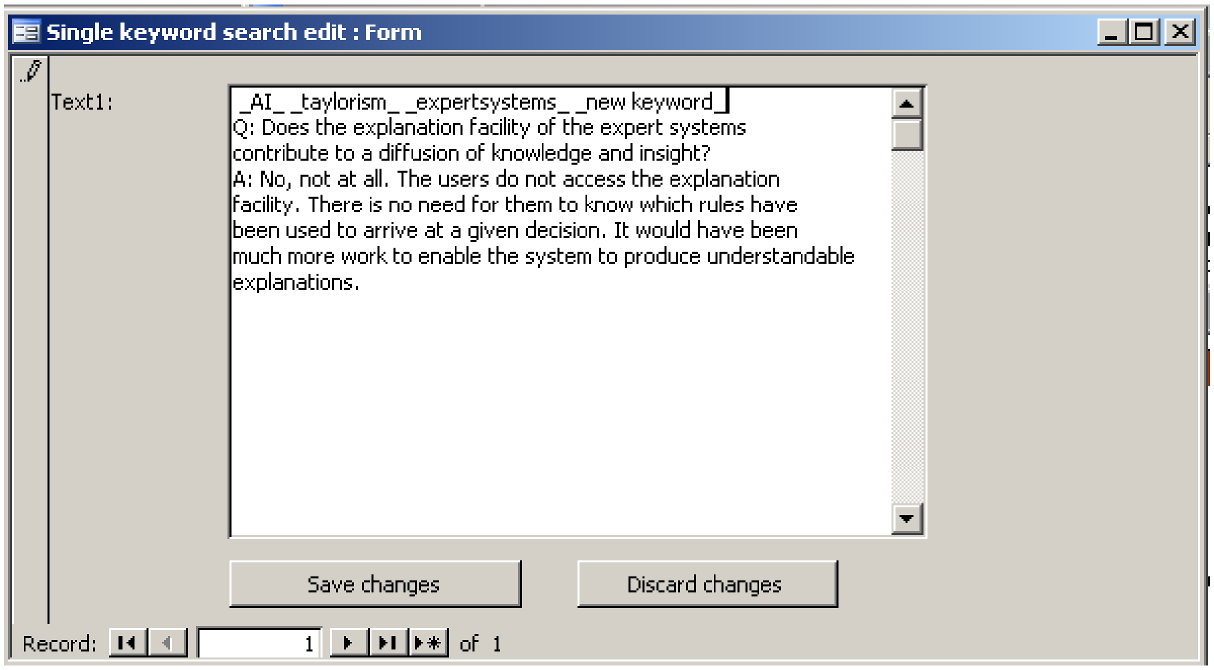
Figure 2 retrieval



```
_AI_ _taylorism_ _expertsystems_
Q: Does the explanation facility of the expert systems
contribute to a diffusion of knowledge and insight?
A: No, not at all. The users do not access the explanation
facility. There is no need for them to know which rules have
been used to arrive at a given decision. It would have been
much more work to enable the system to produce understandable
explanations.
```

Figure 3 output

From here, the user can edit the record or export the retrieved records to an external file. Recoding takes place in the editing form (figure 4).



_AI_ _taylorism_ _expertsystems_ _new keyword_
Q: Does the explanation facility of the expert systems contribute to a diffusion of knowledge and insight?
A: No, not at all. The users do not access the explanation facility. There is no need for them to know which rules have been used to arrive at a given decision. It would have been much more work to enable the system to produce understandable explanations.

Figure 4 editing

Users who find it useful to link retrieved fragments back to their original context can, when preparing the input file(s), insert a special, unique label at the top of each interview (or any other type of section). When building the database, InterviewStreamliner copies this label to each fragment of the interview. The optional label must be preceded by a ~ and be in an otherwise blank line. See the example below.
Suppose that the file consists of two interview transcriptions. The first interview is labeled as: "~systems analyst 1". The second interview is labeled as "~systems analyst 2". After processing the line "~systems analyst 1", InterviewStreamliner will copy this label to each subsequent fragment, until it encounters the line "~systems analyst 2". From that point onwards, each fragment automatically gets labeled "~systems-analyst 2".

```
@
~systems analyst 1
@
_degradation_ _automation _
In our organization, the job of programmer does not amount to
much. For making applications, we use program generators. One
enters a few parameters, and the program generator makes the
complete program of a transaction-processing system.
@
_user participation_ _power_
After some time, he broke down. First, we gave him a booklet
on technical banking terms and acronyms: "read this, it will
help you understand what we are doing". Then he participated
in a meeting of Systems Programming, in which they kept
```

```
introducing words that were completely new to him. Then we
gave him a glossary of specific terms and abbreviations for
the project. This was followed by yet another meeting in which
he again found himself bombarded with completely new
terminology. The final blow was when I handed him a report on
“overnight processing in local areas”. He asked to be sent
back to his boss. In retrospect, this was the most frustrating
thing that you can do to somebody.
@
_power_
Not only are we sitting in an ivory tower, but we also laid a
minefield around us. This happened when we stopped asking the
users about how they did their work and, instead, began to
determine the work flow ourselves. When I had completed a
manual, somebody said: couldn’t you relate this to how the
work used to be done? I said no, because it was not relevant.
Plus, I did not even know exactly how the work process had
been before.
@
~systems analyst 2
@
_degradation_
Q I we look at jobs that are being hollowed out, would it be
right to conclude that especially the job of systems analyst
is hollowed out?
A Yes. For instance, in the market group, someone, on the shop
floor level, gets as assignment to keep in touch with IT. Such
users typically know a lot about IT. They may, for example,
specify that the color of cars should be recorded, and also
say how it is to be done; "I want it in this table and not in
that one." The systems analyst is wedged between the user and
the programmer. In fact, he is more like a systems
administrator.
@
_user participation_
In practice, we tend to be too eager to please the users. We
built a system based on a regional division scheme that users
suggested, although management had already decided that is was
going to be changed.
@
```

Testing in the computer lab confirmed that students can quickly learn to use InterviewStreamliner. Problems that came up related to general computing skills; brushing up those skills has value way beyond the context of qualitative analysis. What should help motivation is that using InterviewStreamliner involves standard software found in countless workplaces.

The use of a relational database management system as a basis provides additional benefits. One benefit is that accessing the data from other programs is easy. And, when using not the runtime version but the full version of Access, all of its flexible database management functionality is available for creative use. Figure 5 shows the database structure.

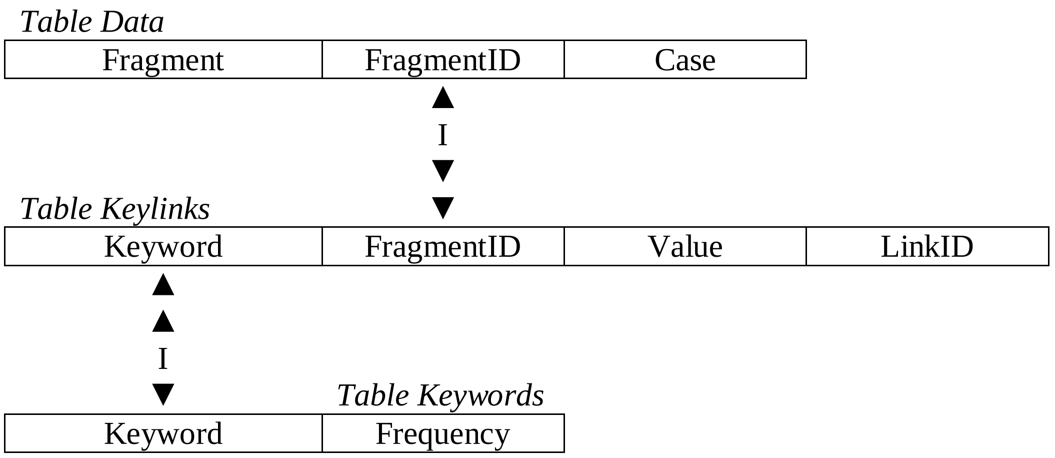


Figure 5 database structure

In this diagram, single arrowheads designate primary keys, double arrowheads foreign keys. The table "Data" holds the text fragments. These are stored in memo fields. In Access, Shift-F2 allows zooming in on the contents of such a field. Access can export the contents of this table to an external file.
The table "Keywords" contains each occurring keyword (once) along with its frequency.
The table "Keylinks" links the tables "Data" and "Keywords". In "Keylinks", there is a row for any occurrence of a keyword in a fragment.
Finally, because InterviewStreamliner leverages the capabilities of a sophisticated database management system, its own programming code can be relatively simple. This opens the program up to users who might be inclined to join in a further development as an open source project. The fact that it is based on a relational, normalized database should make modifying and extending the program easy sailing.

InterviewStreamliner is to be found at http://www.eur.nl/fsw/soc/mtict/software/downloads/ . A manual and technical documentation are also available.

Lewins, A. and C. Silver (2007) Using Software in Qualitative Research. A Step-by-Step Guide. Los Angeles: Sage.
Lewins, A. and C. Silver (2009). Choosing a CAQDAS Package. 6th edition. CAQDAS Networking Project and Qualitative Innovations in CAQDAS Project. Retrieved from http://caqdas.soc.surrey.ac.uk/PDF/2009%20Choosing%20a%20CAQDAS%20Package.pdf, May 20th, 2009

Pruijt, H. (1991) “Relational databases for textual data?” in Best, H., E. Mochmann and M. Thaller (eds), <u>Computers in the Humanities and the Social Sciences. Achievements of the 1980s. Prospects for the 1990s</u>., München, K. G. Saur, p. 414-422.
Schipper, D. (1991), “A plea for incorporating a textual datamodel in relational databases”, in Best, H., E. Mochmann and M. Thaller (eds), <u>Computers in the Humanities and the Social Sciences. Achievements of the 1980s. Prospects for the 1990s</u>., München, K. G. Saur, p. 423-426.