Handling irregular transcripts with regular expressions in a bash script

samgramconverter.sh/ bash 3.2

Author: Nikita Julie Myrting, student ID: 201906799

Link to github repository: https://github.com/NiGitaMyrGit/FinalProject-CDS

Keywords: *reguar expressions; bashscript; transcription conventions; conversation analysis; notational system*

1 Introduction / Goal

With this project I wish to make a user-friendly bash script that can convert one set of characters with another set of characters.

2 Problems and Background / Context

Conversation Analysis (CA) is the analysis of naturally occurring ordinary talk focused on the *structures* of interaction. It is an inductive qualitative method, where case-by-case-analysis leads to generalizations across cases. The building blocks for CA are transcriptions: A transcription is per se a representation of (speech) sounds in the auditory medium to symbols and characters in an orthographic system. Visual data accompanying the auditory is sometimes provided in the transcript as well.
The analysis marks these details: choices must be made on what details to focus on and what to leave out. In conversation analysis these notational systems are referred to as transcript(ion) conventions or a notational system.

Gail Jefferson, one of the founders of conversation analysis, was the first person to make a standardised system with notational symbols for pronunciational and temporal features of speech. The transcript conventions have gone through some changes throughout the years. Jefferson herself, has adjusted and tailored the conventions, as well as inspired other scholars to create alternative systems largely or loosely based on hers. “Most CA practitioners use some version of Jefferson’s notation system. In Germany, many practitioners use Gesprächsanalytisches Transkriptionssystem (GAT) notations (Selting et al., 1998, 2009; Deppermann & Schütte, 2008), which are less precise than Jefferson’s system with respect to measuring pauses, but more precise with respect to prosodic features, noted on a specific tier.” (Wagner, 2020, p. 301)

In this paper I will focus on the conventions primarily used at Aarhus University (AU), since it is the only university in Denmark where CA is an entire subfield of linguistics, and a course is taught and offered. The AU-located research group DanTIN study and describe the grammar of spoken Danish and are behind the webpage samtalegrammatik.dk. They provide their transcription conventions as well, which is a mix of Jefferson’s notational system alongside some symbols added from Talkbank’s transcription programme CLAN .
The transcript systems discussed in this paper primarily focus on speech, but multimodal transcription systems exist as well. Multimodal systems describe gaze and gestures of a speaker with pinpoint precision, and work as an extension often implemented alongside the speech-oriented systems.
In the conversation analytic research field, it is generally accepted to use any transcription conventions the transcriber prefers, as long as each symbols use and relevance is accounted for.
Along these notes transcription conventions are for the most part merely *described* in their usage and interpretation but seldomly *discussed.*
As Jefferson writes in *Glossary of transcript symbols with an introduction* “Although I’d probably rather transcribe than any [sic] do any other part of the work (…) the one thing I’d rather *not* do is talk about transcribing. It’s not a topic.… Transcribing is just something one does to prepare materials for analysis, theorizing, etc. ”(Jefferson, 2004, p. 13).
Henceforth she proceeds to discuss transcription practices, loosely referring to old and alternative conventions. This showcases a problem in the field of Conversational Analysis; the forgotten discussion

about transcription conventions, and how different each transcript can look and appear depending on the transcriber.
It is impossible for any transcriber to fully transcribe every small detail in talk in relation to auditory and multimodal features; it would be immensely time-consuming and result in a very complex and hardly readable transcript.
A transcript customarily has one or more focus points, and each transcript is provided with the necessary detail to showcase the phenomenon(s) and substantiate the analysis. (Jefferson, 2004, p. 15) (Wagner, 2020) The problem I wish to address and provide a solution for, is in the case of different notational systems providing the *same* meaning but using *different* notational symbols
I wish to relieve the reader of constantly looking up each sign in the transcript convention, alongside easing the work of scholars manually altering transcripts produced by others in accordance with their own preferred transcription conventions.
It is impossible for me to solve the problem of completely streamlining transcription.
But it is possible to pick the ripest apples and harvest them: to replace and substitute the symbols that have the exact same meaning but simply uses different notation. Many programs, whether a text editor such as Sublime text and TextEdit or actual software for transcribing such as CLANc, DOTE and ELAN have a search and replace function based on regular expressions (either as the ability to insert a pattern via point-click or the ability to type in a regular expression) implemented in the programme. In Elan you even have the possibility to find and replace on multiple .eaf files at once. The thing that these search functions lack is automatization and the ability to work with filetypes outside the scope of the file-format of the programme. If one want to replace all symbols possible from one convention to another, it is still a tedious task to search and replace each symbol manually.

What I wish to do is make a script, where a user easily can upload a transcription and have symbols from one notational system, that are 'translatable', replaced with those of another one. The script will be able to translate symbols from the Jefferson convention and SamtaleBank to samtalegrammatik's transcription conventions.
A CSV file containing 4 sheets have been placed in my GitHub repository to give an overview of the scopes of the different notational systems, alongside a file where I have all the possible replaceable symbols are listed next to each other.

3 Software Framework

The scripting and coding has been written on my MacBook Pro 14-inch, 2021 with 16 GB memory which runs on the macOS Ventura 13.0.1 operating system. The bash-script and code has been made and tested in terminal (version 2.13) running on the UNIX shell bash (3.2) and written with the text editor Sublime Text (build 4143). Furthermore I have uploaded all my documents to the GitHub repository here: https://github.com/NiGitaMyrGit/FinalProject-CDS.git

4 Data Acquisition and Processing

The bash script has been made with great inspiration from the youtubevideo and article *3 Hours Bash tutorial* provided by Talha Saif Malik (2019) on the channel linuxhint (2019).
The code itself has also been made with help from the Youtube channel Luke Smith, (2018) and Shawn Powers (2022) alongside a sed manual (*Sed, a Stream Editor*, n.d.)
I have tested my script on the transcript *anne_og_beate* downloaded from the Danish SamtaleBank Sam2 corpus (Wagner, 2009) .

5 Implementation

I have provided a README file with a throughout description on how to run and use the script.

For my bashscript called samgramconverter.sh I have used a conditional statement. It is located in the github repository https://github.com/NiGitaMyrGit/FinalProject-CDS.git in the path Final_project > bash-script-unix-shell > samgramconverter.sh

Description of the code in samgramconverter.sh:

In line 1 I make sure the script run in bash with the command #! /bin/bash
In line 3-5 I have a comment that would only be readable if the script itself was opened.
In line 7 is the first echo with text that will be printed out when the user runs the script.
In line 8 I use the command *read* so the computer can read the file input the user provides by typing in the filename and hitting enter.
In line 10 I begin a conditional statement: if [[-f $fileName]]. The -f flag checks whether the file exist or not, and the $ is used to refer back to the fileName in line 8 that the user has provided. In Bash it is generally advised to use double brackets, since it gives one options to run more commands. (Shawn Powers, 2022) It also means that my script now is not suitable for any other shell format than bash.
If the user does *not* provide an existing files the script will go to the *else* option in line 24 which prints out that the input fileName does not exist.
If the user has provided an existing file, the script will run the command provided in line 19.

The sed command:

I have used the sed command (McMahon, n.d.), which enables me to use some regular expressions. I will go through some of them: sed -e "s/£/☺/g" the sed command reads the specified files, -e is a flag that appends the output after the first command has run to the next one allowing me to write several expressions in one. Enclosed in quotation marks is the command sed performs "s/£/☺/g". s/ is equivalent to the linear expression substitute. I have used slashes as delimiters but any delimiters, except the star * and backslashes\ could have been used. £ the pound sterling sign is the input to be replaced. Between the next set of slashes is the smiley symbol ☺ which will replace the £. A final slash ends the regular expression followed by a /g. The g is a flag that makes the replacement globally. If the g is not added, only the first encounter of the symbol in a line will be replaced, but with g added it replaces all the encounters.

In the end of every line in 12-18, I have added a backslash which allows me to put the function on the next line while still running. It makes no difference to the script but gives a much better overview of the many sed commands run.

In line 14, 15 and 18 have a lot of the same sed commands, and there is a reason for this
Using "s/Δ/>/" as an example. Since SamtaleBank uses two identical symbols, the triangles ΔΔ to bracket fast speech whereas samtalegrammatik.dk uses two different symbols: a right caret > and a left caret <. After having a bit of a headache to come up with a solution. First the sed command "s/Δ/>/" runs and replaces the first Δ in a line with a > non-globally. Then the "s/Δ/</" command runs which replaces the next Δ in the text, providing the correct result. The problem now arises if multiple triangle pairs occurs in a transcript on the same line. For this reason i chose to run a cycle of the two command three times, since it nearly impossible to have space for more than two 'fast brackets' on the same line, and the third cycle is just to be on the safe side. The same goes for the upside-down triangles ∇ ∇ in line 15 enclosing slow speech being replaced by a left and a right carat < >, and in line 18 with two question marks on either side ?? ?? to be replaced by parentheses ( ). With the question marks it was important that this code was run before the "s/?/↗/" was run in line 19, since this code otherwise would have replaced all the ?? with double diagonal upwards arrows ↗↗.

Finally "/[a-zA-Z]/s/\./↘/g" -e "/[()]/s/\↘/./g" is an interesting case. The trouble here was that if s/\./↘/g" was run a dots . were replaced with a diagonal downwards arrows ↘ inside parenthesis signaling pauses. With /[a-zA-Z]/ signals that the command should only be run 'in the environment of upper and lower case letters'. By this it stopped replacing the dots when bracket in digits as in measured pauses eg. (0.7). But for some reason sed interprets parenthesis as letters as well, so after having run the first command   I could use the "/[()]/s/\↘/./g to replace the ↘ in the environment of parentheses back to dots.

After the last sed command I put in the $fileName and then append a new output file with the same name (and extension) of the original one but with samgrammed_ in front.
The reader is informed of this with the echo command in line 21.
The fi in line 25 exits the conditional statement.
To test out the script I have provided to examples in the GitHub repository Final_project > examples > which contains to files: anne-og-beate.cha (Wagner, 2009) and funny_carrot.txt.
The .cha file is a format used for the transcription software CLAN.
The .txt file includes all the symbols the script is able to substitute. I have tested them both and it works.

6. Critical evaluation and conclusion

I thought that using regular expressions and a UNIX shell would ease the pain, but I am not sure it did. I encountered many problems and many symbols I was not able to replace, due to the fact that another convention I initially planned to use as well (Hepburn & Bolden, 2012) uses many of the same symbols as Jefferson (2004) but for completely different meanings. The solution would be to make a separate script when translating this convention, but I did not have time for it. Same goes for deleting remaining symbols, that are not used in the samtalegrammatik.dk. Here a new bash script cold be made that deletes all the notational symbols, if the user wanted to do so.
The script works, but is in many ways only halfway useful. With more research and time-at-hand I believe more symbols could have been replaced. Alas, the sed functions is limited and can only interpret different lines: not different words. The solution would he be to split all the words into separate words. This can be done in the Sublime Text editor, but it would not really be user-friendly if the user themselves should do this task. Perhaps the real solution would have been not to use bash, but instead something like python (or a language called The *Perl 5 language interpreter* that can be implemented in a UNIX shell!(Wall, n.d.))
My goal of putting less strain on the reader, when reading the transcript. has not been successful. In fact, quite the contrary; if the reader would have to keep track of two systems instead of one, it would only make things *more* complicated.

The script is however not *entirely* useless: even though a transcriber might want to add and subtract additional symbols, it still saves them half the time of manually finding and replace solutions.

References

Hepburn, A., & Bolden, G. B. (2012). The Conversation Analytic Approach to Transcription. In J. Sidnell & T. Stivers (Eds.), *The Handbook of Conversation Analysis* (1st ed., pp. 57–76). Wiley. https://doi.org/10.1002/9781118325001.ch4

Jefferson, G. (2004). Glossary of transcript symbols with an introduction. In G. H. Lerner (Ed.), *Pragmatics & Beyond New Series* (Vol. 125, pp. 13–31). John Benjamins Publishing Company. https://doi.org/10.1075/pbns.125.02jef

linuxhint (Director). (2019, May 27). *Bash Scripting Full Course 3 Hours*. https://www.youtube.com/watch?v=e7BufAVwDiM

Luke Smith (Director). (2018, December 15). *Using `sed` and Regular Expressions (Unix/Linux command line)*. https://www.youtube.com/watch?v=QaGhpqRll_k

Malik, T. S. (2019). *3 Hour Bash Tutorial*. https://linuxhint.com/3hr_bash_tutorial/

McMahon, L. E. (n.d.). *A sed command* (Version 7) [AT&T UNIX].

Sed, a stream editor. (n.d.). Retrieved 12 January 2023, from https://www.gnu.org/software/sed/manual/sed.html

Shawn Powers (Director). (2022, March 17). *BASH Conditionals Make Scripting AWESOME! (IF/THEN/ELSE/ELIF/CASE)*. https://www.youtube.com/watch?v=QALD8r0JQPY

Wagner, J. (2009). *SamtaleBank Danish Sam2 Corpus* [Data set]. TalkBank. https://doi.org/10.21415/T5B88Z

Wagner, J. (2020). Conversation Analysis: Transcriptions and Data. In C. A. Chapelle (Ed.), *The Encyclopedia of Applied Linguistics* (1st ed., pp. 1–8). Wiley. https://doi.org/10.1002/9781405198431.wbeal0215.pub2

Wall, L. (n.d.). *The Perl 5 language interpreter*. Retrieved 12 January 2023, from https://www.perl.org/

- B Required Metadata

Please fill in the right column column with the correct information about your digital resources, and leave the left columns as they are

Table 1 – Software metadata

Nr	**Software metadata description**	***Please fill in this column***
S1	Current software version	Terminal Version 2.13, bash 3.2
S2	Permanent link to Github repository where you put your script or R project	https://github.com/NiGitaMyrGit/FinalProject-CDS.git
S3	Legal Software License	GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007
S4	Computing platform / Operating System	Apple MacBook pro 14 inch, 2021, MacOS Ventura version 13.0.1
S5	Installation requirements & dependencies for software not used in class	You need to have the UNIX shell bash installed. On mac terminal it is preinstalled but running on zsh shell. On windows bash will have to be downloaded. To see how read the README file in the GitHub repository.
S6	Support email for questions	201906799@post.au.dk