Overview

<2018-12-19 Wed>

Notes for creating VT PSYC 4364 Senior Seminar “Cogntive Neuroscience of Decision Making” Spring 2019.

Activities

Critical reading questions

Make list of submitted questions

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Download from Canvas<HTML> </HTML> <HTML> </HTML>Make directory for the week, e.g.<HTML> </HTML>

```
mkdir ~/Google Drive/teaching/CNDM/discussions/wk2_gustation/article/submissions/
```

<HTML> </HTML>Move<HTML> </HTML>

```
submissions.zip
```

<HTML> </HTML>to the “submissions” directory for the week.<HTML> </HTML> <HTML> </HTML>Unzip it, and REMOVE THE ZIP FILE. The script that handles extracting the text from the html files needs the submissions/ dir. to include ONLY html files.<HTML> </HTML><HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Extract questions from html files using scripts<HTML> </HTML> <HTML> </HTML>Edit this file to have the correct directory names defined near the top:<HTML> </HTML>

```
~/Google Drive/teaching/code/Canvas/text/extract_text_from_html_files.sh
```

<HTML> </HTML>This file runs xsltproc using an associated .xsl script:<HTML> </HTML>

```
~/Google Drive/teaching/code/Canvas/text/getQuestionsCNDM.xsl
```

<HTML> </HTML>Run the script from whichever directory you like, because the script uses absolute paths for filenames.<HTML> </HTML>

```
bash ~/Google Drive/teaching/code/Canvas/text/extract_text_from_html_files.sh
```

<HTML> </HTML>This will make a new file, whose name is defined in the script; CHANGE it to something appropriate, but not “final,” because it will need to be edited by hand and then run through another script. Example:<HTML> </HTML>

```
article_questions_UNSORTED.txt
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Edit the questions by hand<HTML> </HTML> <HTML> </HTML>Good regexp to replace: non-ascii characters. In particular, the script above often gets a non-printing character that looks like an orange underscore in emacs (it appears there, but not in other programs that display the .txt file).<HTML> </HTML> <HTML> </HTML>In emacs, to replace non-ascii characters with nothing:<HTML> </HTML>

```
M-x replace-regexp
[^[:ascii]]

```

<HTML> </HTML>Ideally each question line begins with a category like “Clarification:”.<HTML> </HTML> <HTML> </HTML>To sort the lines using emacs, select all lines, then:<HTML> </HTML>

```
M-x sort-lines
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Clean up the lines<HTML> </HTML> <HTML> </HTML>They aren't always given the correct category labels, etc.<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Organize the questions into themes<HTML> </HTML>
<HTML> </HTML>DON'T put spaces in between lines to separate the themes. That can be done after the next script appends question numbers to each line.<HTML> </HTML> <HTML> </HTML>Try these emacs procedures:<HTML> </HTML> <HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Highlight one term<HTML> </HTML>

```
M-x highlight-regexp
```

<HTML> </HTML>and also M-x unhighlight-regexp.<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Kill and yank all lines with term<HTML> </HTML>
<HTML> </HTML>I tried using M-x sort-regexp-fields but couldn't get it to do what I wanted, which was to sort all lines with a given term in them next to each other.<HTML> </HTML>
<HTML> </HTML>However, found this emacs function, which allows me to do that by killing all such lines together so that they can be yanked back together, all next to each other!<HTML> </HTML>
<HTML> </HTML>From https://www.emacswiki.org/emacs/KillMatchingLines<HTML> </HTML>

```
(defun kill-matching-lines (regexp &optional rstart rend interactive)
  "Kill lines containing matches for REGEXP.

See `flush-lines' or `keep-lines' for behavior of this command.

If the buffer is read-only, Emacs will beep and refrain from deleting
the line, but put the line in the kill ring anyway.  This means that
you can use this command to copy text from a read-only buffer.
\(If the variable `kill-read-only-ok' is non-nil, then this won't
even beep.)"
  (interactive
   (keep-lines-read-args "Kill lines containing match for regexp"))
  (let ((buffer-file-name nil)) ;; HACK for `clone-buffer'
    (with-current-buffer (clone-buffer nil nil)
      (let ((inhibit-read-only t))
        (keep-lines regexp rstart rend interactive)
        (kill-region (or rstart (line-beginning-position))
                     (or rend (point-max))))
      (kill-buffer)))
  (unless (and buffer-read-only kill-read-only-ok)
    ;; Delete lines or make the "Buffer is read-only" error.
    (flush-lines regexp rstart rend interactive)))
```

<HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Save to a new name<HTML> </HTML>
<HTML> </HTML>If name generated by the script was *$_{UNSORTED}$.txt, do *$_{SORTED}$.txt.<HTML> </HTML><HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Append question numbers<HTML> </HTML>
<HTML> </HTML>Run this script on the hand-edited file:<HTML> </HTML>

```
~/Google Drive/teaching/code/Canvas/text/add_question_numbers.sh [INPUT FILE
NAME ending in .txt]
```

<HTML> </HTML>The output file will be named like the input file, but with the .docx extension instead of .txt. It will be in MS Word format.<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Add spaces between themes<HTML> </HTML>
<HTML> </HTML>Edit the .docx file by hand.<HTML> </HTML> <HTML> </HTML>Select all and change all text to bold.<HTML> </HTML><HTML>
</HTML><HTML>
</HTML>

Discussions

Collaborative document

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Download document in two formats<HTML> </HTML> <HTML> </HTML>Using Google Docs, export file as both .docx and

.txt.<HTML> </HTML> <HTML> </HTML>Suggested filename: wk[number]_[topic], e.g. $wk2_{gustation}$.docx.<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Bash script to detect usernames in collaborative document<HTML> </HTML> <HTML> </HTML>Change filenames in this script, then run it using bash:<HTML> </HTML>

```
~/Google Drive/teaching/code/Canvas/text/find_usernames.sh
```

<HTML> </HTML>Will produce an output file; current example name:<HTML> </HTML>

```
username_scores.txt
```

<HTML> </HTML>Gives count of each username (case-insensitive) found in .txt version of collaborative document.<HTML> </HTML><HTML>
</HTML><HTML>
</HTML>

Discussion leaders

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Assigning groups<HTML> </HTML> <HTML> </HTML>There are 12 weeks that need student discussion leader groups<HTML> </HTML> <HTML> </HTML>There are 28 students.<HTML> </HTML> <HTML> </HTML>28/12 = 2.33<HTML> </HTML> <HTML> </HTML>2a + 3b = 28 a + b = 12<HTML> </HTML> <HTML> </HTML>b = 12 - a<HTML> </HTML> <HTML> </HTML>2a + 3(12 - a) = 28 -a + 36 = 28 a = 8<HTML> </HTML> <HTML> </HTML>b = 4<HTML> </HTML> <HTML> </HTML>8 groups of 2 4 groups of 3<HTML> </HTML> <HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Use shuf in bash to randomize list of students<HTML> </HTML>

```
shuf usernames_key_HANDFIXED_GOOD.txt > groups_shuffled_list.txt
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Select groups in order from randomized list<HTML> </HTML> <HTML> </HTML>Take first 8 pairs, then 4 triples.<HTML> </HTML> <HTML> </HTML>Assign names: GroupA, GroupB, etc.<HTML> </HTML> <HTML> </HTML>Save to document:<HTML> </HTML>

```
groups_list.txt
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Assign groups on Canvas<HTML> </HTML> <HTML> </HTML>Name “group set:” $discussion_{leadergroups}$<HTML> </HTML> <HTML> </HTML>Select “I'll create groups manually.”<HTML> </HTML> <HTML> </HTML>Create groups GroupA etc.<HTML> </HTML> <HTML> </HTML>Drag student names into their groups.<HTML> </HTML><HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Post survey to Canvas<HTML> </HTML> <HTML> </HTML>Make it an assignment, not a quiz/survey. This way, it can be a group assignment on Canvas.<HTML> </HTML>

<HTML> </HTML>Make submission “Text Entry” and ask groups to list their top three preferences in order.<HTML> </HTML><HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Summary document<HTML> </HTML>
<HTML> </HTML>2019-08-26<HTML> </HTML> <HTML> </HTML>Made template .docx file<HTML> </HTML>

```
~/Google Drive/teaching/CNDM/syllabus/CNDM_discussion_summary_TEMPLATE.org
```

<HTML>
</HTML><HTML>
</HTML>

Materials

Username list

These are in a file:

username$_{list}$.txt

Making the list

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>List of component words<HTML> </HTML> <HTML> </HTML>I used the “Mnemonics” usernames generated using website<HTML> </HTML>
<HTML> </HTML>https://www.michaelfogleman.com/phrases/<HTML> </HTML>
<HTML> </HTML>These usernames are pairs of words taken from the Mnemonic Encoding Word List.<HTML> </HTML>
<HTML> </HTML>https://gist.github.com/fogleman/c4a1f69f34c7e8a00da8<HTML> </HTML>
<HTML> </HTML>http://web.archive.org/web/20090918202746/http://tothink.com/mnemonic/wordlist.html<HTML> </HTML>
<HTML> </HTML>http://web.archive.org/web/20091003023412/http://tothink.com:80/mnemonic/wordlist.txt<HTML> </HTML> <HTML> </HTML>I have copied the list to a file for future use:<HTML> </HTML>
<HTML> </HTML>mnemonic$_{wordlist}$.txt<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Generating two-component usernames<HTML> </HTML> <HTML> </HTML>2019-08-26<HTML> </HTML>
<HTML> </HTML>[Couldn't find any notes on how I did this for the first semester, Spring 2019, so let's assume I did it by hand then.]<HTML> </HTML> <HTML> </HTML>By hand in emacs, made version of mnemonic$_{wordlist}$.txt that has only the words, capitalized, in a single column:<HTML> </HTML>

```
~/Google Drive/teaching/CNDM/admin/usernames/mnemonic_word_list_one_column.txt
```

<HTML> </HTML>Then can use “shuf” command to randomize:<HTML> </HTML>

```
shuf mnemonic_word_list_one_column.txt > randomized_list.txt
```

<HTML> </HTML>Finally, can then just remove every other newline to join together pairs of words (by hand), until have enough for the class to choose from.<HTML> </HTML>
<HTML> </HTML>[Decided against the following, seems too annoying for students' use: Can also append “F19”, “S20”, etc. to beginning of strings to make it easy to tell when sections added to the documents, especially for the use of the website documents.]<HTML> </HTML>
<HTML> </HTML>Delete the rest of the column. Save result as<HTML> </HTML>

```
usernames_for_papercutter.txt
```

<HTML> </HTML>for that instance of the course.<HTML> </HTML> <HTML> </HTML>Will most likely want to remove some combinations by hand, e.g. AlcoholSomething, or anything with a proper name (which are included in the mnemonic words). IMPORTANT: Also compare to past lists to try and avoid duplicating even single components that have been used in the past, if possible. Probably easiest to do this using the file “randomized$_{list}$.txt”.<HTML> </HTML> <HTML> </HTML>After letting students choose names from pieces of paper, cull the list and call it<HTML> </HTML>

```
usernames_list.txt
```

<HTML>
</HTML><HTML>
</HTML>

Making key to match usernames with student names

Download submissions to a non-credit “assignment” from Canvas for which student submitted the text of their usernames.

Edit the top of this script by hand to set the directories on which it will run.

```
~/Google\ Drive/teaching/code/Canvas/text/extract_usernames.sh
```

This will produce a file

```
usernames_key.txt
```

which has two columns: students' last names and usernames. NOTE however that if students have two-word last names, only the final name will get printed.

To overcome this limitation, and to produce a document that can printed and used as an attendance checklist, can run this version of the previous script:

```
~/Google\ Drive/teaching/code/Canvas/text/extract_realnames.sh
```

which produces

```
realnames_key.txt
```

This prints three names for all students (prints an “x” for middle name of those with only two names on Canvas) as well as the username, in four columns.

IMPORTANT NOTE: after compiling the list using scripts, e.g.

```
extract_usernames.sh
```

MUST check for non-printing characters, which will often be included in output of adc's “extract from Canvas html” scripts. Emacs will display, e.g. the non-printing character that looks like an underscore in emacs. Remove all of these! Otherwise the script that uses grep to find usernames in the collaborative document

```
find_usernames.sh
```

will fail for names that have the non-printing characters as part of the username string.

Email list

Online search for “Canvas email list” led me to:

https://community.canvaslms.com/thread/9885

Which included instruction for using Canvas API via URLs:

```
James Jones Thank you for your thorough API Request information both here and
on another page I visited yesterday. Using what you shared, I was able to
piece together a solution for my situation and figured I ought to share it to
benefit the community. Applying your API request to the URL looks something
like this:

https://canvas.institution.edu/api/v1/courses/XXX/users?enrollment_type[]=stud
ent&include[]=email&per_page=100&page=1

To modify this to your case there are three things to know:

canvas.institution.edu needs to be your organization's page
XXX needs to be the course number
If there are more than 100 students, you increase the page value at the end of
the URL to get the next 100 students.
You can leave out the "while(1);" from the beginning of the JSON code, copy
the rest, and paste it into a converter.
```

```
P.S. If there are more than 100 students, replace the ending hard bracket with
a comma and add the next page's results to the string before running the
converter to get it compiled into a single document."
```

Link to a “converter”:

http://www.convertcsv.com/json-to-csv.htm

I could have used converter, but instead I just pasted the text from the resulting web page into a file

```
email_line.txt
```

and used grep:

```
grep -o "[a-z|0-9]*@vt.edu" email_line.txt  | sort | sed s/$/,/g >
email_list.txt
```

“-o” option makes grep print only the matching text.

sed command adds commas to ends of lines – did this so could paste list into Zotero group membership invite web page.

Zotero group

I used the email list (above) to invite students to the group I created:

```
cognitive_neuroscience_decision_making
```

2019-09-01

Last week, I backed up the Spring 2019 semester's zotero library in two ways, and then deleted it from the online group to make way for the Fall 2019 semester's students.

1. Exported library to zotero rdf format, added to a new directory called

```
~/Google Drive/teaching/CNDM/zotero_group_lib/
```

1. Moved all the files within zotero out of the shared group folder to the the CNDM folder under “teaching.”

Publishing

Website

2019-08-05

This is about publishing at least the article discussion summary documents on dokuwiki.

Currently the idea is to do this process separately for each week's discussion.

Summary list of steps for posting summary documents to dokuwiki web pages

- Hand edit .docx file
    - Add Heading “Topic: [topic name]” at top
        - Add one-sentence explanation of topic if necessary
    - Add “Article Discussed” section
        - Add reference to article
    - Move neuroimaging section to after Brief Summary
        - Add sentence with link to Neurosynth
    - Fix formatting, at very least Top 5 Pubmed Articles text
    - Move the Unanswered Questions to main questions section
    - Make “Questions Posed by Class” L1 heading
        - Make/check topic L2 headings for questions
- On Linux terminal
    - Modify and run the bash script to generate dokuwiki .txt file from .docx
- On dokuwiki site
    - Make a dokuwiki page for the topic
    - Paste contents of dokuwiki .txt file
    - Upload image files using media manager
    - Hand edit formatting as necessary
    - Add link to Zotero library to topic page

Prepare the summary document

Anthony decided to change the format, compared to the assignment instructions, for the web pages.

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Modify the .docx file<HTML> </HTML> <HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>First section is reference to the article<HTML> </HTML> <HTML> </HTML>This was not part of the Spring 2019 semester instructions, so add it if necessary.<HTML> </HTML> <HTML> </HTML>Section name (headling level 1): “Article Discussed”<HTML> </HTML> <HTML> </HTML>Body of section is a citation.<HTML> </HTML> <HTML> </HTML>Ideally this reference will appear in the Bibliography section, but don't mess with Zotero if fixing retroactively.<HTML> </HTML><HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Put the neuroimaging parts near the beginning, instead of at the end.<HTML> </HTML> <HTML> </HTML>Currently (based on the instructions for the Spring 2019 semester) these come after the Brief

Summary.<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Check the .docx document for any weird or nonstandard formatting, and fix it.<HTML> </HTML> <HTML> </HTML>Example: the “Top 5 Pubmed articles” section in $wk2_{gustation}$ was formatted as a set of single lines with newlines at end of each, for some reason.<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Move the two “Unanswered Questions” text into the thematic sections of answered questions<HTML> </HTML> <HTML> </HTML>Because they have been answered now!<HTML> </HTML><HTML>
</HTML><HTML>
</HTML>
<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Convert from docx to dokuwiki format using pandoc<HTML> </HTML> <HTML> </HTML>The following will create a directory named “./media” with image files in it named image1.png, etc.<HTML> </HTML>

```
pandoc --extract-media . input.docx -o output.txt -t dokuwiki
```

<HTML> </HTML>This is from https://stackoverflow.com/questions/39956497/pandoc-convert-docx-to-markdown-with-embedded-images<HTML> </HTML><HTML>
</HTML><HTML>
</HTML>

Make the dokuwiki .txt file for the web page

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Fix the dokuwiki file<HTML> </HTML> <HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Bash script to automate the following steps<HTML> </HTML>

```
~/Google\ Drive/teaching/code/dokuwiki/cndm_fix_dokuwiki_txt.sh
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Change the media (image) file names<HTML> </HTML> <HTML> </HTML>Assuming that the media files are all image files (I don't know how different files might get named), the files will be named “image1.png”, “image2.png”, etc.<HTML> </HTML> <HTML> </HTML>The plan is to upload multiple summary documents' image files to the same dokuwiki namespace, e.g. “teaching:imageFileName.png”. The dokuwiki media manager will want unique names for all of the various files. Therefore, it won't work to leave generic image file names like “image1.png” after converting each individual summary document, because then there will be multiple files with the same name.<HTML> </HTML> <HTML> </HTML>Solution: append a name for the specific summary document to the image file names. Can do this in the ./media/ dir:<HTML> </HTML>

```
rename 's/.png/_gustation.png/' *.png
```

<HTML> </HTML>Then will need to change the references to the image files in the dokuwiki .txt document.<HTML> </HTML> <HTML> </HTML>Here are example lines from a dokuwiki .txt document:<HTML> </HTML>

```
====== Neurosynth map for the term: ======
```

```
{{:./media/image1.png?570x570|Macintosh
HD:Users:acate:Downloads:mni_icbm152_t1_tal_nlin_asym_09a_acMasked (2)2.png}}
```

<HTML> </HTML>Need to<HTML> </HTML> <HTML>
</HTML>
<HTML>
</HTML>Change “image1.png” to “image1$_{gustation}$.png” (for this summary document about gustation)<HTML>
</HTML> <HTML>
</HTML>Change the path name from “./media/” to “teaching:”<HTML>
</HTML> <HTML>
</HTML>Remove the image alt text “Macintosh HD:Users … (2)2.png”<HTML>
</HTML><HTML>
</HTML>

```
sed 's/.png/_gustation.png/g' -i.bak dokuwikiFileName.txt
sed 's/:\.\/media\//teaching:/g' -i.bak dokuwikiFileName.txt
sed 's/|.*}}/|}}/g' -i.bak dokuwikiFileName.txt
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Remove the table of contents<HTML> </HTML> <HTML> </HTML>If there is one in the .docx document. It will get reproduced as plain text after pandoc conversion to dokuwiki.<HTML> </HTML><HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Remove question labels<HTML> </HTML>
<HTML> </HTML>Remove the “Clarification:” etc. labels, and remove the numbers after the “Q[0-9]+”. Remove spaces around the terms, too<HTML> </HTML>

```
sed 's/\(Q[0-9]*[[:space:]]*\)[Cc]larification:[[:space:]]*/\1/g' -i.bak
dokuwikiFileName.txt
sed 's/\(Q[0-9]*[[:space:]]*\)[Ii]mplication:[[:space:]]*/\1/g' -i.bak
dokuwikiFileName.txt
sed 's/\(Q[0-9]*[[:space:]]*\)[Rr]elated:[[:space:]]*/\1/g' -i.bak
dokuwikiFileName.txt
```

<HTML> </HTML>NOTE: In many or all .docx files the questions are in bold text, which will get converted to text wrapped in pairs of “**” in dokuwiki format.<HTML> </HTML>
<HTML> </HTML>To un-bold the question text only, and not all bold text, use this command by searching for the Q[0-9]* labels. It gets rid of the pair of “**”. The “\1” refers to the pattern identified in the first part of the s///g statement using $$.<HTML> </HTML>

```
sed 's/\*\*\(Q[0-9]*.*\)\*\*/\1/g' -i.bak dokuwikiFileName.txt [BUT SEE BELOW;
CAN DO ADDITIONAL STUFF IN THIS COMMAND AS WELL]
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Make each question a subheading<HTML> </HTML> <HTML> </HTML>[NOTE: AS DESCRIBED BELOW, I DECIDED TO SCRAP THIS IDEA; did continue to set questions to headings, but at a higher level for bigger text. Just seems like a good idea in case want to link to questions in the future.]<HTML> </HTML>
<HTML> </HTML>Could do this in MS Word, etc.<HTML> </HTML>
<HTML> </HTML>Better to do this after converting to dokuwiki .txt file, by searching for “Q[0-9]+” lines and wrapping them in appropriate number of “=”.<HTML> </HTML>
<HTML> </HTML>Best to combine this step with the un-bolding step from above. Also remove numbers after the “Q” here.<HTML> </HTML>

```
sed 's/\*\*Q[0-9]*\(.*\)\*\*/== Q: \1 ==/g' -i.bak dokuwikiFileName.txt
```

<HTML> </HTML>Then would need to set dokuwiki table of contents (toc) to exclude that level of heading. (This should be the case by default for level 5 headings).<HTML> </HTML>
<HTML> </HTML>Using two equals signs (“==”) corresponds to a level 5 heading in dokuwiki. The resulting html will be like:<HTML> </HTML>

```
<h5 id="qwhat_is_gustatory">Q:  What is gustatory?</h5>
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Make list of question section links<HTML> </HTML> <HTML> </HTML>Then could make list of all questions across all pages for different topic, create links to their sections (?).<HTML> </HTML>
<HTML> </HTML>Appears that could find text for making links by looking at the html source for the dokuwiki pages, specifically by searching for headings of a specific level:<HTML> </HTML>

```
<h5 id="qwhat_is_gustatory">Q:  What is gustatory?</h5>
```

<HTML> </HTML>The id gets appended to the page URL when navigating, e.g., via the table of contents to a specific section:<HTML> </HTML>

```
http://visneuro.psyc.vt.edu/doku.php?id=teaching:cndm_topic_gustation#qwhat_is_gustatory
```

<HTML> </HTML>The conversion of the messy heading text (messy because of whitespace, capital letters) gets converted to a simpler string by dokuwiki.<HTML> </HTML>
<HTML> </HTML>Presumably would use xslt to extract the ids and the heading text, and output html that could be embedded in a dokuwiki page.<HTML> </HTML><HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Tweak the automatically generated table of contents<HTML> </HTML> <HTML> </HTML>This assumes dokuwiki has the plugin “toctweak” installed.<HTML> </HTML> <HTML> </HTML>Append this to top of file to prevent higher-numbered headings from appearing in toc:<HTML> </HTML>

```
"~~TOC 1-2~~"
```

<HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML>
<HTML>
</HTML><HTML> </HTML>Manual changes<HTML> </HTML>
<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Link to zotero library<HTML> </HTML> <HTML> </HTML>After posting the web page (as below), add a link to the zotero library for the discussion at top (just before the first heading “Article Discussed”).<HTML> </HTML> <HTML> </HTML>Here is the base URL for the course's Zotero site:<HTML> </HTML>

```
https://www.zotero.org/groups/2279132/cognitive_neuroscience_decision_making/items?
```

<HTML> </HTML>For gustation, the link is:<HTML> </HTML>

```
https://www.zotero.org/groups/2279132/cognitive_neuroscience_decision_making/items/collectionKey/2Z7TGN6D
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Add spaces and horizontal rules arouund sections<HTML> </HTML> <HTML> </HTML>Scheme:<HTML> </HTML> <HTML>
</HTML> <HTML>
</HTML>Horizontal rule (with spaces surrounding) at end of L1 sections <HTML>
</HTML> <HTML>
</HTML>Except for first two (“Topic:” and “Article Discussed”)<HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML> <HTML>
</HTML>Extra empty line (“\\”) after L1 headings <HTML>
</HTML> <HTML>
</HTML>Except for first two, which also get one before<HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML> <HTML>
</HTML>Two extra empty lines before and one after L2 headings<HTML>
</HTML> <HTML>
</HTML>Extra empty line before L3 headings (which should always have “Q: ” text) <HTML>
</HTML> <HTML>
</HTML>Except for first one in section<HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML><HTML>
</HTML>

<HTML> </HTML>Tried using sed, but way too confusing, so used emacs.<HTML> </HTML> <HTML> </HTML>Wrote new section into<HTML> </HTML>

```
cndm_fix_dokuwiki.sh
```

<HTML> </HTML>Also have script that does this for a single file. Need to edit input file name in script. Can use to fix existing dokuwiki pages that have already had extensive hand edits, by copying text from page to a temp file, running script on that file, then pasting output back to dokuwiki.<HTML> </HTML>

```
txt2md_cndm_dokuwiki.el
```

<HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Make a list of links to individual questions<HTML> </HTML> <HTML> </HTML>Make the list in dokuwiki syntax, so can paste the resulting text into a dokuwiki page.<HTML> </HTML> <HTML> </HTML>Use xlstproc to process the html code of the web page you have just made for the discussion. [Note: this assumes that you have actually done some of the following steps below, which I know is out of order.]<HTML> </HTML> <HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>XSL template used as reference for writing this code<HTML> </HTML> <HTML> </HTML>Anthony consulted xls templates he had made previously for other activities to remember how to write xlst code:<HTML> </HTML>

```
~/Google Drive/teaching/code/Canvas/text/getQuestionsCNDM.xsl
```

<HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>Format for dokuwiki links to follow<HTML> </HTML> <HTML> </HTML>[Note not typing the double square brackets below, because emacs org mode will display them and their contents as a link in this here file, which is inconvenient.]<HTML> </HTML> <HTML> </HTML>Section headings (the thematic headings

created by authors of the summary document) ought to be L2 headings. (There should be a L1 heading “Questions posed by students” to begin the section.)<HTML> </HTML> <HTML> </HTML>[Don't actually type the square brackets below]<HTML> </HTML> <HTML> </HTML>[newline] Section Heading text [newline] (2x [)Question URL|Question Heading text(2x]) [newline] [next one] [newline] ...<HTML> </HTML> <HTML> </HTML>UPDATE: Too time consuming for Anthony to figure out xslt code for selecting the topic headings, so just make a list of all the questions:<HTML> </HTML> <HTML> </HTML>(2x [)Question URL|Question Heading text(2x]) [newline] [next one] [newline] ...<HTML> </HTML><HTML> </HTML> <HTML> </HTML><HTML> </HTML>xsl file<HTML> </HTML>

```
~/Google Drive/teaching/code/dokuwiki/makeQuestionLinkList.xsl
```

<HTML> </HTML>It works great! However, I decided not to use it, see below.<HTML> </HTML><HTML>
</HTML> <HTML>
</HTML><HTML> </HTML>PROBLEM: Links don't take visitor to section!!!<HTML> </HTML> <HTML> </HTML>2019-08-06<HTML> </HTML> <HTML> </HTML>If go to address bar and hit Enter, it will work, otherwise, the location is off!!!!!!<HTML> </HTML> <HTML> </HTML>I tried using “absolute” URLs (not internal dokuwiki page links), but same result!<HTML> </HTML> <HTML> </HTML>Also got spam warning when did the latter, turned off spam blacklisting, which solved the spam problem but not the location problem. Turned spam blacklisting back on.<HTML> </HTML> <HTML> </HTML>Also, the list is overwhelming to skim. It seems like readers might as well browse the actual web page itself.<HTML> </HTML> <HTML> </HTML>DECIDED TO SCRAP THE WHOLE LINKS TO QUESTIONS PROJECT!!<HTML> </HTML><HTML>
</HTML><HTML>
</HTML> <HTML>
</HTML><HTML>
</HTML>

Post the dokuwiki page

1. Create a new link for the discussion on the main course page
2. Paste the contents of the dokuwiki .txt file
3. Upload the image files (./media) using the media manager

Add search bar to the main web page

Can make it specific to the namespace, if have installed the “searchform” dokuwiki plugin.

See its page for syntax.

Add graphviz graph of production process

Wrote graphviz dot code to make a nice graph. Can copy and paste this code to dokuwiki, wrapped in tags:

```
<graphviz dot left>
</graphviz>
```

File:

```
~/Google Drive/teaching/CNDM/website/graphviz/cndm_document_process.gv
```

Code to export to image:

```
dot -Tpng ~/Google\
Drive/teaching/CNDM/website/graphviz/cndm_document_process.gv -o ~/Google\
Drive/teaching/CNDM/website/graphviz/cndm_document_process.png
```

Export to .svg for use of graphic elsewhere: substitute -Tsvg and *.svg in code above, i.e.:

```
dot -Tsvg ~/Google\
Drive/teaching/CNDM/website/graphviz/cndm_document_process.gv -o ~/Google\
Drive/teaching/CNDM/website/graphviz/cndm_document_process.svg
```

PDF

Moved course to sub-namespace in teaching namespace

Namespace is now “teaching:cndm”. Main topic page is now “teaching:cndm:cndm”.

So that can export entire namespace, and none of the other teaching pages, to a single PDF

Make export link using dw2pdf plugin

Can use dw2pdf dokuwiki plugin

See the plugin page for syntax.