Principles of Complex Systems, Vols. 1 and 2
CSYS/MATH 6701, 6713
University of Vermont, Fall 2025
"Oh my god. I get it. I get it."
Assignment 13

Community: Mac Finds His Pride, S13E10
Episode links: IMDB, Fandom, TV Tropes.

Due: Friday, February 6, by 11:59 pm
https://pdodds.w3.uvm.edu/teaching/courses/2025-2026pocsverse/assignments/13/
Some useful reminders:
Deliverator: Prof. Peter Sheridan Dodds (contact through Teams)
Office: The Ether and/or Innovation, fourth floor
Office hours: See Teams calendar
Course website: https://pdodds.w3.uvm.edu/teaching/courses/2025-2026pocsverse
Overleaf: LaTeX templates and settings for all assignments are available at https://www.overleaf.com/read/tsxfwwmwdgxj.

Some guidelines:

1. Each student should submit their own assignment.
2. All parts are worth 3 points unless marked otherwise.
3. Please show all your work/workings/workingses clearly and list the names of others with whom you ~~conspired~~ collaborated.
4. We recommend that you write up your assignments in LaTeX (using the Overleaf template). However, if you are new to LaTeX or it is all proving too much, you may submit handwritten versions. Whatever you do, please only submit single PDFs.
5. For coding, we recommend you improve your skills with Python. And it's going to be a no for the catachrestic Excel. Please do not use any kind of AI thing unless directed. The (evil) Deliverator uses (evil) Matlab.
6. There is no need to include your code but you can if you are feeling especially proud.

Assignment submission:

Via **Brightspace** (which is not to be confused with the death vortex of the same name, just a weird coincidence). Again: One PDF document per assignment only.

Please submit your project's current draft in pdf format via Brightspace.

Semester goal: A paper based on a large-scale text or corpus, building through assignments.

Four stories to analyze:

- **Pride and Prejudice**
  https://www.gutenberg.org/ebooks/1342
- **Frankenstein; or the Modern Prometheus**
  https://www.gutenberg.org/ebooks/84
- **Moby Dick; or, The Whale**
  https://www.gutenberg.org/ebooks/2701
- **Les Misérables**
  https://www.gutenberg.org/ebooks/135

Data:

For this assignment, the novels have been processed into 1-grams with an attempt to capture all elements including punctuation.

You can use the data below, or what you produced in the previous assignment.

The basic data format is as a time series with one 1-gram per line (links below).

For each story, also linked to below are the rank distributions of 1-grams by counts.

https://pdodds.w3.uvm.edu/permanent-share/pride_and_prejudice_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/pride_and_prejudice_1grams.txt

https://pdodds.w3.uvm.edu/permanent-share/frankenstein_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/frankenstein_1grams.txt

https://pdodds.w3.uvm.edu/permanent-share/moby-dick_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/moby-dick_1grams.txt

https://pdodds.w3.uvm.edu/permanent-share/les_miserables_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/les_miserables_1grams.txt

Instrument—Shifterator:

- For word shifts, use either:

  The Python package described in Ref. [1].

  Or the D3.js shifterator created by Andy Reagan here:
  https://observablehq.com/@andyreagan/d3-shifterator-v4-preloaded.

  Links to paper versions (arXiv is always best), Github repository, and an exhilarating Twitter feed can be found here:
  https://pdodds.w3.uvm.edu/research/papers/gallagher2021a/.

  Various Matlab versions made by the Unreliable Deliverator do exist and need to be shared on Gitplaces. A more sophisticated map+list version has long been in development.

  Github repository: https://github.com/ryanjgallagher/shifterator

  Very Important Notes for the Python version:

  - The default setting for reference happiness score for the labMT lexicon in the Shifterator **is wrong**. It's set at 5. Please make sure to use the average of the reference text. Again: **It's wrong as is**, and this is due to a misunderstanding that cannot be righted. There is only so much the Deliverator can do.
  - You will need Python 3.8. Apparently it all goes to pieces otherwise.

Things to do:

1. (3 points)

   Lexical calculus (a subset of type calculus):

   Derive the word shift equation for simple additive lexical instruments.

   You will have the derivation per class.

   The idea is to simply work through it yourself.

   There are no advanced mathematics here.

   But over and over, people do not understand what's going on.

   Word shifts are a kind of discrete derivative (difference) with words on the inside.

   Per lectures, the goal is to derive:

$$\delta h_{\mathrm{avg},i} = \frac{100}{\left| h_{\mathrm{avg}}^{(\mathrm{comp})} - h_{\mathrm{avg}}^{(\mathrm{ref})} \right|} \underbrace{\left[h_{\mathrm{avg}}(w_i) - h_{\mathrm{avg}}^{(\mathrm{ref})} \right]}_{+/-} \underbrace{\left[p_i^{(\mathrm{comp})} - p_i^{(\mathrm{ref})} \right]}_{\uparrow/\downarrow} .$$

   Performed in class and in numerous papers [2, 3, 4].

2. (3 points total)

   Task: Take the UTF-8 text versions of each of these three novels and parse them into narrative time series of 1-grams which include all punctuation, numbers, and words.

   See below for instructions.

   To report: For each novel, present your output for the first paragraph, rendered as a single, wrapped line.

3. (3 points total)

   Task: For each novel, determine the size rankings of 1-grams with size being counts.

   To report: Display the 1-grams and counts for the first 30 1-grams (suggest a single table with four columns).

4. (3 points total)

   For each novel, plot the size rank distribution for 1-grams and estimate the exponent α in $S_r \propto r^{-\alpha}$.

Outputs provided for comparison:

The basic data format is as a time series with one 1-gram per line (links below).

For each story, also linked to below are the rank distributions of 1-grams by counts.

https://pdodds.w3.uvm.edu/permanent-share/pride_and_prejudice_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/pride_and_prejudice_1grams.txt

https://pdodds.w3.uvm.edu/permanent-share/frankenstein_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/frankenstein_1grams.txt

https://pdodds.w3.uvm.edu/permanent-share/moby-dick_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/moby-dick_1grams.txt

General instructions for processing texts:

- Using an editor, remove the start and finish material for each of the three novels that Gutenberg adds to books somewhat inconsistently.
- Using a judicious selection of regex operations, create a script* (Python is strongly recommended but you can use whatever you like) to break up the text into meaningful 1-grams that may be words, punctuation, and numbers.
  Regex = Regular Expression.
  * If you want to use tokenizer, you can. But the idea is to be careful and really understand what's happening the text as you smash it into pieces (storyons).
- See excerpt of Perl code below for an example. Yes, Perl. Regex is regex.

```
## for Frankenstein
$text =~ s/D--n/Damn/g;

## separate out some basic punctuation
$text =~ s/([\!\?\,\.])/ \1 /g;
$text =~ s/:/ : /g;
$text =~ s/;/ \; /g;

## remove underscores used for emphasis
$text =~ s/_//g;

## isolate parentheses
$text =~ s/\(/ ( /g;
$text =~ s/\)/ ) /g;
```

```
## dash madness
$text =~ s/----/ --- /g; ## long dash
$text =~ s/--/ --- /g; ## em dash
$text =~ s/;-/ --- /g; ## em dash
$text =~ s/-/ --- /g; ## em dash

## handle specific salutations
$text =~ s/Mr \./Mr./g;
$text =~ s/Mrs \./Mrs./g;
$text =~ s/Dr \./Dr./g;

## clean up white space duplication
$text =~ s/\s+/ /g;

## separate quotes
## opening quotes should have a space before them (except for em dashes, treat
$text =~ s/\s"/ " /g;
## closing quotes should be what's left:
$text =~ s/"/ " /g;
$text =~ s/"/ " /g;
$text =~ s/"/ " /g;

## separate off opening single quotation mark
$text =~ s/‘/ ‘ /g;

## dyspunctional catapostrophes:
## clean up apostrophes and opening and closing single quote mark
## closing single quote mark should generally be isolated
## leave alone to preserve contractions
$text =~ s/‘/ ' /g;

$text =~ s/’/'/g;
## opening quote mark
## \p{L} stands for any UTF-8 letter
$text =~ s/(\s)'(\p{L})/\1' \2/g;
## closing quote mark (will be a problem with contractions)
$text =~ s/(\p{L})'(\s)/\1 '\2/g;

## split off possession indicator
$text =~ s/'s/ 's/g;
```

```
## remove any white space at the front
$text =~ s/^\s+//;

## add new line at the end
$text = $text."\n";

## now create time series text by replacing spaces with returns
($timeseriestext = $text) =~ s/ /\n/g;

## last: remove any white space redundancies
$timeseriestext =~ s/\s+/\n/ms;
```

5. (3 points)

   Measure the average happiness of each text using the labMT word list with the lexical lens:

$$\mathcal{L} = \{\tau \in \Omega \| h_{\rm avg}(\tau) \leq 4 \text{ or } h_{\rm avg}(\tau) \geq 6\} \tag{1}$$

   where Ω is the labMT lexicon and τ is a word in Ω.

6. (3 points each for 18 points)

   For the following $T_{\rm ref}$ and $T_{\rm comp}$ for each novel, generate a collection of word shifts as described below.

   Continue to use the same lexical lens $\mathcal{L}$ as above.

   Pride and Prejudice:

   - Pride and Prejudice:
     $T_{\rm ref}$ = the first 40% of the book,
     $T_{\rm comp}$ = 70% to 75% of the book.
   - Frankenstein:
     $T_{\rm ref}$ = the first 20% of the book,
     $T_{\rm comp}$ = the last 10% of the book.
   - Moby Dick:
     $T_{\rm ref}$ = the whole book,
     $T_{\rm comp}$ = 80% to 90% of the book.
   - Les Misérables:
     $T_{\rm ref}$ = the whole book,
     $T_{\rm comp}$ = 80% to 90% of the book.

For each novel, perform the following:

(a) produce word shifts comparing text $T_{\rm comp}$ relative to text $T_{\rm ref}$. Important: Use the average happiness of text $T_{\rm ref}$ as the baseline (again, this is not the default in the Python package).

(b) Interpret the word shifts. Does what you see make sense? Are there any surprises? Are some words being used in what the average person might not think is their primary meaning? For example, "crying" in Moby Dick means yelling, and "sick" can mean "awesome."

(c) Reverse the comparison: Produce word shifts comparing text $T_{\rm ref}$ relative to text $T_{\rm comp}$, but now now use the average happiness of text $T_{\rm comp}$ as the baseline.

(d) Comment on any asymmetries you see between the word shifts of (a) and (c) (the basic word shifts we use are asymmetric).

(e) Now go back and again produce word shifts comparing text $T_{\rm ref}$ relative to text $T_{\rm comp}$, but this time use 5 as the baseline reference score (neutral on the happiness-sadness spectrum of 1–9 for labMT).

(f) Compared to your first word shifts, how interpretable are these ones?

7. (0 points, just do it)

   A start on the semester project:

   - Please use Overleaf for writing up your project.
   - Build your paper using:
     https://github.com/petersheridandodds/universal-paper-template
   - Please use Github and Gitlab to share the code and data things you make.
   - For this first assignment, just getting the paper template up is enough.

   Some of you may of course have projects underway from PoCS, Vol. I.

   If not, please begin formulating project ideas.

   See storyology slides.

   General suggestion: Come up with some rich, text-based set of stories or corpora for analysis.

   For example: One (longish) book, or a book series, or a TV series.

   Less so stories: Could be legal writings, government documents, the Epstein files, ...

   Data would be the original text (books), subtitles, screenplay, or scripts (TV series).

   - You must be able to obtain the full text.
   - You will want something with at least around 10^5 words. More than 10^6 would be great.
   - Transcripts of shows may be good for extracting temporal character interaction networks.

   Please talk about possibilities with others in the class.

   For this assignment, simply list at least one possibility (which may be your existing project), noting the approximate text size in number of words, or whatever measure of size makes sense.

   - Maybe: Explore and find inspiration in the Pudding.

     Example: Film dialogue analysis.
     https://pudding.cool/2017/03/film-dialogue/
   - ConvoKit: Cornell Conversation Analysis Toolkit
   - Open Psychometrics. Everything except the which character study, which we will cover later.

References

[1] R. J. Gallagher, M. R. Frank, L. Mitchell, A. J. Schwartz, A. J. Reagan, C. M. Danforth, and P. S. Dodds. Generalized word shift graphs: A method for visualizing and explaining pairwise comparisons between texts. EPJ Data Science, 10:4, 2021. Available online at https://arxiv.org/abs/2008.02250. pdf

[2] P. S. Dodds and C. M. Danforth. Measuring the happiness of large-scale written expression: Songs, blogs, and presidents. Journal of Happiness Studies, 11(4):441–456, 2009. pdf

[3] P. S. Dodds, K. D. Harris, I. M. Kloumann, C. A. Bliss, and C. M. Danforth. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. PLoS ONE, 6:e26752, 2011. pdf

[4] P. S. Dodds, E. M. Clark, S. Desu, M. R. Frank, A. J. Reagan, J. R. Williams, L. Mitchell, K. D. Harris, I. M. Kloumann, J. P. Bagrow, K. Megerdoomian, M. T. McMahon, B. F. Tivnan, and C. M. Danforth. Human language reveals a universal positivity bias. Proc. Natl. Acad. Sci., 112(8):2389–2394, 2015. Available online at http://www.pnas.org/content/112/8/2389. pdf