With Thanks to Woolf and emacs, Reading 'The Waves' with Stephen Ramsay

Chris Forster

`http://cforster.com/2013/02/reading-the-waves-with-stephen-rams`

I am currently teaching a graduate course (eng630: "Digital Humanities": Emerging Tools and Debates in Literary Study) and, as much as possible, I'm trying to make clear the mechanics behind some of the text-analysis in the works we're reading. So, this week, as I prepared to discuss Stephen Ramsay's *Reading Machines*, I wanted to reproduce some of the analysis done there. The first chapter, for instance, offers a *tf-idf* reading of Woolf's *The Waves*. Here is how Ramsay describes it:

> It is possible—and indeed an easy matter—to use a computer to transform Woolf's novel into lists of tokens in whcih each list represents the words spoken by the characters ordered from most distinctive to least distinctive term. *Tf-idf*, one of the classic formulas from the field of information retrieval, endeavours to generate lists of distinctive terms for each document in a corpus. We might therefore conceive of Woolf's novel as a 'corpus' of separate documents (each speaker's monologue representing a separate document), and use the formual to factor the presence of a word in a particular

> speaker's vocabulary against the presence of the word in other speakers' vocabularies. (11)

This post summarizes how I tried to do just that, and the different results I got. I'm not sure what accounts for the differences from Ramsay's (and Sara Steger's) results; I'll try to show you what I mean below. In a future post I'll use the same "method" on aa different text (spoiler: it's *Ulysses*).

Readers familiar with *The Waves*, and the demands of text processing, will immediately recognize why the analysis of the characters' monologues would present itself as a tractable problem ("indeed an easy matter"). While, in theory, one could do a similar analysis for any novel (or any work with multiple speakers), the narrative structure of *The Waves* makes it particularly available to this sort of analysis. Chapters describing the process of the sun across the sky in the course of a single day alternate with chapters in which characters speak in semi-monologue about their lives. This device itself is the novel's most obvious departure from the conventions of narrative fiction, but it also makes it "an easy matter" (well, maybe for some people) to extract these dialogues. If you had good, marked-up data, you could easily extract this information (as Lincoln Mullen shows in this post, working with the Folger's TEI Shakespeare); but if all you have is unstructured plaintext, you're going to have a problem. Woolf's novel though, even in plaintext, carries a good deal of this informational structure in its novelistic form (there is, as they say, no such thing as an unmarked text).

Here is a chunk of *The Waves*, quoted at random:

> 'Where is Bernard?' said Neville. 'He has my knife. We were in the tool-shed making boats, and Susan came past the door... Now we must drop our toys. Now we must go in together. The copy-books are laid out side by side on the green baize table.'
>
> 'I will not conjugate the verb,' said Louis, 'until Bernard

has said it…

There is always a short phrase (starting with an opening single quotation mark—i.e. an apostrophe—and a capital letter), some text, a closing single quote (variously punctuated), the word `said` followed by a character name and some punctuation mark, an opening single quotation mark and some words. This single "monologue" may continue into the next paragraph (which would then, consistent with convention, be opened by a single quotation mark—i.e. an apostrophe). Finally the monologue is closed by an apostrophe before the narrative turns to another character (and another opening apostro-quote), or to one of those sun-dappled interludes.

Whew; describing what is so obvious to any reader of the text is painful (as, I imagine, is reading my description of it), but it is this highly structured convention which makes Woolf's novel comparatively available to processing. Even absent TEI (or other) markup, Woolf's convention creates an ad-hoc ordered hierarchy of content objects, at least for the reader interested in the characters' monologues. Someone with more regex-fu than I have might be able to cut out character dialogue programmaticly.○I think the regex would go something like `/'([\^,]+,)' said Louis, '(*)'/` and would capture, as $1 and $2 the material the character says… in theory. I tried to sort this out, but quickly gave up. Instead, I manually paged through the text and pasted together all the text said by a single character, so that from that opening apostro-quote to the closing apstro-quote would all be one one line, and the phrase `[character name] said` would occur somewhere near the front of that line.○In emacs, checking twitter and listening to podcasts, this represented an hour and a half's labor; labor, mind you, which was sufficiently mindless that I enjoyed a beer. Though, as you'll see below, *this fact* led me to redo the entire thing.

And so, with thanks to Woolf for her highly structured departure from novelistic convention, and to emacs for keybindings that made this somewhat less loathsome, you're ready to extract your data. It is now simply a matter of grepping the file for each character:

```
grep 'said Louis' the-waves.txt > characters/louis.txt`
grep 'said Neville' the-waves.txt > characters/neville.txt`
...
```

And so on.○The text Ramsay & Steger use, and that I also used, comes from Project Gutenberg Australia. Because of copyright, I cannot share the processed data I am working with—and, as you'll see, this extraction process is a crucial step. If you'd be interested in seeing or using this data, to save yourself the hour and a half's labor, however, just drop me an email. I have relatives in Australia who would be happy to host you during the term of your interaction with this copyrighted material.

At which point, the actual, real analysis begins. Here is the code I used, in `R` (using the `tm` package), to get my results. It assumes that that each individual's speech is contained in a single text file in a directory immediately below the working directory, called 'characters' (that's what all that grepping above was about).

```
# This code relies on the tm (text mining) package
library('tm')

# Create a corpus based on the subdirectory
characters <- Corpus(DirSource('characters/'))

# To aid processing lets make everything lower-case
characters <- tm_map(characters,tolower)

# And remove punctuation
characters <- tm_map(characters,removePunctuation)

# And we'll remove stopwords - this step, is optional. But
# of the code I'm pasting here I removed them, in an effort
# alas!) to match Ramsay & Steger's result.
```

```
characters <- tm_map(characters,removeWords, stopwords('eng

# Now, we create a Document Term Matrix - that is, a set of
# the frequencies for each word in each document. The secre
# sauce is that control=list(weighting=weightTfIdf) line, w
# asks that those not be raw counts, but tfidf scores.
dtm <- DocumentTermMatrix(characters,   control=list(weight
```

And here is just a taste of what that looks like. (This code asks R○Personify much? to let me see the 45th through 55th terms in the matrix (the terms are arranged alphabetically) for all texts.○You access the matrix by requesting row and column: `matrix[row, column]`; so the empty row field requests all rows (that is, all texts), and columns (which represent words) 45 through 55 (a range chosen entirely at random).

```
>Inspect(dtm[,45:55])
A document-term matrix (6 documents, 11 terms)

Non-/sparse entries: 16/50
Sparsity           : 76%
Maximal term length: 13
Weighting          : term frequency - inverse document freq

             Terms
Docs               account     accounts   accretions accumu
  bernard.txt 0.0002033962 0.0002033962 0.0001247118 0.0002
  jinny.txt   0.0000000000 0.0000000000 0.0000000000 0.0000
  louis.txt   0.0000000000 0.0000000000 0.0000000000 0.0000
  neville.txt 0.0000000000 0.0000000000 0.0004414937 0.0000
  rhoda.txt   0.0000000000 0.0000000000 0.0000000000 0.0000
  susan.txt   0.0000000000 0.0000000000 0.0000000000 0.0000
             Terms
Docs          accumulations     accuracy   accurately
  bernard.txt  0.0002033962 0.0002033962 0.0000000000 0.000
  jinny.txt    0.0000000000 0.0000000000 0.0000000000 0.000
```

```
  louis.txt    0.0000000000 0.0000000000 0.0000000000 0.000
  neville.txt  0.0000000000 0.0000000000 0.0000000000 0.000
  rhoda.txt    0.0000000000 0.0000000000 0.0007776662 0.000
  susan.txt    0.0000000000 0.0000000000 0.0000000000 0.000
           Terms
Docs            acknowledge
  bernard.txt 0.0000786844
  jinny.txt   0.0000000000
  louis.txt   0.0002762431
  neville.txt 0.0002785515
  rhoda.txt   0.0000000000
  susan.txt   0.0000000000
```

So, this shows us, for instance, that Bernard, Louis, and Neville, all use the word acknowledge (Jinny, Rhoda, and Susan don't); and Louis and Neville use it more than Bernard (but at the exact rate as each other).

At this point, we've got the data. All that's needed is a little `R` data-finesse to get it back out in the order we want it. I'm quite new to `R`, so I may be missing the better/more obvious way to do this, but this way seems to work. I load the data into a matrix, and then extract it into lists (I think I'm getting my `R` data types right), ordered by the word's score. We can then output as many (score, term) pairs from the re-ordered lists that we want (say, the top 24 terms).

```
m           <- as.matrix(dtm)

bernard     <- sort(m[1,], decreasing=TRUE)
jinny       <- sort(m[2,], decreasing=TRUE)
louis       <- sort(m[3,], decreasing=TRUE)
neville     <- sort(m[4,], decreasing=TRUE)
rhoda       <- sort(m[5,], decreasing=TRUE)
susan       <- sort(m[6,], decreasing=TRUE)
```

```
>louis[1:24]
    western      accent     grained        thou        wilt
0.006426702 0.005691854 0.004284468 0.004284468 0.004284468
   boasting        nile     average      clerks         oak
0.003502680 0.003502680 0.002856312 0.002856312 0.002762431
 australian        boys    pitchers       steel      beater
0.002627010 0.002209945 0.002189175 0.002189175 0.002142234
    custard  eatingshop     england       eyres  fourthirty
0.002142234 0.002142234 0.002142234 0.002142234 0.002142234

> bernard[1:24]
  thats     hampton        lady   curiosity      letter
0.002237358 0.001870677 0.001870677 0.001830566 0.001830566
    elderly      heaven     married    observed       byron
0.001627170 0.001627170 0.001627170 0.001627170 0.001621254
     dinner      willow      phrase         fin      simple
0.001496542 0.001496542 0.001495004 0.001423774 0.001423774
       self       stick       sense      nature    thinking
0.001371830 0.001371830 0.001288768 0.001247118 0.001247118

> neville[1:24]
      story        ones      doomed immitigable      papers
0.003342618 0.003090456 0.002880181 0.002880181 0.002880181
 perfection       camel      detect   hosepipes      hubbub
0.002207469 0.002160136 0.002160136 0.002160136 0.002160136
     mallet      marvel   squirting        boys       byron
0.002160136 0.002160136 0.002160136 0.001949861 0.001765975
      scene shakespeare       stair      abject   admirable
0.001765975 0.001765975 0.001671309 0.001440091 0.001440091

> jinny[1:24]
     tunnel    prepared   billowing        game      native
0.003833041 0.003194201 0.003125710 0.003125710 0.003125710
    quicker  melancholy      bodies        band       bodys
0.003125710 0.002555361 0.002121992 0.002083807 0.002083807
```

```
      coach        crag      dazzle     deftly    equipped
0.002083807 0.002083807 0.002083807 0.002083807 0.00208380
     felled     glasses        jump    lockets    matthews
0.002083807 0.002083807 0.002083807 0.002083807 0.00208380

> rhoda[1:24]
    oblong        dips       tiger      fuller      themoh
0.005443664 0.003888331 0.003337767 0.003110665 0.00311066
     fallen   suspended      cliffs     garland  manybacked
0.002707581 0.002384119 0.002332999 0.002332999 0.00233299
       pond   structure      terror       bunch        foam
0.002332999 0.002332999 0.002105897 0.001907295 0.00190729
      party      puddle       dream       pools     violets
0.001907295 0.001907295 0.001805054 0.001805054 0.00180505

> susan[1:24]
    kitchen      setter     washing  windowpane        bury
0.006213103 0.004053254 0.004053254 0.004053254 0.00313602
       gate      horses       apron     seasons    squirrel
0.003106551 0.003106551 0.003039940 0.003039940 0.00303994
     butter       clean         wet      winter        baby
0.002485241 0.002485241 0.002485241 0.002063764 0.00202662
   cabbages    carbolic       clara      cradle        eggs
0.002026627 0.002026627 0.002026627 0.002026627 0.00202662
```

My data doesn't quite match Ramsay & Steger's (qtd. in Ramsay 13); look at the Louis data to see what I mean (I've reordered the terms alphabetically so that you can see the similarities and differences more easily):

Louis

Ramsay & Steger

Me

accent

accent

attempt

australian

australian

average

average

beast

beast

beaten

beaten

boasting

bobbing

bobbing

boys

clerks

clerks

custard

custard

discord

disorder

eating-shop

eatingshop

england

england

eyres

eyres

four-thirty

fourthirty

grained

grained

ham

ham

mr

nile

nile

oak

pitchers

pitchers

stamps

steel

steel

thou

thou

western

western

wilt

wilt

The terms `fourthirty` and `eatingshop` are victims here of the way `R` removed punctuation. `R` can also explain one other of the differences: Ramsay's list has the word `mr`, which my list lacks. `mr` is on the list of stopwords I removed from the text. But the others? I don't have any explanation for those. Ramsay's list has these words, which my list lacks (in addition to `mr`): *attempt*, *discord*, and *disorder*. And my list has *oak*, *stamp*, *boys*, and *boasting*, which his lacks.

Well, so, okay; but pretty good, right? Well, maybe not. It only gets worse for the other characters. Here is a summary of the discrepancies for the other characters:

Bernard (4 Shared)

Here my list and Ramsay & Steger's are *very* different.

The lists share only four terms: *letter*, *curiosity*, *simple*, and *canopy*.

Ramsay & Steger's then has: *arrive, bandaged, bowled, brushed, buzzing, complex, concrete, deeply, detachment, final, getting, hoot, hums, important, low, moffat, rabbit, thinks, tick, tooth*

important would be removed by my stoplist… the rest though should otherwise be in my list.

But mine has: *thats, hampton, lady, ones, elderly, heaven, married, observed, byron, phrases, dinner, willow, phrase, fin, describe, self, stick, sense, nature, thinking.*

Let's look at some of the words and try to sort this out; *hoot* seems a pretty unique word. Going back through the text, I find seven instances of *hoot* or *hoots*. They breakdown this way by character:

- **Louis**: 'a siren hoots'

- **Bernard**: 'But now list; tick, tick; hoot, hoot; the world has hailed us back to it... Then tick, tick (the clock); then hoot, hoot (the cars)'; 'a siren hoots'
- **Rhoda**: 'the steamer hoots'

Well, *hoot* seems unique to Bernard. Okay, let me jump back into R.

```
>inspect(dtm[,c('hoot')])
A document-term matrix (6 documents, 1 terms)

Non-/sparse entries: 1/5
Sparsity           : 83%
Maximal term length: 4
Weighting          : term frequency - inverse document freq

             Terms
Docs                   hoot
  bernard.txt 0.0008135849
  jinny.txt   0.0000000000
  louis.txt   0.0000000000
  neville.txt 0.0000000000
  rhoda.txt   0.0000000000
  susan.txt   0.0000000000
```

Just so that we aren't confused, lets grab the raw counts (rather than the *tfidf* scores).

```
> raw <- DocumentTermMatrix(characters)
> inspect(raw[,c('hoot')])
A document-term matrix (6 documents, 1 terms)

Non-/sparse entries: 1/5
Sparsity           : 83%
Maximal term length: 4
Weighting          : term frequency (tf)
```

```
              Terms
Docs           hoot
  bernard.txt    4
  jinny.txt      0
  louis.txt      0
  neville.txt    0
  rhoda.txt      0
  susan.txt      0
```

Well, that's no help then; *hoot* is unique to Bernard. At this point I begin to suspect something unpleasant. Maybe in my manual data munging, I bollocks'd something. Obviously, It seems like I got the occurrences of *hoots* in there, attributed to the right person (though maybe I deleted some other *hoots*?); but if I deleted something, or double pasted something, that could change the complexion of corpus as a whole, and so dilute the score (or inflate the score of some of these other terms showing up in my list).

So, at this point I went back and reprocessed the file again to insure I didn't break anything. I used this bit of elisp (courtesy of this) to remove (I included it in a macro for a first pass) hard newlines within a paragraph:

```
(defun remove-line-breaks ()
  "Remove line endings in a paragraph."
  (interactive)
  (let ((fill-column (point-max)))
    (fill-paragraph nil)))
```

And I ran it again. My scores shifted ever so slightly, but my top terms for Bernard remained the same.

Back in R, let's compare my lowest rank term with *hoot* again:

```
>inspect(dtm[,c('canopy','hoot')])
A document-term matrix (6 documents, 2 terms)

Non-/sparse entries: 2/10
```

```
Sparsity           : 83%
Maximal term length: 6
Weighting          : term frequency - inverse document frec

             Terms
Docs                canopy         hoot
  bernard.txt 0.001219801 0.0008132009
  jinny.txt   0.000000000 0.0000000000
  louis.txt   0.000000000 0.0000000000
  neville.txt 0.000000000 0.0000000000
  rhoda.txt   0.000000000 0.0000000000
  susan.txt   0.000000000 0.0000000000
> inspect(raw[,c('canopy','hoot')])
A document-term matrix (6 documents, 2 terms)

Non-/sparse entries: 2/10
Sparsity           : 83%
Maximal term length: 6
Weighting          : term frequency (tf)

             Terms
Docs          canopy hoot
  bernard.txt      6    4
  jinny.txt        0    0
  louis.txt        0    0
  neville.txt      0    0
  rhoda.txt        0    0
  susan.txt        0    0
```

That is to say, *canopy*, based on my raw scores, does look more distinctive than *hoot*. What about *moffat* (from *Mrs Moffat* in the text○So, if `mr` showed up in their analysis, why not `mrs` here? Because other characters talk about other Mrses—Mrs Crane, Mrs Constable.) which ranks high on Ramsay & Steger's list, but not at all on mine.

```
inspect(dtm[,c('moffat','canopy')])
A document-term matrix (6 documents, 2 terms)

Non-/sparse entries: 2/10
Sparsity           : 83%
Maximal term length: 6
Weighting          : term frequency - inverse document frec

             Terms
Docs               moffat      canopy
  bernard.txt 0.001219801 0.001219801
  jinny.txt   0.000000000 0.000000000
  louis.txt   0.000000000 0.000000000
  neville.txt 0.000000000 0.000000000
  rhoda.txt   0.000000000 0.000000000
  susan.txt   0.000000000 0.000000000
> inspect(raw[,c('moffat','canopy')])
A document-term matrix (6 documents, 2 terms)

Non-/sparse entries: 2/10
Sparsity           : 83%
Maximal term length: 6
Weighting          : term frequency (tf)

             Terms
Docs          moffat canopy
  bernard.txt      6      6
  jinny.txt        0      0
  louis.txt        0      0
  neville.txt      0      0
  rhoda.txt        0      0
  susan.txt        0      0
```

So `moffat`'s score is just the same as `canopy` (but there are a lot of terms with that score, and terms with the same score are then ranked alphabetically, so it gets pushed off our top 24 listRamsay & Steger's scores are likewise ranked alphabetically when they have equal scores; have a look at those lists on page 13, and you'll see islands of alphabetical ordering.).

So, let me jump back to my initial, raw file; I check there, and *Moffat* indeed occurs 6 times.

So what on earth is going on here? At this point, I don't know. Here are, I think, the possibilities. The fact that the greatest discrepancy comes from the character with the most monologue data is perhaps meaningful, but *how* it's meaningful is not obvious. So:

- Perhaps, despite two attempts to get this data all set, I bollocks'd something that is upsetting the scores.
- Perhaps Ramsay & Steger stemmed their data; I haven't got the stemmer working in `R` properly yet, so that could account for a difference (but terms like *grained* and *bobbing* appear on their list, and don't appear to have been stemmed).
- Could the interlude chapters be upsetting things? I discard them from my analysis entirely. If they were included they might change the overall complexion.
- While I am very wary of suggesting Ramsay & Steger's data is wrong, I will note that *if* they tried to manipulate this data using regex, the data itself isn't consistent. There are cases where paragraphs are missing opening apostro-quote marks (four of them by my count) and a paragraph missing a closing apostro-quote. Depending on how you built your regex, these could throw things off and produce the dilution effect I am worried about.

After tinkering for a bit, I suspected that this might be so. But looking at the raw counts for my data makes me doubt that. One thing you might suspect, if carving up the text into characters' monologues were the problem, would be that some key term might be misattributed; but, for instance, my raw counts of *catullus* seem consistent with Ramsay & Steger's results:

```
> inspect(dtm[,c('story','catullus')])
A document-term matrix (6 documents, 2 terms)

Non-/sparse entries: 5/7
Sparsity           : 58%
Maximal term length: 8
Weighting          : term frequency - inverse document frec

             Terms
Docs                  story    catullus
  bernard.txt 0.0011797090 0.000124653
  jinny.txt   0.0000000000 0.000000000
  louis.txt   0.0002763958 0.000000000
  neville.txt 0.0031438302 0.002076189
  rhoda.txt   0.0000000000 0.000000000
  susan.txt   0.0000000000 0.000000000
> inspect(raw[,c('story','catullus')])
A document-term matrix (6 documents, 2 terms)

Non-/sparse entries: 5/7
Sparsity           : 58%
Maximal term length: 8
Weighting          : term frequency (tf)

             Terms
Docs          story catullus
  bernard.txt    15        1
  jinny.txt       0        0
  louis.txt       1        0
  neville.txt    12        5
  rhoda.txt       0        0
  susan.txt       0        0
```

That is, Ramsay & Steger think Catullus is distinctive for Neville.

And, indeed, it appears to be so. The difference between my results and theirs the *tfidf* score—that is, in *how* distinctive it is. If their corpus were differently constructed than mine in some way, it might affect *how* distinctive it is.

So, there may be a data carving problem; who miscarved though is not obvious from this data, I don't think. It is also possible that there may be some algorithmic difference; I am using the *tf-idf* algorithm built into R as a sort of black box. My scores are very different from the one's Ramsay & Steger share on pg. 12. So we're definitely doing *something* different. And that might account for these differences. What clear I need to do is return to algorithm to better understand what's going on here.

For now, though, I don't know. I'll here just summarize the data for the rest of the characters.These are using the reprocessed data, so they may be a little different from above; there were no differences in top terms for Louis or Bernard, and these scores were extracted using exactly the same code as above.

Neville (12 Shared)

- **Shared**: doomed, immitigable, papers, camel, detect, hubbub, loads, mallet, marvel, abject, admirable, ajax
- **Just Ramsay & Steger**: catullus, bookcase, bored, expose, incredible, lack, shoots, squirting, waits, stair, aloud
- **Just mine**: boys, byron, cheep, founder, hosepipes, ones, perfection, scene, shakespeare, story

Jinny (20 Shared)

- **Shared**: tunnel, prepared, melancholy, billowing, game, native, peers, quicker, band, cabinet, coach, crag, dazzle, deftly, equipped, eyebrows, felled, jump, lockets
- **Just Ramsay & Steger**: fiery, victory, banners, frightened, gaze

- **Just mine**: bodies, bodys, matthews, murmured, prepare

Rhoda (13 Shared)

- **Shared**: oblong, dips, bunch, fuller, party, cliffs, manybacked, minnows, pond, structure, tiger, swallow, bow
- **Just Ramsay & Steger**: moonlight, them— (**this result indicates that we're handling puncutation differently**…), allowed, empress, fleet, garland, immune, wonder, africa, amorous, attitude
- **Just mine**: caverns, chirp, choke, column, fallen, foam, pools, puddle, suspended, terror, violets

Susan (16 Shared)

- **Shared**: setter, washing, apron, squirrel, windowpane, kitchen, baby, bitten, boil, cabbages, carbolic, clara, cradle, eggs, ernest, seasons
- **Just Ramsay & Steger**: cow, pear, betty, hams, hare, lettuce, locked, maids
- **Just mine**: beds, bury, butter, cart, clean, gate, wet, winter

Oh, and here is the breakdown of the amount of text I have for each character:

```
wc *.txt
      46   32608  182921 bernard.txt
      33    6331   34467 jinny.txt
      46    8905   49588 louis.txt
      39   10011   55543 neville.txt
      40    8147   44839 rhoda.txt
      34    6131   33023 susan.txt
     238   72133  400381 total
```

Bernard has the most,○Of course, because the final chapter is offered entirely in his voice. followed by Neville, Louis, Rhoda, Jinny, and Susan.

Works Cited

Ramsay, Stephen. *Reading Machines.* Urbana: U of Illinois P, 2011. Print.