Text Analytics 101

I have, for the past several years now, introduced my undergraduate students to some elements of textual analysis using computational methods. I use *text analytics* here only tentatively: many readers will perhaps be more familiar with, and indeed prefer, the older term text mining, but for me that term is close to data mining, which usually refers to working with a great deal more texts than I am going to discuss here — and that is why I suppose the term *data mining* has morphed into *big data*. (More on this anon.)

The number of texts here is one. That's right one text, and, in fact the one text is a short story. Keeping the text small is one way I have found to keep any psychological barriers to entry low when I introduce students to text analytics. The particular short story I have chosen in the past, and which I use here, is Richard Connell's "The Most Dangerous Game." The 1924 story has two advantages when working with students: first, it is in the public domain, and, second, the text's story has been so widely adapted that students are already familiar with it and have probably seem an adaptation of the story in some fashion within their own recent memory. E.g., only a year or so ago, the FX Network's adult cartoon series, *Archer*, featured a version of the story entitled, "El Contador" (The Accountant).

If we are to use a computer to make possible certain kinds of analysis of texts, what are the kinds of things we might like to know?

1. First, we want to know the overall length of the story, and we want to know some basic information like how many words, sentences, and paragraphs are used to tell the story, or make the argument in the case of essays. (This kind of information is useful for later mapping our the overall shape and structure of a text.)
2. Second, we want to know how many of the words in the word count are actually unique, what those words are, and which words get used the most often and which the least. (This establishes the vocabulary used, points to any particular registers, and begins to reveal the interaction between words and meaning.)
3. Third, using the word frequency distribution, the fancier term for counting individual words, we want to visualize the text both as a graph and as a word cloud. In doing so, we can begin to "see" for ourselves which words matter and which words don't matter. (This introduces the idea of *function words* in the form of a word stop list.)
4. Fourth, we want to use our new-found insight into word usage to examine particular instances: we want to see words in context and we want to see what words mean within the context of a particular text. (This highlights the role of context and offers a companion to meaning to be found simply in the words used.)

With that list in mind, I would like to introduce the Useful Python Scripts for Texts repository. The repository contains the text of the story as well as the scripts that will produce the results outlined above. Please note that, at this stage, the scripts are designed to be run with the target text inside the same folder (directory) as they are. If you want to use a different text, simply copy and paste it into the folder and change the filename in the script. The ReadMe file explains how to save the output of the scripts to a file, which will come in handy for Step 3

above.

For those readers already familiar with Python, and by familiar I mean you already have it installed and know how to access it, you can skip the next bit. For everyone else, a bit of review won't hurt. Some people are going to want to know why I am doing all this in Python and not using off-the-shelf solutions. This is not the moment to engage in a recapitulation of all the usual arguments in favor of open source software, how it not only parallels the academy's ideals but how it practically makes possible the spread of ideas in a world sometimes hostile to such spread. What matters here is:

- Python is free, widely-available, and has a large community to help anyone interested in using it.
- Python runs on all the three major operating systems: Windows, Linux, and Mac.
- The Python scripts used here are similarly free and available for users to do with them what they want.

The scripts are, in fact, placed them in the public domain under the Creative Commons license for doing so. The links for the scripts take users to a GitHub page from which they can be downloaded, or, if you have a GitHub account yourself, you can fork the repo itself. Please feel free to do either.

Finally, please note that a basic working installation of Python will let you perform Steps 1-3 above. If you are interested in looking at words in context and in examining other kinds of relationships between words, Step 4, then you are going to need to have the Natural Language Toolkit installed. It's not difficult to do so. If you are using a Mac, I posted instructions last year. Instructions for getting Python installed on a Windows PC are available, with further instructions for the NLTK also available. I assume everyone else is running Linux or BSD and, really, you don't need my help. (Please note that there are a variety of suggested ways of getting the NLTK installed on a Mac, but the MacPorts route is really the way to go. Trust me: I've gone some of the other ways.)

Questions Raised by Counting

Now we can start working with an actual text and looking at some actual numbers. Every time I do this with students I find it helpful to have a conversation about how these numbers won't in and of themselves tell us much about the text, but the various features they reveal or the questions they lead us to ask are useful. These numbers can't draw conclusions, that's the job of the human analyst, but they can provoke inquiry. And, sometimes, they reveal dimensions of a text that maybe we would not have thought about without seeing it quantified.

With that noted, let's plunge into some rough stats for "the Most Dangerous Game" which we get, simply enough by runing `stats.py`:

```
python stats.py
```

And it prints out the following, which we can copy and paste anywhere:

```
COUNTS
Paragraphs      : 205
Section Breaks: 0
Sentences  : 717
```

```
Words      : 7959

AVERAGES
Sentences per paragraph: 3
Words per paragraph: 38
```

So an 8000 word story told in two hundred paragraphs and seven hundred sentences. (According to Lee Masterson, this puts MDG in the territory of the novelette, which seems odd to me or an index of how things have changed. If you want to know the other counts, see this Askville Answer.)

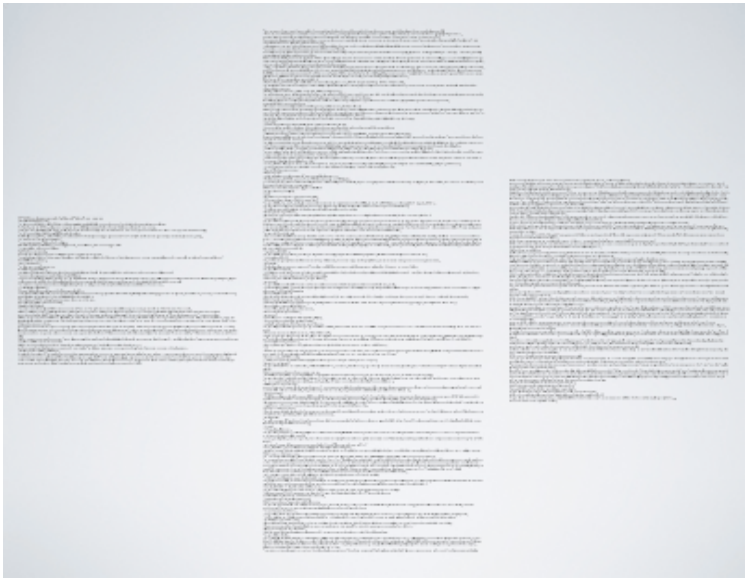
The other *counts*, as I called them in this script — feel free to change that, are for paragraphs and sentences. These numbers in and of themselves aren't terribly interesting, until you play with them a bit, as I did to get the two averages: neither of which is something we typically discuss when examining texts — and probably the only reason most of us, and especially our students, are familiar with word counts is because we have had to deal with either minimums or maximums. We rarely think of them as having any kind of significant descriptive power. And yet when we combine some of these counts we end up with some interesting averages.

The first one, sentences per paragraph, is striking. *Three?* That seems like a terribly small number, which drives most readers to look at the story more closely. What they discover, as they skim through the pages of the PDF version of the story is that a great deal of the story is told in dialogue. There is so much dialogue that you have to scour the story for the moments of non-talking action. There are, in fact, two passages of extended narration of action: the first occurs when the ship on which we first meet Rainsford sinks and he has to make his way to the island and the second is the famous game itself.

These moments of action help to delimit the principle sections of the text:

1. At sea and Rainsford's arrival on the island
2. At the chateau and the dialogue between Rainsford and Zaroff which reveals the nature of the game
3. The hunt itself and the story's conclusion

A fun thing to do is to copy and paste the text of these three sections into an image so students can "see" the story in its entirety:



All of MDG in very, very small text.

For those familiar with the Hollywood Formula, the story meets the idealized ratio of 1-2-1 of content pretty closely. Closer to our topic at hand, the quick visualization also lets us see that the first and second sections have a lot of thin lines, representing paragraphs that are made up mostly of dialogue, and that the third section has some fatter lines. If we take our new insight into the text and do a little counting of paragraphs and words in these sections, we get the following results:

- At Sea: 1029 words / 37 paragraphs = 28 words per paragraph
- At the Chateau: 4407 words / 129 paragraphs = 34 words per paragraph
- The Hunt: 2367 words / 38 paragraphs = 62 words per paragraph

Words per paragraph is an odd measure, but one that reveals that the "action" parts of the story actually takes place in longer paragraphs.

The Words of the Story

Let's find a bit more about the words themselves by running the next script, `words.py`. This script produces quite a bit of output, and so my best advice is to capture it by entering the following at the command line:

```
python words.py > mdgwords.txt
```

This tells the command shell to send the output of the script to the file `mdgwords.txt`. You can name the file anything you want. Or you don't need to direct the output to a file: you could just watch the output fly by and then copy and paste it into a file. I am asking you to handle output like this, instead of writing it to a file for you because I am working on making it possible for you to work with texts of your own choosing without editing the script. (I'm getting there, I'm getting there.)

At the top of the file you'll see a line of redundant information and a line of new information:

```
Words in text: 7959
Unique words: 1987
```

This is some new data. For fun, I like to give students the task of figuring out the mean for the words in a story: here it's obviously something like four occurrences per word. (4.0055 to be exact.) But that's obviously not what really happens. Below a dashed line in the text, students will see a list of words that begins like this:

```
Sorted by highest frequency first:
the,505
he,248
a,248
of,172
and,162
i,154
to,148
was,140
his,137
rainsford,117
```

The occurs five hundred times in this short story? (It's every sixteenth word by my count.) At this point, I like to talk about the list in its raw form above or I will ask students to trim off the first few lines of the file above and save the document as a comma-separated value file (`.csv`). Once the file consists of only the *word, number* pairing seen above, it can be imported into Excel, where it can be easily turned into a bar chart. The first eight words in the list dominate any visualization — the same thing can be seen in a word cloud when no common words are dropped (or stopped). (A built-in word cloud script is in the works, but is not currently available.)

It may sound strange, but I have found it very effective to work my way through the list of words with a class with an Excel spread sheet projected at the front of the room: we highlight the words we think are interesting. Regularly what we find is that we have to scroll past the first screen of words before the first words that seem interesting us, based on our reading of the story, begin to turn up:

32	no	30
33	they	30
34	one	29
35	hunting	28
36	be	27
37	then	27
38	at	26
39	were	26
40	out	26
41	so	25
42	up	25
43	when	25
44	by	24
45	night	22
46	now	22
47	if	22
48	down	22
49	its	21
50	eyes	21
51	could	21

The first interesting words begin to show up

We have to go even further down before we begin to see a large percentage of words that seem significant. (Please note that the current version of the script sorts words first by their frequency and then alphabetically.)

69	man	16
70	hunt	16
71	into	15
72	saw	15
73	sea	15
74	can	15
75	them	15
76	sound	15
77	thought	15
78	must	15
79	island	14
80	good	14
81	again	14
82	ivan	14
83	off	13
84	place	13
85	black	13
86	we	13
87	come	13
88	think	13
89	been	13
90	your	13
91	tree	13
92	why	12
93	like	12
94	face	12

The Meaningful Middle

This interesting middle range of words continues for a while until words begin to drop in usage: I regularly find that three occurrences is about the threshold for short stories for most readers.

359	opened	3
360	sign	3
361	wounded	3
362	finished	3
363	stone	3
364	tall	3
365	knocker	3
366	above	3
367	within	3
368	straight	3
369	heart	3
370	fell	3
371	sanger	3
372	dressed	3
373	gray	3
374	mans	3
375	slight	3
376	pleasure	3
377	explained	3
378	russian	3
379	showed	3
380	please	3
381	wait	3
382	below	3
383	hall	3
384	heads	3

And finally things taper off…

Again, all of this simply prompts readers to ask more and better questions. As someone for whom the language of a text is terribly important, I find it terribly ironic that it's numbers that makes that point best with some readers.

Words in Context

With this list of words in hand, it's time to re-engage the text. I find it useful to assign students each a word with a high-value frequency, a middle value, and a low value. In the past, I have asked them simply to use the *Find* feature in their PDF viewer, but, if you have installed the NLTK, you can try out the next script, `concordance.py`.

`concordance.py` relies on several functions available through the NLTK that make it possible to work with texts. If you open the script, you can see for yourself, but if you simply run it, you will see:

```
% python concordance.py
Enter the word you would like to see in context:
```

If you enter the word *hunter*, the program will print out:

```
Building index...
Displaying 10 of 10 matches:
ld , " agreed Rainsford . " For the hunter , " amended Whitney. " Not for
ainsford. " You ? ? ? re a big-game hunter , not a philosopher. Who cares
been a fairly large animal too. The hunter had his nerve with him to tac
st three shots I heard was when the hunter flushed his quarry and wounded
. Sanger Rainsford , the celebrated hunter , to my home. " Automatically
. " They were no match at all for a hunter with his wits about him , and
nsford. " " Thank you , I ? ? ? m a hunter , not a murderer. " " Dear me
ling of security. Even so zealous a hunter as General Zaroff could not
 ? ? a small automatic pistol . The hunter shook his head several times ,
a spring. But the sharp eyes of the hunter stopped before they reached the
```

Such an output offers a very quick and easy way to see all the uses of hunter piled on top of each other. The first two line in our results beg another question, though, since they turn up the use of the word hunter in the dialogue between Rainsford and his first interlocutor, Whitney, which contains in it the line that foreshadows much of the rest of the story: *The world is made up of two classes—-the hunters and the huntees.* That's an important line, but it doesn't show up in our search for *hunter* because from a computer's point of view *hunter* and *hunters* are not the same word, or *token* to use the more precise term from linguistics. (Is there a way to see *hunter*, *hunters*, and *huntee* all in the same list? There is, but it involves a bit more scripting than will reward us at this moment. You can see the draft script in the UPST directory: `concordstem.py`.)

Next Steps

In the next iteration of Text Analytics, and the *Useful Python Scripts for Texts*, we will take a look at things like collocations, bigrams, synonyms within a text, and other relations.

Thanks for reading. I hope this helps convince you how easy this kind of analysis is and, at the same time, how rewarding it can be. I have found all of these techniques especially powerful when working with undergraduates. Word clouds have become extremely popular, and they are a powerful visualization tool, but they really should represent the middle or end of an analytical process, after analysts have given some thought to the particular words involved.

Related

Data Mining vs Big Data
2013 February 20
In "note"

Middling Data
2013 September 21
In "note"

Python Text Processing for Freshmen
2012 December 21
In "note"

Posted on **2013 February 21** *by* **johnlaudun**. *This entry was posted in* **note** *and tagged* **texts**, **textualanalysis**. *Bookmark the* **permalink**. **Edit**

5 thoughts on "Text Analytics 101"

Gunjan Amit

— 2013 February 24 at 22:40 — (Edit)

Very Informative Article. Thanks!

Reply

Pingt Scotland's Collections and the Digital Humanities (Edit)

Libby Hemphill

— 2014 October 15 at 14:16 — (Edit)

Thank you! This is really helpful for my intro class in digital humanities research. I'm making some modifications to the scripts (esp. allowing user to specify file to use in one config file). OK to send pull requests? https://github.com/libbyh/upst

Reply

johnlaudun

— 2014 October 16 at 08:42 — (Edit)

Oh, certainly. That way I can keep up with all the cool stuff you'll add. And, my apologies for this repo

languishing as it has. I've been finishing up work on a book on a non-computational topic, but I plan to get back to it in the next few months, so your pull/fork could not have come at a better time.

Reply

Ping! Introducing Text Analytics to Undergraduates | Libby Hemphill (Edit)

Leave a Reply

Enter your comment here...