Adverbly Adjectives

on Feb 26

Last updated by Vasundhara G.

Contents

Project Overview

We are interested in a classification of the [*adverb*-ly *adjective*]

construction and a study of its use:

- Across different genres
- In opinionated text
- Over time

To answer these questions we use corpus-based methods, counting instances and the distribution of the construction in several corpora: COCA, CORE, SOCC, Movie$, Cornell and NYT reviews. See the data section for a description of these corpora

Data - What and Where

- COCA

The Corpus of Contemporary American English, is a large corpus (>500 million words) equally divided among spoken, fiction, popular magazines, newspapers, and academic texts, collected from 1990 till present.

COCA is available at corpus.byu.edu/coca/ but this website has a terrible interface. The corpus is also available for purchase and we have bought a copy but unfortunately due to copyright limitations, every 200 words, 10 words are replaced with '@' symbols. Since the website is so awful to use, we use our downloaded copy of it. Due to this, and due to its large size, we handle it a little differently from the other corpora (see the section Notes on COCA)

The COCA data is located in *Vault -> Discourse Lab -> Data -> COCA*. The raw texts are available in the folder *Texts*. Note that the older data is divided by genre into 5 zip files, but the data from 2012 to 2015 (all genres) is in a separate zip file. The POS data is also available in the folder *POS*.

- CORE

The Corpus of Online Registers of English is a corpus of >50 million words from the web, collected by Biber et al, and organized into several fine-grained registers. The detailed classification into subgenres and then into genres is described in their paper, available on Vault (*Vault -> Discourse Lab -> Data -> CORE -> asi.23308-1.pdf*)

One problem that arose was that some texts are classified as "hybrids", e.g., equally belonging to the Informational Description and Opinion subgenres. So we chose to disregard these hybrid texts and only take into account the texts that were well classified into one subgenre. The subgenres were too fine-grained for our purposes but the genres worked well so we mapped each subgenre to the appropriate genre for our counts and lists.

The CORE data is located in *Vault -> Discourse Lab -> Data -> CORE*. The raw texts are available in the folder *CORE registers_untagged*. The mapping of text to subgenre is in *core_texts.xlsx*, and the POS data is available in the folder *pos*.

- SOCC

The SFU Opinion and Comments Corpus is a corpus of opinion articles from The Globe and Mail and comments on these articles online. For this project we are only interested in the articles, not the comments. There are no genres in this corpus.

The SOCC data is located in *Vault -> Discourse Lab -> Data -> Globe_and_Mail*. It can alternatively also be downloaded following the instructions here: github.com/sfu-discourse-lab/SOCC. The raw texts are in *CSVs -> gnm_articles.csv*. The POS data is available in the folder *pos*.

- Movie$

The Movie$ corpus is a collection of professional reviews by film critics. It was collected by researchers at CMU and is available here:

cs.cmu.edu/~ark/movie$-data/

The data is located in *Vault -> Discourse Lab -> Data -> Movie$_corpus*. The POS data is available in the folder *pos*.

- Cornell

The Cornell movie review dataset contains amateur movie reviews created by Internet users. It was collected by researchers at Cornell University and is available here: cs.cornell.edu/people/pabo/movie-review-data/. Note that while there several datasets on that page, the one we are interested in and have used so far is called polarity dataset v2.0

Our copy of the data is located in *Vault -> Discourse Lab -> Data -> Cornell_reviews*. The raw texts are in the *raw* folder, the POS data is available in the folder *pos*, and the file *cornell_all_reviews.txt* contains all the reviews, separated by two newlines, for easy readability.

- NYT

The New York Times has an API that can be used to access all Movie Reviews that they have published since 1915. With this API, I collected the names and URLs of all movie reviews till the 17th of January 2019. Then I scraped the NYT website for the content of as many of those reviews as possible. The details of the data collection are in the NYT Extraction section below.

The cleaned data is located in *Vault -> Discourse Lab -> Data -> NYT*. The raw texts are in *nyt_reviews_all.csv*. These include multiple entries of some reviews where there were multiple movies reviewed in one publication and so they are listed separately by movie being reviewed, but the content of the review was always the same. The file *unique_review_texts.txt* has the unique reviews of those. The POS data for these unique reviews is in *NYT_pos.csv*.

NYT Extraction

The New York Times has a Movie Reviews API that can be used to get at all the movie reviews they have published over time, including older ones before the digital versions which have been digitized using OCR. The API will return URLs for the movie reviews. along with the headline, byline, publication date, title of movie being reviewed, opening date of the movie, and a short summary. After collecting a list of all the URLs, I needed to scrape the New York Times online to get the text content of all the reviews.

First you will need to get an API key for the NYT Movie Reviews API. Follow the instructions at this link to do that: developer.nytimes.com/get-started.

Unfortunately as of the creation of this document, the *publication-date* API parameter is broken. This means that we cannot query the API by the date that reviews were published (which is really what we are interested in).

The other API parameters pertaining to dates are *opening-date* and *last-updated*, neither of which work because the former is the opening date of the movie and the latter is irrelevant because articles could be updated even years after being published.

Luckily, results are ordered by publication date by default. So what I did was to just hit the API page by page and collate all the results together.

The additional trouble with this was that the server where the API is hosted is quite moody. There were lots of weird server error codes that were thrown in the middle of my trawling for results. The only solution is to catch errors and try again.

The code for this section is in **NYT 1** (extraction) and **NYT 2** (collation and data cleaning).

The next step was to scrape the NYT website using Beautiful Soup. The code for this is in **NYT 3**, where I downloaded each movie review as a text file on its own. This was the easiest approach for me to manually check that the scraping was happening correctly.

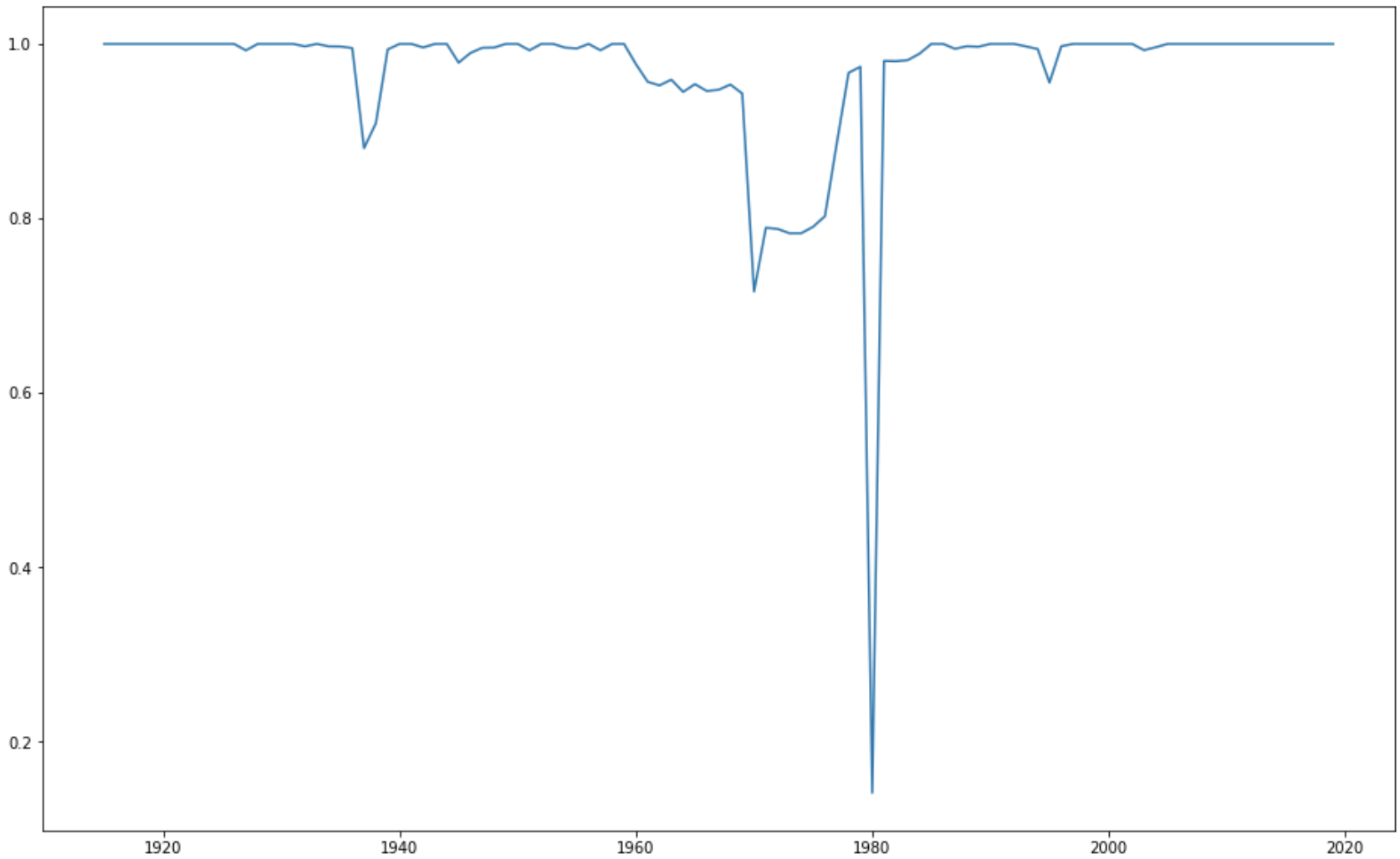
If you are doing this by searching for the movie review on NYT you will quickly run out of your 10 free articles for the month. Contact Maite for the lab's NYT credentials so you can read as many reviews as you want.

After scraping, there were still 727 missing reviews because the API provided broken or invalid links for these. To find some of these, I used the NYT Search page. I tried a number of combinations - searching NYT for the headline, the byline, limiting by date, etc., and then picking the first of the search results, but no approach was perfect.

Sometimes it would seem like results were found but they were actually the obituary of that NYT movie reviewer, or it was a review for the book or a play, not a movie. I had to be fairly restrictive and I finally managed to get another 105 reviews added to the list. All of this scraping for missing reviews is in **NYT 4**.

Finally, in **NYT 5**, I put it all together. The texts of all the reviews in one CSV along with the metadata I got from the API, in one CSV. Note that there were duplicates in the review texts, because sometimes 5 movies were reviewed in the same publication. The API would have a result for each of the 5 movies with the exact same URL, and therefore the exact same text. This is why the unique review texts are in a different CSV from this main CSV that has everything.

Another peculiarity of the NYT corpus: of the missing 600 odd reviews, most of them are from 1980. For whatever reason, their links for 1980 seem to all be wrong. Refer to this image below to see what proportion of the articles from the API we actually have the texts for. Notice the big dip right at 1980. Other than that we have most of what we were looking for.



Proportion of NYT movie reviews for which I could scrape the text 39.2 KB View full-size Download

Data Pipeline

The first time I approached the data pipeline, I did it inefficiently. I was generating lists of adverbly adjectives for each corpus differently because every corpus was in a different format and/or had to be preprocessed differently.

So when I had a chance to fix my approach, I chose to create a pipeline where I would normalize every corpus to a simple format that contained all the information I needed, and then I would create generic scripts / notebooks to extract adverbly adjectives, happy legomena, median lists and summary statistics from that generic format.

The normalized format of the corpus is a flattened CSV where there are 5 columns (two of which is optional): **text_num**, **token**, **pos**, **genre** (optional)

and **decade** (optional).

Sentence boundaries are not necessary because in this case we only care about adjacent tokens of a certain pattern - the first one is an adverb ending in '-ly' and the second one is an adjective. If genres or decades (or both) are applicable for the corpus, e.g., CORE/COCA or TIME/NYT, then the fields for genre and decade can be added as well. It is implicitly assumed that one text will have the same genre all through.

One disadvantage of this approach is that it isn't terribly successful with COCA because COCA is enormous, which is why one of the sections in this document includes notes on how this particular corpus was handled. However, for 6 other corpora on which this pipeline was used, it worked like magic.

Getting different corpora into this standard format will take different forms as some need to be preprocessed for XML tags, some may be in CSV format already but others may be in JSON or XML or separate text files in a folder. Thus there are separate notebooks for each corpus titled **< corpus_name > - Extracting POS**.

Once this POS data is collected, put it all in a directory named *pos* at the same level as the notebooks for the next stages of the pipeline. Then running the adverbly adjectives pipeline is simply a matter of re-running the notebook. It looks for all the corpora in the *pos* directory and filters adverbly adjective combinations in all of them. The results are collected in a new subdirectory called *adverbly_adjectives* at the same level in the hierarchy. If genre, decade or both were provided, there will be lists of adverbly adjectives provided for the entire corpus as well as by genre and by decade. The lists include the adverbly adjective, along with its raw and normalized counts in the corpus (or by genre or decade as specified). They are arranged by count in descending order (i.e., most frequent combinations first).

Once this is done, run the hapax legomena pipeline. This simply creates sublists in exactly the same format as the adverbly adjectives, consisting of all the pairs that appear only once in the corpus or the decade/genre subcorpus.

You can also run the median/middle lists pipeline at this point. The motivation behind this was that there would be interesting adverbly adjective pairs in the middle of the distribution - not hapax legomena and not the most frequent ones.

Unfortunately, if you look for the true median of the distributions of adverbly adjectives, you always get a hapax legomenon. Inevitably the 50 before and after that one also tend to be hapax legomena. We already have lists of hapax legomena so this information is useless. To get a more interesting subset, I began by throwing away the hapax legomena. Then from this I created “median lists” and “middle lists.”

Median lists: From the remaining adverbly adjectives, I found the median +/- 50. This tended to consist of pairs with relatively high frequency, e.g., for COCA they had raw counts between 461 and 6204. This corresponds to exactly the first 100 pairs sorted by frequency so again it’s not quite the middle of the distribution. For other corpora it might have worked better.

Middle lists: From the remaining adverbly adjectives, I took the middle of the list +/- 100. So for example if I had 500 pairs that were not hapax legomena, this would correspond to rows 200-300. This seemed to have more interesting data. For COCA the raw counts were around 3 all of these pairs.

Finally with all of this information, the summary stats pipeline can be run. The output of this final pipeline is two files - *corpus_stats.csv* and *genre_stats.csv*. They contain the following statistics by corpus and by genre, respectively:

- num_tokens: This is the number of words, not the number of actual tokens (for normalized counts later). Using the POS data in the *pos* folder, what is counted as a word is any token that contains at least one letter - so **4th** would be a word, but **4** would not. **I'm** would be a word, but **!!!!** would not.
- adv_adj_counts: This is the number of adverbly adjectives, the raw counts.
- hapax_legomena_counts: This is the number of hapax legomena, i.e, the number of adverbly adjectives that have frequency 1.
- adv_adj_freqmil: This is the normalized frequency of adverbly adjectives, calculated by dividing adv_adj_counts by num_tokens.
- hapax_legomena_freqmil: This is the normalized frequency of hapax legomena, calculated by dividing hapax_legomena_counts by num_tokens.

Another piece of information that might be useful is the 500 frequencies count. The top 500 most common adverbly adjective pairs in COCA were taken and annotated for subtype according to the classification developed by Maite and Cliff. Since it is interesting to see the distribution of raw counts and relative frequencies of these top 500 pairs across all the other corpora (and genres), I have a notebook to generate that as well. You can run it if you have the following file from Vault: *COCA_top500_annotated_sorted_frequency.xlsx* in the top-level directory along with the notebook and the *adverbly_adjectives* directory (after you have generated them).

Scripts - What and Where

All the scripts are on GitHub under the adverbly adjectives project - github.com/sfu-discourse-lab/adverbly_adjectives

As per the descriptions in the Data Pipeline section above, there are

subfolders that contain notebooks for all the following tasks:

- Corpus conversion - to flat format with POS
- Adverbly adjectives extraction
- Hapax legomena
- Median lists
- Summary stats
- 500 frequencies across corpora
- Comparing movie reviews
- NYT
- Visualizations
- Oxymorons

Notes on COCA

COCA is not standard as it is extremely large. It is unwieldy to convert it to the flattened POS format used for all the other corpora and in addition, since data is missing, we decided to just use the existing POS annotation that comes with the downloadable version of the corpus.

Helen had already extracted lists of adverbly adjectives for COCA using this pre-existing POS data, so I simply pruned the list to make sure that all the words were alphabetical (no punctuation separating the components of the pair), and normalized with the counts in the downloadable version of COCA.

These counts are available in the file *coca_wordcount.xlsx* which is in *Adverbly adjectives -> COCA*.

The only two issues with this approach - which is why these data must be taken with a grain of salt - are:

1. The POS data were not tagged with spaCy as for all the other corpora, and we don't know what POS tagging was used

2. As mentioned before, every 200 words, 10 words are replaced with '@' symbols so we don't have any way of accounting for those

Movie Review Analysis

We were interested in comparing the use of adverbly adjectives between professional and amateur movie reviews as they are both forms of opinionated texts, but that have different characteristics. I looked at keyness, lexical diversity, word and sentence length, common word unigrams and bigrams, as well as common POS unigrams and bigrams. The results are summarized in this document: Professional vs laypeople's reviews - Adverbly adjectives

Oxymorons

The oxymoron subproject did not get anywhere useful and so we are leaving it aside for now.

What we wanted to investigate were adverbly adjective pairs where the sentiment polarity of the adverb and the adjective were in opposite directions. The contrast creates an interesting effect and if we were able to generate lists, it might have been nice to systematically analyze how their meaning was created as a compound. Examples are beautifully ugly, violently happy, etc.

There is documentation in the bucket Oxymorons - Adverbly adjectives on where we got with this project and the different approaches taken. No approach was good enough to reliably come up with lists.