# Time Series NLP

Exploring Author Characteristics from Longitudinal Writing Samples


Adam Laughlin
Computer Science
University of Colorado Boulder
adam.laughlin@colorado.edu

Jiaheng Zhao
Computer Science
University of Colorado Boulder
jizh3194@colorado.com

Kyle Bremont
Computer Science
University of Colorado Boulder
kyle.bremont@colorado.edu


https://docs.google.com/spreadsheets/d/1W16oubAN8cnCcXPauFbGFYG_GsfnHex-JeBOz69zDUg/edit#gid=0

## 1 Problem Statement / Motivation

What can we learn about a person through their writing? We apply natural language processing (NLP) and other data science methods to a single-author finance blog spanning 17 years. We seek to identify relatively stable characteristics such as introversion/extraversion and more variable characteristics such as affect and optimism. Of particular interest are change patterns (e.g., cycles) in variable characteristics. If possible, we'll incorporate additional data such as market changes and news events to provide context for changing characteristics.

## 2 Literature Survey (Previous Work)

Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text Fantastic exploration of linguistic features used in personality trait prediction. The features are likely useful in non-personality cases.

Linguistic Styles: Language Use As An Individual Difference Another text breaking down linguistic cues that can be used to classify personality characteristics. Between this and the last paper, we have many linguistic cues to work with, though no specific python framework or project tying them together.

A Framework for Emotion Mining from Text in Online Social Networks: Excellent resource for thinking about how to derive affect from text, plus well-documented pre-processing, features, and evaluation metrics

Approaches towards Emotion Extraction from Text: Comparison and classification of affect detection methods. Conclusion for our context is finding a hybrid model is likely best for precision, followed by keyword for explainability.

Emotion and Sentiment Analysis: A Practitioner's Guide to NLP: Extracting sentiment via two python tools

Lexical Predictors of Personality Type: Seeking lexical predictors of personality type and other characteristics. Good discussion on why they might choose some types of features vs others.

LIWC Language Manual - The Development and Psychometric Properties of LIWC2015: Many subtle issues to consider when mapping text to psychometrics.

## 3 **Proposed Work**

3.1 Possible Starting Models

- None
- Trained NLP Model
- Trainded NLP + Dataset(s) mapping psych characteristics to writing samples
- Trained NLP + Papers Mapping NLP attributes to psych characteristics
- Trained NLP->Psych Model (LIWC)

3.2 Work starting with Model:

3.2.1 None:

- Train NLP Model on large text dataset (basically create NTLK or spaCy)
- Goto 3.2.2

3.2.2 Trained NLP (e.g., Spacy):

- Collect Psych&Text Dataset
- Goto 3.2.3

3.2.3 Trained NLP + Dataset(s) mapping writer characteristics to writing samples:

- Extract many text units (words, ngrams)
- Extract many text features per unit (e.g., tense, word count, word length, std devs, other custom, like tentativeness for "I think"...)
- Customize Dictionary (e.g., "kind of")
- Train model to recognize psyc
- characteristics from text
- Isolate most important features per characteristic
- Save as Trained NLP->Psych Model

3.2.4 Trained NLP + Papers Mapping NLP Attributes to writer characteristics

- Decide doc units (word, noun chunk, sentence, paragraph, document, document collection)
- Create map of linguistic features to psych attributes based on those present in papers
- Train NLPsych model that extracts linguistic features, e.g.:
    - tokens (e.g., 'I ran' -> ['I', 'ran'])
    - semantics (meanings)
    - lemmas (e.g., did->do)
    - parts of speech (e.g., noun, verb)
    - named entities (Denver, My Dog)
    - Derive numeric features, e.g.:
    - words per unit
    - sentence length std deviation
- And maps them to psych features, e.g:
    - big 5 personality traits
    - sentiment (valence+intensity)
    - emotion (specific emotion words)
    - optimism
- Tweak until it gets similar results to the papers on the test data
- Goto 3.2.5

3.2.5 Trained NLPsych Model

3.2.5.1 Extract Blog Data

- crawl blog
- store site indices' post metadata
- store post content

3.2.5.2 Preprocess (done during crawl)

- pre-cleaning (ignoring empty posts)
- preprocessing (only text, no html or media)

3.2.5.3 Store

- Store posts+meta as files and JSON

3.2.5.4 Transform posts+meta into csvs

- Decide useful linguistic features
    - e.g. most frequent words
- Decide useful psych features
    - e.g., optimism, personality
- Clean (e.g., remove irrelevant columns)
- Normalize (e.g., zscore attributes)

3.2.6.2 Decide useful psych features

3.3 EVALUATE

- # of "interesting" patterns found
- Key results learned from this effort

## 4 **Relation to Prior Work**

4.1 Differences

- single-person long-duration (longitudinal) focus vs many-person short duration focus
- focus on attribute change over time
- broader focus than only sentiment or personality
- python for all steps >= 3.2
- if we have time, integrating context such as market changes and news to assess their correlation with other patterns

4.2 Similarities

- linguistic cues derived from text
- may also derive sentiment and personality

## 5 **Data Set**

5.1 Challenges: We were unable to find good datasets for longitudinal text analysis, so we created our own by writing a crawler to gather posts from a long-running blog.

5.2 Content: 38,723 single-author blog posts spanning 17 years. Posts contain text only. No pictures, videos, or other media. JSON metadata file contains year, month, day, title, author, url.

5.3 Storage: Posts stored as 212mb worth of individual text files (212mb), zipped to 54mb. Metadata stored as 10mb JSON file. Both available on github.

## 6 **Evaluation Methods**

Evaluating a data mining system is mainly based on five main aspects: accuracy, performance, functionality, availability, and accessibility.

accuracy
The most critical factor in evaluating a data mining system is accuracy. By performing the prediction and classification accuracy of the algorithm on the data mining system, we can judge whether the algorithm in the system is reasonable and whether the data acquisition is comprehensive and the data preprocessing work is perfect.

performance
We need to check if the system can run and the stability of its operation.

Feature
We need to check whether our system can make the user adjust the parameters of the algorithm and algorithm and whether it can randomly extract data from the data set to build a pre-drilling model; whether the mining results can be expressed in different forms.

Availability
We need to check if the visualization of the system is good;

Accessibility
We need to check if the user is allowed to change the error value in the dataset or to clean the data; whether to allow global substitution of values; whether to discretize continuous data; whether to extract subsets from the dataset according to user-defined rules; The null value in the middle is replaced by an appropriate mean or user-specified value; whether the results of one analysis can be fed back into another analysis.

6 **Tools** (Slightly out of date. New tools recently found and proposed work updated. We'll align it with current tools+work in the final paper.)

package management:
- python: pip + requirements.txt
- node.js: yarn + package.json

code storage
- github

Tools by Proposed Work Section

3.1 EXTRACT

3.1.1 Collect
- beautifulsoup or selenium or (I used puppeteer for simplicity and performance)
- nodejs
- chromium
- css selectors

3.1.2 Preprocess (done during crawl)
- custom javascript

3.1.3 Store
- nodejs "fs"
- github

3.2 TRANSFORM (TBD)

3.2.1 Integrate
- nodejs "fs" library
- combine text +metadata in csv

3.2.1 Clean
- Remove guest authors

3.2.3 Derive features/attributes

Derive doc unit metadata, e.g.:
- spacy (trying first for performance + documentation + simplicity)
- or ntlk
- or stanfordnlp

Derive numeric features, e.g.:
- numpy
- pandas

Derive categorical/ordinal features, e.g.:
- numpy
- pandas
- maybe other python libs (tbd)

3.2.4 Map doc units -> features, e.g.:
- custom python
- pandas
- numpy

3.2.5 Clean features (as necessary)
- spacy, or… need to research this

3.2.6 Normalize features
- custom python
- pandas
- numpy
- sklearn

3.2.7 Store derived features as pickle or csv
- pandas and/or pickle and/or spacy (TBD)

3.3 LOAD (TBD)
- pandas

3.4 ANALYZE (TBD)
- pandas
- matplotlib
- seaborn
- lime

## 7 **Milestones**

7.1 Done: Proposed work 3.1 (Extract)

7.1.1 Lessons Learned Adam

Learning about scraping, then implementing it, took about 40-50 hours, excluding the actual time to scrape once it was working. Add another 12 for scraping time, but that ran while I slept. There are so many little details and edge cases to handle, including the unexpected ones that crash a scraper like a post without a content div, or one listed under year "3". A complete scrape in one go is highly unlikely so it had to be able to do a partial scrape, crash, and continue where it left off. Ultimately it comes down to writing a error-tolerant, thorough-logging, interruptible, parallelizeable, asynchronous graph traversal algorithm. Plus deeply understanding the site's information architecture and html structure. Data format(s) and timing are also an important choice, since serializing+storing a GB-sized object on every page visit brings your crawler to, well... a crawl.

Also read many research papers under the assumption we would be manually mapping their ffeatures to linguistic cues extracted by toolkits like spaCy, NTLK, and StanfordNLP. Learned a lot about scraping, crawling, blog architectures, NLP toolkits, scraping toolkits, and linguistic constructs.

7.1.2 Lessons Learned Henry

I learned how to use tools to crawl our target files and store the collected data in GitHub.

7.1.3 Lessons Learned Kyle

7.2 Done: 7/18 Iteration 1

7.2.1 Lessons Learned Adam

Discovered that previous researchers published a tool they developed that maps raw text to psych features. Got it working, and was able to generate ~80 features for us to do analysis on.

Also realized that we were working with multiple models at different stages, with different ways of creating those models. That enabled rewriting proposed work starting based on model stages (Section 3.1).

7.2.2 Lessons Learned Henry

I learned how to clean data and classify important data.

7.2.3 Lessons Learned Kyle

7.3 Done: 7/25 Iteration 2

7.3.1 Lessons Learned Adam

Wrote a pipeline transforming the initial raw blog data and metadata into increasingly derived features. Did initial cleaning by aggregating date fields, removing author (since there's only 1), removing filename since it's irrelevant after deriving data. In the middle of EDA currently.

Having one or more agreed upon places where cleaning happens may cost processing time, by leaving irrelevant data in longer, but can save confusion and communication overhead about the state of data fields at different points in the pipeline. This is especially important when integrating data from multiple sources and indices.

7.3.2 Lessons Learned Henry

pipeline

7.3.3 Lessons Learned Kyle

Extracting content from text files and appending the data to a csv file.

7.4 TBD: 8/02 Iteration 2

7.4.1 Lessons Learned Adam

7.4.2 Lessons Learned Henry

7.4.3 Lessons Learned Kyle

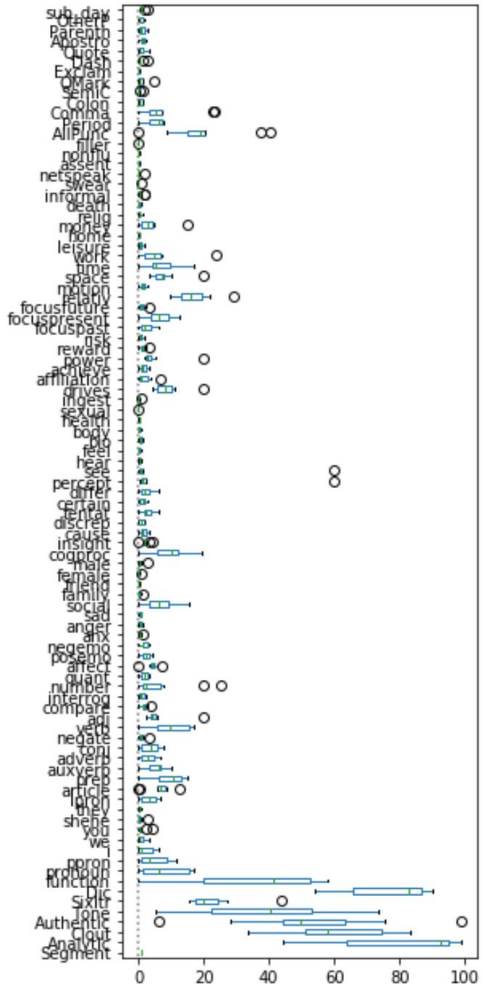
7.5 TBD: 8/09 Final Report Written

## 8 **Results**

Adam:

A fair amount of communication, organization, and administration overhead, plus a midterm in another class, led to less time than I'd have liked for actual analysis. My results mostly consist of cleaned and transformed data, plus a number of psych features derived with a new tool, and the integration of those two. Initial EDA on the integrated features yields around 95 total, mostly floats of various ranges above 0.

```
ax = df3.drop(['WC','WPS'],axis=1).plot.
ax.vlines([0],ymin=0,ymax=100,linestyles
```




Boxplot excludes Word Count (WC) and Words Per Sentence (WPS) do to size differences.

Next steps involve dimensionality reduction and normalizing (e.g., zscoring) as appropriate. Then on to another data integration phase to get actual words used, and pattern detection, clustering, and signal analysis.

Henry:

After statistics, I found that the author used a lot of words about emphasizing his own point of view. so I think the author is a subjectively conscious person. Although sometimes he uses some objective analysis to illustrate his point of view. But that's just a way for him to convince others to agree with his personal point of view. Furthermore, From the analysis of the data we can find that The author is a positive person. The data provided to us by LIWC can prove my judgment. Because if we want to judge whether a person's personality is positive or negative, it can be judged based on the statistical results of the emotional words in his article. According to the data provided to us by LIWC. Positive emotional words are almost twice as negative as negative emotional words. So we can think of him as an positive person.

Kyle:

For my analysis I decided to look at if the author of these blog posts is overall a positive or negative person based on what he writes about. LIWC provides us with two statistics pertaining to this: negative and positive emotion words. I took the average scores for both throughout all of his blog posts. His negative emotion average was 1.48 and his positive emotion average was 2.67, almost double the negative emotion average. This leads me to believe the author is a more positive person overall.