Using Natural Language Processing to Analyse the Shape of Stories

Luca Davies

BSc. (Hons.) Computer Science

March 2020

1 Abstract

This is an abstract

2 Introduction

2.1 Test

Writer Kurt Vonnegut suggested at various points in his career that all stories may be categorised into a relatively small number of basic archetypes based upon the emotional ups and downs experienced within the narrative. He gave each of these novel names such as Man-In-Hole and Boy-Meets-Girl as a simple reference for each.

Literature is one of the defining features of the human race. No other creature on Earth can writes, let alone writes about itself. Humanity takes this a step further again, making up fictitious stories, some rooted in reality, some with no basis at all in which we dream and imagine worlds and situations that may never be possible to achieve. Through written word we express emotion. All emotions, happiness, sadness, anger and fear; love, hate, awe and grief – everything humans can feel, we write about. We create characters to express and receive these emotions, they act as vehicles to transfer their feelings to a reader. With all this freedom to write and create without bound, are we actually as free as we perceive?

Vonnegut suggests that all stories are members of one of a very small number of categories of story that define roughly how the emotional progression of that story pans out. Do writers naturally and unknowingly write literature that falls into these types or is each story as unique as the next, taking its readers along its own path as they go?

This concept and these questions drew me toward this project proposal. Not only is it a fascinating endeavour to map the emotions of a novel, but as an exploration of the freedom of writers to convey emotions and of how humans often generate categories and groups without even trying.

2.2 Motivation

2.3 Aims & Objectives

The aims of this report are as such:

- Design and develop an application to process a range of corpora to produce a graphing of the emotions during its literary course

- Analyse a range of corpora for compliance with Kurt Vonnegut's theories and story shapes

- Otherwise attempt to identify trends the texts processed

- Study the produced graphs and compare expectation surrounding the source text

3 Background

3.1 Overview

This chapter will examine and summarise existing literature and studies in this area and topic. That is, natural language processing and sentiment analysis as a tool for extracting statistical data that describe emotions in (mainly) fictional works of literature.

The processes involved for finding useful information included searching for online articles and pre-existing projects while searching libraries to use in the implementation, Google Scholar, Lancaster University Library OneSearch and Kurt Vonnegut's own lectures on this topic.

Existing projects will help to prove the developed application is performing up to standard (or not) and can be used as a side-by-side comparison of literature processed for this report and by these projects as well as to examine the processing of literature unavailable for the purposes of this report.

3.2 Kurt Vonnegut on "The Shape of Stories"

Vonnegut described a number of potential story types as displayed below in figure something.

3.3 The Hedonometer Project

The Hedonometer project was established to gauge happiness, the world over, starting with Twitter and other social media outlets. The project's scope has since widened to process corpora direct from books, film scripts, news outlets and other foreign language literature.

4 Sentiplot Tool

This section will detail the design and implementation of the tool used to carry out this study, *Sentiplot*.

4.1 Languages & Libraries

This section briefly discusses the choice of programming language and NLP/SA tools used in the Sentiplot tool.
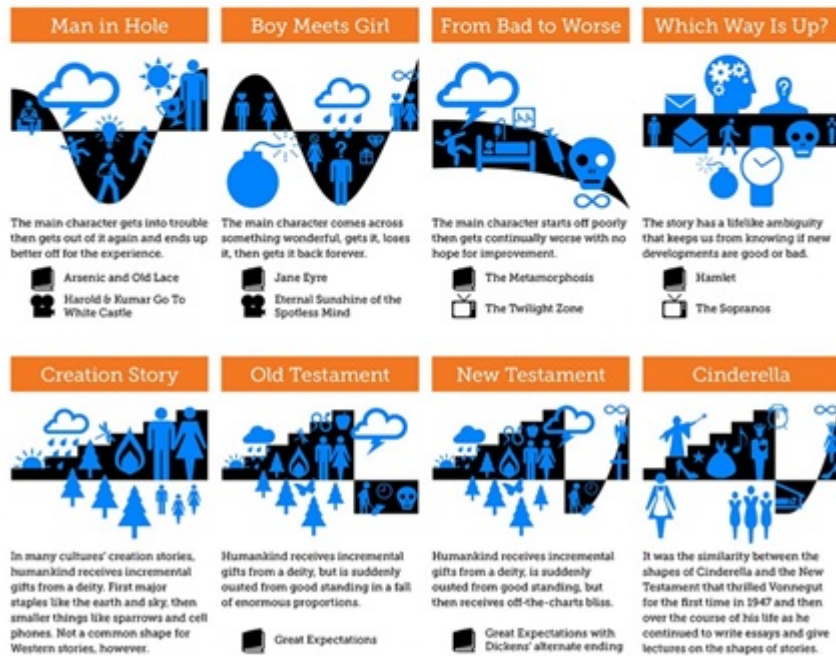
Figure 1: Vonnegut's Story Types

4.1.1 Top-level Language

When selecting which language would be most appropriate to use for this project, my major considerations were my prior knowledge of potential languages and their ease of creation of a pleasant user interface. Availability of NLP tools in each language was also to be considered, but as the end implementation shows, this was not imperative due to cross-platform and cross-language capabilities of the chosen combination of languages and tools.

A listing of considered languages follows:

- Java

  - Strong language knowledge and familiarity
  - JavaFX and Swing availabile for interfaces
  - Native lanuage of Stanford CoreNLP

- Python

  - Very limited language knowledge
  - No knowledge of interface building
  - Native language of industry standard NLTK

- C# (with .NET)

  - Strongest language knowledge and familiarity
  - Extreme ease in creating interfaces via Windows Form using Visual Studio
  - Cross-platform variants of CoreNLP and VADER availabile via simple packages

Taking all these points into consideration C# was selected, due to its pros specific to myself and the availability of non-native libraries via APIs. This permitted me to use the full Microsoft Visual Studio suite for development, including the Windows Forms designer.

4.1.2 Natural Language Processing Tools

Various tools across a wide array of languages are freely available to provide standard NLP functions and advanced processing capabilities.

Initially a full CoreNLP pipeline was employed to process corpora but this proved to be extremely slow, loading around 2 gigabytes of models very slowly into memory before even processing anything. The pipeline was modified to tokenise and sentence-split only and the actual sentiment analysis was performed by VADER.

Both the CoreNLP and VADER libraries are non-native to C# but have easy to use APIs for direct manipulation of their types and methods outside of their native environments. CoreNLP has an in-house developed API for C# and VADER has a third-party API called *VADERSharp*.

4.2   Implementation

4.2.1   Structure

The application is written in C# .NET interfacing with both Java and Python libraries via APIs as detailed in the previous section. Windows Forms was chosen for the interface as a quick and easy to build platform, stable on Windows with seamless integration into C# and the .NET framework.

The application is composed of two forms, each with their designer code. The first allows the user to select a file to load text from and set processing granularity. The second presents the results of the analysis in multiple ways and provides facility to save these results as images of the output graphs.

4.2.2   Sentiplot Form (Main Form

This is the main form for the application. It is loaded on start-up and contains all needed information or otherwise calls other classes to complete its task.

Its key functions include:

- Initialising the CoreNLP pipeline ot tokenise the input text

- Generating an OpenFileDialog object to allow the user to select a *.txt* or *.html* file

- Parsing the content of the input file to prepare it for processing using regular expressions and simple character-based splitting

4.2.3   ResultsViewer

This form displays the results of VADER's processing of the input text. It displays a graph of the entire text from start to finish, a full list of all tokenised sentences and their associated sentiment scores (in tabular form) and individual graphs for each chapter in the text (HTML file only).

- Handling the output data and feeding it into the main chart and table

- Allow hiding/showing of each different type of sentiment score for the main graph

- Dynamically producing more tabs on the form to show each succesive group of five chapters

4.3   User Interface

The implemented interface did not need to be excessively complex or particularly appealing as it mainly served to function as a harness to conduct the study, with more importance placed on the code-behind and results.

To provide quick results and ease of programming, I used the Windows Forms suite to produce a visually simple but functional interface. It has two screens: a

screen to load the desired text to be processed and to set the granularity of the analysis, and a screen to present the processed results. The following sections provide an overview of these screens.

4.3.1   Text Selection & Options

Figure 2 shows the main presented screen, after having loaded an HTML file (Hamlet in this case) which has been parsed to produce plaintext, having stripped out all the unneeded HTML tags.
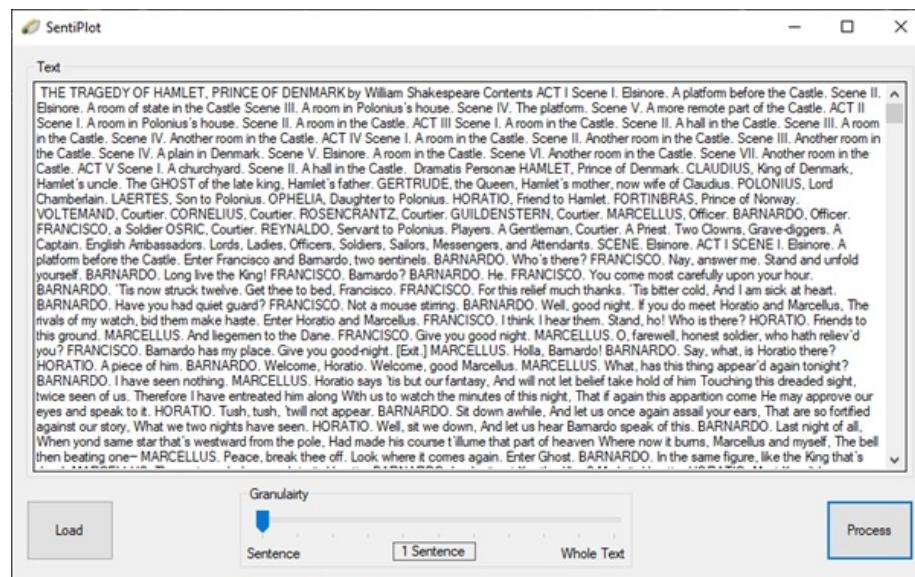

Figure 2: Main Sentiplot window

4.3.2   Results Display

Figure 3 shows the results screen for Hamlet, initially showing the graph of the entire text. The maximum and minimum points are labelled with the start of the related sentence. Hovering the mouse over these labels shows the entire sentence or sentences that produced that datapoint.

For every sentence analysed VADER returns four sentiment scores: positive, neutral and negative (each holding a -1 to 1 score regarding match to that sentiment), and compound which acts as a single representation for the sentiment in the sentence parsed. Each of these scores have their own graph which may have their visibility toggled on and off with the check boxes in the bottom left. (Default is compound alone, as it proved the most indictive of sentiment, as advised by VADER documentation.) This graph can be saved to a JPG by clicking the save button.
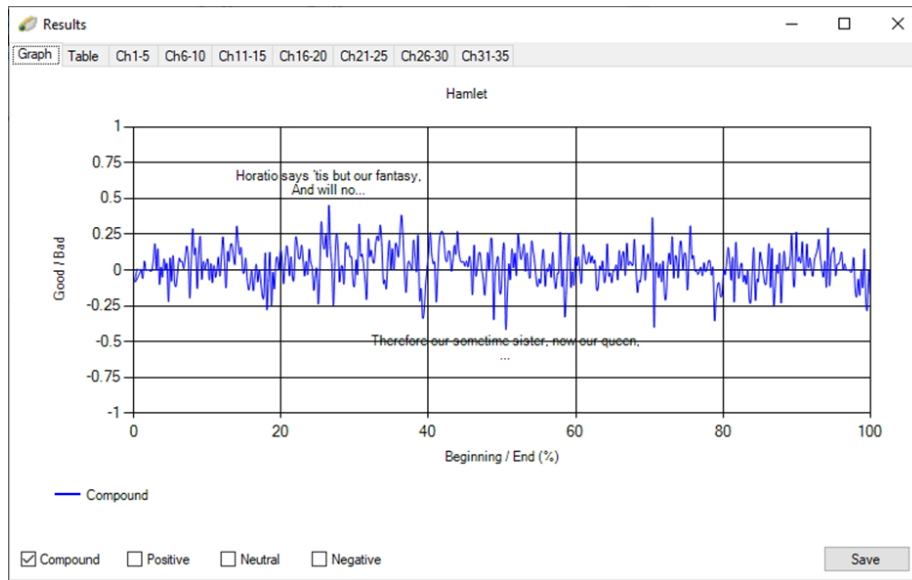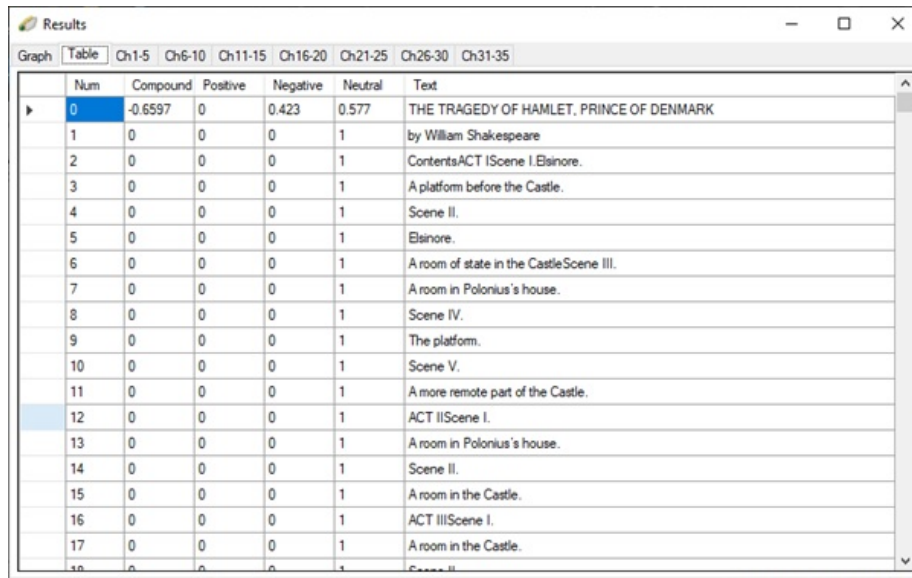
Figure 3: ResultsViewer window

Figure 4 shows the Table tab of the results screen. This simply lists each individual sentence token in the input with VADER's output value for the four scores mentioned previously. The table's default ordering in by sentence index, or the chronological order in the text, but the it may be order by any of the fields shown.

Figure 4: ResultsViewer window showing table

Figure 5 shows one of the chapter tabs from the results screen. Each of graphs the compound sentiment score of up to 5 chapters from the processed text. Again, maxima and minima are labelled with their relevant sentence(s), expandable by hovering the mouse over the label.
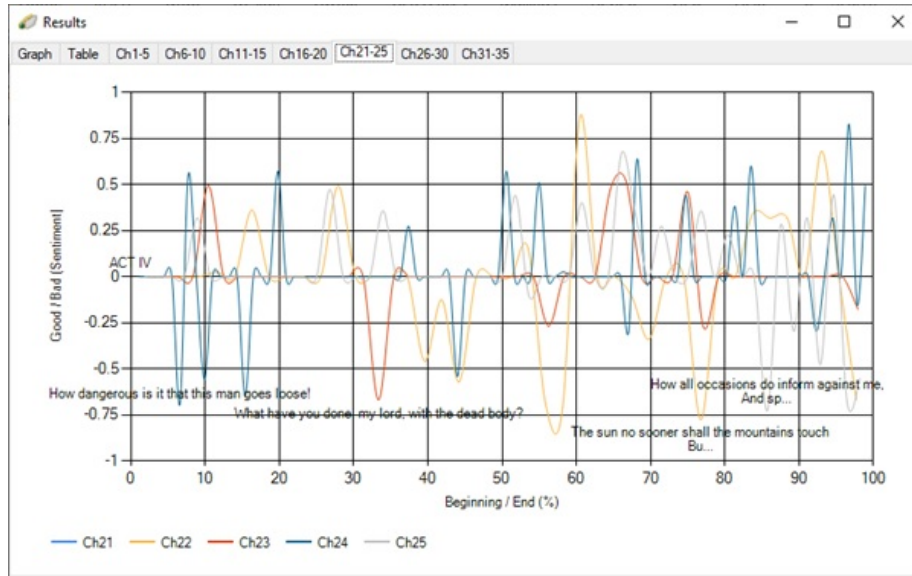
Figure 5: ResultsViewer window showing chapter graphs

5 Experimentation

The following section covers the various experimentation and analysis carried out using Sentiplot as a tool for Sentiment Analysis and graphing accordingly.

5.1 Analysis Block Size

Sentiplot incorporates one primary setting for its analysis of texts: the granularity slider. This allows the user to select varying degrees of granularity for the produced graphs (as a percentage of the text being analysed). This allowed analysis of texts at a number detail levels to find which produced the most obvious curves to conform with Vonnegut's story curves. This may have introduced a small amount of positive-results bias, accepting those granularities that appear most curve-like for a given text instead of what may have acutally reflected the curve of the text. Figure 6 show the differing graphs that may be produced by changing the granularity.

The following options are provided:

- 1 sentence

- 0.1%

- 0.2%

- 0.5%

- 1%

- 2%

- 5%

- 10%

- 25%

- 50%

- 100%

VADER is designed for parsing short lengths of text (less than 140 characters, as per a Tweet), so in order to produce more coarse graphs than single-sentence, VADER is still fed text on a per-sentence basis with the results summed up to N sentences where N is the corresponding number of sentences for a given granularity setting for a given text. The relevant data point is then plotted N sentences beyond the previous (i.e. at the end of the analysed block).

By default, every sentence of the text is plotted as its own point. This gives highly detailed graph but it very difficult to discern any sort of shape. Sentence to sentence, sentiment can vary wildly from the most negative in one to the most positive in the next (context and corpus dependant). Due to this, the resultant graph is highly spiked and difficult to garner information from.

By contrast, using the more coarse options of 25% and upward produce graphs that do show a clear line, however, at this level of coarseness, the data is compressed so much so that much of the detail is lost, smoothing the curve that should be shown beyond useful limits.

Through trial-and-error experimentation, the ideal setting was found to lie between 0.2% and 2%, erring toward the 2% for most texts.

With this being said, it's possible, that a different tool may have been better suited to this task than VADER due to it's native design choice to parse text the length of Tweets, not entire corpora.

5.2   Hand Analysis Vs. VADER

VADER also lacks any context processing - it cannot understand running themes in texts in order to better understand the sentiment behind that which is being said.

5.3   Reader Analysis & Reflection

5.4   Shakespeare: Comedies Vs. Tragedies

5.5   Early Modern Vs. Modern English

VADER is not trained to work on the likes of formal literature. In fact, it is trained to work on highly modern and informal text, to such an extent that it understands and can analyse Internet slang and emoji.
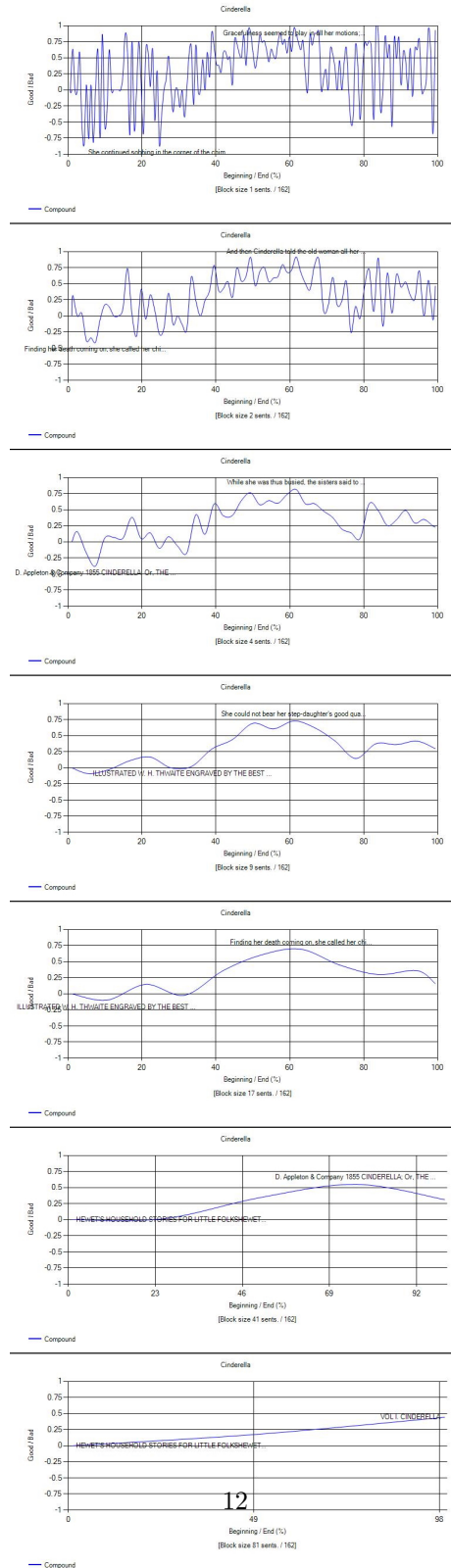
Figure 6: Each graph shows the result of decreasing the granularity of analysis and show how there appears to be a sweet spot. In this case, four of the highest options summed only a single sentence due to short length of the text. Only one of these has been included.

6 Conclusion