Representing Films as Character Graphs

Victor Dumitrescu

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2015

Abstract

This project sets out to explore a representation of film in terms of its characters and their interactions, bringing together different kinds of textual information, such as screenplays and plot summaries. The work described in this report builds upon the existing literature in computational linguistics concerned with studying characters in films and other works of fiction. The character graph model presented here captures two main aspects of film characters: their personalities and their role in the social network of the narrative. We discuss the process of building and evaluating character graphs and show that they accurately predict the sentiment that film characters express towards each other through dialogue.

Acknowledgements

I would like to thank Jon Oberlander for his inspiring supervision. I always left his office with something to ponder and excited about the direction that I would take the project next.

I am grateful to Philip Gorinski and Mirella Lapata for providing me with their Script-Base corpus and their research.

I would also like to thank my family. Were it not for their continuous support, I would not be submitting this project today.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This project started with the idea of constructing a meaningful representation of films, that would enable computational analysis. A film can be seen from many, sometimes complementary, points of view: as a story, comparable to literary narratives [21], as a social network of characters or as a succession of scenes. Engrained in our appreciation of films is the belief that we can abstract common aspects and describe them in terms of these qualities. Film genres are perhaps the most prevalent such example in cinematography. Film going audiences will generally discern between comedies and thrillers, for example. This is of course not limited to films. Kurt Vonnegut [31] famously illustrates the "shape of stories", such as "boy meets girl" and "Cinderella" as a curve on an *ill fortune - good fortune* axis[1]. A more comprehensive analysis of plot structure (controversially) proposes that most stories can be categorised as one of seven "basic plots" [9].

Our work will not address this idea directly, but this point will come up again in the final section of the report. Instead, we will take a data-driven approach and explore the ways in which natural language processing and machine learning techniques can be used to derive some insights into the film's narrative. Such representations can be useful in tasks such as automatic summarisation and automatic generation of stories (more details follow in the next chapter). Machine analysis of film scripts is also applied in the entertainment industry, with companies like Epagogix[2] claiming to be able to predict the financial success of a film based on its script and suggest improvements that would increase it.

1.1 Accomplishments

The main accomplishments of this project are the development and evaluation of a character graph representation of films. In order to achieve this, we had to design and implement a software system which can:

[1]I would like to thank my fellow student Craig Wilkinson for bringing this to my attention.

[2]www.epagogix.com

- **Process the data**, mainly film scripts and plot summaries; we also built tools to acquire data (similarity judgements, metadata) from online sources
- **Construct a probabilistic personality model** of film character, using plot summaries of films
- **Construct a character graph**, using the persona model and character interaction data derived from the film scripts, based on a topic model
- **Construct sentiment trajectories** of films using sentiment analysis techniques and derive labels that can be used to evaluate character graphs through a suite of prediction tasks
- **Construct adequate feature vectors** that enable machine learning tasks; support a set of transformations on these feature vectors in order to carry out the prediction tasks under different conditions
- **Train and evaluate different predictors** and allow parameter optimisation in order to achieve best results
- **Export data**: plotting character graphs and sentiment trajectories, pickling and exporting data to various formats
- **Cope with rapidly changing requirements** and enable easy customisation and addition of features

The final snapshot of the code submitted for this project does not capture the full evolution of the system, which had to be constantly changed to implement and test new ideas. Naturally, for many of these tasks, I relied on external libraries, which are referenced when discussing the relevant aspects of the implementation.

1.2 Structure of the report

The report is split into the following chapters:

- Chapter 2 (Context and background) introduces and explains ideas from different areas of research, on which the work described in the following chapters is based
- Chapter 3 (Aims and hypotheses) presents the central hypothesis of the project and discusses alternative hypotheses which were tried but could not be evaluated
- Chapter 4 (Design and implementation) gives a detailed account of how the tasks outlined in the above list were implemented and motivates the design choices that were made
- Chapter 5 (Results and discussion) offers detailed results (both successful and unsuccessful) for a number of prediction tasks and assesses the strengths and weaknesses of the character graph representation

- Chapter 6 (Conclusions and future work) presents possible extensions of the work done for this project, lists alternative approaches that could have been considered and attempts to place the results in a wider context

Chapter 2

Context and background

2.1 Background work

Researchers have been interested in computationally analysing narrative fiction for decades. However, this interest has dramatically increased along with the size of raw data available and the ability to store and process ever-growing data sets. In order to establish the basis of the work described in this report, we will highlight some of the previous literature which takes a character-centric point of view when approaching narrative fiction. This line of work is intertwined with many other topics in social network analysis, machine learning and sentiment analysis. We will will also give an overview of the contributions in these area which serve as a basis for our work.

2.1.1 The Persona Model

One of the things we can think of doing when talking about fictional characters is constructing a representation of their personality. Recent work by David Bamman and others has explored the idea of learning the type or *persona* of film [5] and literary [6] characters. A character persona is a probabilistic model of a character's personality traits, constructed using certain keywords from a film's plot summary. The cited papers present a number of variations of this model, but we will only describe the *Dirichlet Persona Model* introduced in [5], which is the model that we adapted for this project.

Instead of trying to build these models of character personality using information conveyed in the films themselves, such as dialogue or descriptions from the screenplays, Bamman and his colleagues relied on summaries of films' storyline, written by human editors on Wikipedia. These are usually concise, straightforward descriptions of the events and characters of a film, written in a neutral style. A more detailed discussion of the this data follows in Section 2.2.

A core assumption in their model is that character personalities are primarily represented by three types of words in the plot summaries:

- **Agent verbs:** Verbs for which the character is an agent (e.g. "Starling *travels* to the victim's hometown")
- **Patient verbs:** Verbs for which the character is the patient or object (e.g. "Starling *is led* to the house of Jack Gordon", where *Starling* is a passive nominal subject)
- **Modifiers:** Other words which describe the characters (e.g. "Catherine is still *alive*")

The classification of tokens into one of these categories is based on their Stanford Typed Dependencies [14], assigned using the Stanford CoreNLP natural language analysis toolkit[1].

In the *Dirichlet Persona Model* (Figure 2.1), these words, along with their classification into the 3 categories described above are the only data used to infer the persona of a character. In addition to this model, they propose an augmented one (*Persona Regression*), which also takes some metadata into account, namely the age and gender of the characters (based on the gender and date of birth of the actor portraying that character) and the genre of the film.

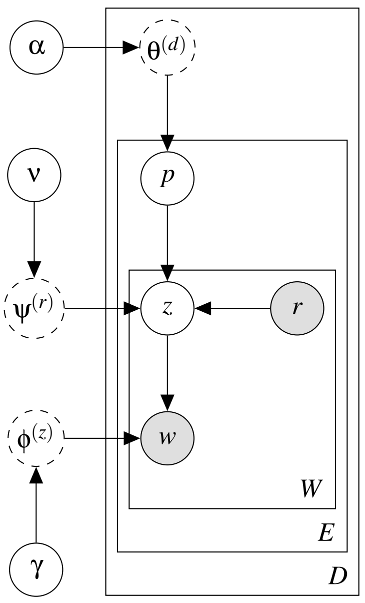


Figure 2.1: Graphical model of the *Dirichlet Persona Model* presented in [5]. Figure adapted from the one in the paper. As can be seen when comparing with Figure 4.5 below, it extends the LDA model.

Graphical models are useful for representing dependence relationships between random variables and allowing inference to be modelled in terms of graph operations [7]. In the notation used in Figure 2.1, nodes are random variables, directed edges represent dependence and a *plate* (rectangle) shows that a certain random variable is replicated a number of times, which is shown in the bottom-right corner of the plate. Nodes that are shaded represent observed variables, while dashed nodes signify that they are *soft*

[1]www.nlp.stanford.edu/software/corenlp.shtml

evidence for the variable which depends on them, meaning that the probability mass is divided among different states. This is in contrast to *hard evidence*, where all the probability mass rests in one state. Such a relationship defines a *soft-* or *hard-clustering* of the data. In the former case, data points can be assigned to more than one cluster (e.g. words can belong to more than one topic), while in the latter each point is assigned to a single cluster (e.g. characters have a single persona).

In the generative model shown above, the words w and their role r (*agent*, *patient*, *modifier*) are the observed variables. W is the total number of words. Each word is assigned a topic z. Topics ϕ are multinomial distributions over words in the vocabulary, with a Dirichlet prior, parameterised by γ. A persona p is then defined as a set of 3 multinomial distributions ψ over the topics, also with a Dirichlet prior, each parameterised by a ν_r. At the document (film) level, characters' personas are also drawn from a multinomial distribution θ, with a Dirichlet prior, parameterised by α. The authors use collapsed Gibbs sampling [19] to sample the latent topic z of each word and the persona p of each character. Performing this type of inference, while optimising all the hyperparameters on a large data set, is computationally expensive. This is one of the reasons for which we sought to implement and test a simplified version of this model, described in Section 4.2, which is based directly on an implementation of Latent Dirichlet Allocation.

2.1.2 Social networks in narrative fiction

Another strand of work in this field is concerned with the interactions between characters and how they contribute to defining the story, rather than with their individual personalities. When analysing narratives and the characters that make up the story, we can think of representations that would capture their interactions and the ways in which they affect the story progression. An intuitive representation is that of a social network: a graph in which nodes are used to denote characters and edges represent some kind of interaction between them. Different variations of this model have been proposed.

Extracting and analysing social networks of characters from works of fiction has already proven valuable for a number of applications. In the context of novels, such an approach can be used to answer literary theory questions by enabling the analysis of a large sample of writings, rather then the small subset of well-known works which are usually examined by literary scholars. In their paper [16], Elson et al. propose a way in which social networks can be extracted from English novels, for the purpose of answering disputed questions about the correlation between different features of narratives (e.g. first- or third-person perspective, urban or rural setting) and character interactions (e.g. cohesion of the social network). Their approach uses the number and length of quoted speech acts which are inferred to be directed from one character to another to assign a weight to the relationship between the characters.

Others have tried to extend the social network representation beyond quoted speech, also in the context of literary fiction. Notably, Agarwal et al. introduced a wider concept of *social events* [2], meant to include any "deliberate" interaction not captured by direct speech (e.g., using the notation from the paper with braces for characters and brackets for social events, "{He} [was looking] at {Tomas} and [trying to smile]."[2]). The authors proposed a method for automatically extracting social networks from texts and evaluated their result by comparing their extracted social network from "The Adventures of Alice in Wonderland" with a gold-standard annotated network, using standard network measures such as degree centrality, betweenness centrality and graph density [1].

2.1.3 Sentiment analysis

Sentiment analysis is an active area of research which generally aims to reliably detect the subjective feeling or opinion expressed by someone, usually in writing, towards an idea or topic, a product or another person [30]. In the wake of the "big data revolution" [24], which generated a substantial interest for many organisations to collect and analyse customer data in order to guide their strategies, sentiment analysis (in this context, often called *opinion mining* [27]) became a key technique for gauging the interest and reactions of the public automatically, on an unprecedented scale. One of the most common sentiment analysis tasks, especially important for industrial applications, is detecting *sentiment polarity*: deciding whether the sentiment expressed (e.g. in a product review, a tweet, a blog post) is positive or negative.

Success in any sentiment analysis task is subject to overcoming many challenges, some of them related to limitations in state-of-the-art natural language processing tools in general, but others specific to the domain. One major obstacle, which seems to be inherent to dealing with subjectivity, is the fact that the accuracy of any such system is limited by the rate of agreement between human judgements. Different studies found that humans will agree on the sentiment polarity of a sentence between 75% and 90% of the time [17, 32]. Another difficulty in building automatic sentiment analysis systems is that the way in which people express feelings can be different depending on the context (e.g. product reviews and film reviews) [30], meaning that systems which perform well in one domain will not necessarily generalise. Sentiment analysis also depends on reliably performing generic NLP tasks, such as named entity recognition, co-reference resolution and dependency parsing[3], which are still not completely solved problems. It also struggles with ambiguous uses of language, such as sarcasm or words which change their meaning depending on the context in which they are used.

2.1.3.1 Approaches to automatic sentiment analysis

There are a number of general-purpose and domain-specific sentiment analysis systems in wide use today.

[2]Kundera, Milan. *The Unbearable Lightness of Being*. Translated by Richmond Hoxie. London: Faber & Faber, 1984.

[3]These concepts will be explained in Chapter 4, when discussing the implementation of the text processing pipeline for the project.

Sentiment analysis is usually broken down into different levels of granularity and treated differently in each case [22]:

- **Phrase level** Many sentiment analysis systems rely on gold-standard corpora which annotate large numbers of words and phrases with a sentiment valence.
- **Sentence level** In the most basic form, the sentiment of a sentence can be computed by summing the sentiment valences of its constituents. However, the best sentiment analysis models will exploit the structure of sentences in more sophisticated ways [29].
- **Document level** Assessing the sentiment of a document as a whole (i.e. a text composed of multiple sentences, such as a blog post) assumes that it expresses a single, coherent point of view about a single topic. Therefore, it has limited applicability, since this assumption often does not hold in practice.

2.1.3.2 The VADER sentiment analysis system

For this project, we used VADER (*Valence Aware Dictionary for sEntiment Reasoning*) [20], a rule-based sentiment analysis system, geared towards analysing social media content. It is implemented in Python and it is available as an open-source project under the MIT licence[4].

We chose to use VADER because it is a simple model which uses a combination of features (a gold-standard sentiment valence lexicon and learned composition rules that correlate with sentiment expression in different contexts) to determine the overall sentiment of a sentence. It achieves state-of-the-art results, comparable to more sophisticated and expensive models such as the recursive deep model of Socher et. al.[29]. Its accuracy is significantly better than those which compute the sentence-level sentiment as the sum of the sentiment of its parts.

2.1.4 Sentiment analysis in the study of fiction

Among the first efforts in using sentiment analysis on literary fiction is the work of Alm et al. [3], on analysing children's fairy tales. The purpose of their research was to improve text-to-speech synthesis, by taking emotional cues from the story into account, similar to how a person would read a story to a child. They also explore the role of "emotion trajectories", which model the distribution of emotions over the course of the narrative, in the evolution of stories written for children [4].

Building on this previous work and on that of Elson et al. [16] (discussed above), Elsner uses sentiment trajectories of individual characters and a representation of relationships between them in order to compare the high-level plot structure of novels, using a kernel to do one-to-one comparisons of characters between different novels. [15].

[4]www.pypi.python.org/pypi/vaderSentiment

Most of the previous literature in this area focuses on literary fiction rather than films. An example of recent work that employs, among other techniques, social network analysis and sentiment analysis is Gorinski and Lapata's paper [18] on film scene extraction for the purpose of automatic film summarisation. For this work, they compiled the ScriptBase corpus, on which this project is also based. They extract the social network of a film using the number of character-to-character interactions as edge weights, which is also the approach used in this project. When constructing chains of scenes, they take into account a "diversity" metric, which is computed in terms of the difference, from one scene to the next, in the set of participating characters and the overall sentiment of each scene. In this project, however, we are looking at character-to-character sentiment rather than in scenes as a whole.

This idea has been previously explored, recently by Nalisnick et al. [25], in their paper on character-to-character sentiment analysis in Shakespeare's plays. They exploit the semi-structured format of theatrical plays and introduce a simple set of assumptions to identify the speaker and listener of each line of dialogue. For this project, we adapted their approach in order to fit film scripts rather than plays in constructing our sentiment trajectories. A more detailed discussion of their work follows in Section 4.3, where we explain the design and implementation of these trajectories.

2.2 Data sets

We have mainly used two data sets to support the work done for this project. They have both been compiled as part of previous research efforts focused on computationally analysing films.

2.2.1 The CMU Movie Summary Corpus

A collection of 42,306 Wikipedia plot summaries of films, along with metadata extracted from Freebase. The summaries are processed using the *Stanford CoreNLP* tools, specifically tagging, parsing, named entity recognition and coreference resolution. Words of interest are also annotated with WordNet supersenses. Figure 2.2 shows the beginning of the plot summary for *The Silence of the Lambs*.

Wikipedia's guidelines for editors writing plot summaries state that it should be "an overview of the film's main events, [avoiding] minutiae like dialogue, scene-by-scene breakdowns, individual jokes, and technical detail."[5] More details on how the plot summaries in the CMU Move Summary Corpus are processed and annotated can be found in Section 4.2.1, which discusses our use of the corpus. Appendix A provides a step-by-step example to illustrate this process.

[5] www.en.wikipedia.org/wiki/Wikipedia:How_to_write_a_plot_summary

```
Clarice Starling  is pulled from her training at the FBI Academy
at Quantico, Virginia, by Jack Crawford  of the Bureau's Behavioral
Science Unit. He assigns her to interview Hannibal Lecter, a
former psychiatrist and incarcerated cannibalistic serial killer,
whose insight might prove useful in the pursuit of a serial killer
nicknamed "Buffalo Bill", who skins his female victims' corpses.

Starling travels to the Baltimore State Hospital for the Criminally
Insane, where she is led by Frederick Chilton (Anthony Heald) to
Lecter's solitary quarters. Although initially pleasant and
courteous, Lecter grows impatient with Starling's attempts at
"dissecting" him and rebuffs her.
```

Figure 2.2: Fragment of the Wikipedia plot summary for *The Silence of the Lambs* (1994)

2.2.2 ScriptBase

ScriptBase [18] is a corpus of 1,276 film scripts, along with IMDB and Wikipedia metadata, including cast list, (unprocessed) plot summaries and taglines. Of these scripts, 127 have been similarly processed using the *Stanford CoreNLP* pipeline and standardized as XML files following a set schema.

The 127 processed films were the staring point for us to construct the character graphs. Of these films, 96 had a correspondent in the CMU Movie Summary Corpus and we were able to construct a persona model for at least one of the characters in 86 of these films. Thus, our core data set was constructed based on these 86 films. More details on the corpus and constructing the data set follow in Chapter 4. Figure 2.3 shows an excerpt from the script of *The Silence of the Lambs*.

```
NEW ANGLE - REVEALS CLARICE

now wearing a more feminine skirt suit. Hair neatly coiled,
elegant shoulder bag, briefcase. He has rudely left her
standing.

            CHILTON
Will you be in Baltimore overnight...?
Because this can be quite a fun town,
if you have the right guide.

Clarice tries, unsuccessfully, to hide her distaste for him.

            CLARICE
I’m sure it’s a great town, Dr.
Chilton, but my instructions are to
talk to Lecter and report back this
afternoon.

            CHILTON
     (pause, sourly)
I see.
     (beat)
Let’s make this quick, then. I’m
busy.

                                                  CUT TO:

INT. ASYLUM CORRIDOR - UPPER FLOOR - DAY

Clarice flinches as a heavy steel gate CLANGS shut behind
her, the bolt shooting home. Chilton walks ahead of her.
```

Figure 2.3: Fragment of the script for *The Silence of the Lambs* (1994)

Chapter 3

Aims and hypotheses

The aims of this project were to find and experiment with different representations of films and to identify what kinds of features can be captured by these representations. A big part of the work done for the project was directed at probing these possibilities, by constructing variations of the character graph representation and measuring their effectiveness using a number of prediction tasks. Inevitably, not all of them produced significant results or were even practical to test. Along with our main hypothesis, we will present some of the work which did not yield any fruit and attempt to discuss the underlying reasons.

3.1 Inter-character sentiment

In one sentence, the main hypothesis is that the character graph representation introduced in this project can be used to predict inter-character sentiment. As discussed in Chapter 2, there have been previous attempts to link character social networks and some form of sentiment analysis. By using character personas and a measure of the strength of two characters' relationship to predict aspects of the sentiment in speech acts directed from one character to the other, we can bring together two distinct aspects of film narratives and show that they correlate. On the level of the text resources used, we link the raw dialogue of a film (from the film scripts) and the human-produced plot summaries. On the narrative level, we correlate the *actions* of characters (as described in the plot summaries) with their utterances. Success in this task could potentially be useful in other NLP problems, such as automatic summarisation and generation of narratives. Details of how we constructed the character graphs follow in Chapter 4 on design and implementation, while the full discussion of results obtained is the subject of Chapter 5.

3.2 Alternative hypotheses

This section presents two of the hypotheses which were strongly considered and explored and discusses why they were not feasible to test. More details on the design and implementation of the code written to investigate them are included in the next chapter (Section 4.5).

3.2.1 TV Tropes

TV Tropes[1] is an online public wiki maintaining a collection of "tropes", such as plot devices, dialogue commonalities, motifs and other conventions or stereotypes that are widely used in film and television series. More recently it has also grown to include literature, video games and other media. Apart from collecting the tropes themselves, TV Tropes maintains pages for numerous individual films and other works of fiction which include a list of the tropes that they feature.

TV Tropes have been used in past research efforts, most notably by Bamman et al. [5], as part of one of the evaluation measures for their film persona models introduced in Section 2.1.1. They selected 72 tropes which correspond to character types (e.g. *The Hardboiled Detective*) and manually annotated 501 characters. They attempted to recover gold-standard clusters of characters as determined by their trope assignment and used an information-theoretic measure to assess the discrepancy between these and the latent persona clusters.

We considered the possibility of evaluating our character graph representations on the basis of being able to capture different kinds of relationships between 2 or more characters (e.g. one of the *Triang Relations*[2] or *Four-Man Band*). For this purpose, we have implemented a scraper in order to collect the tropes associated with the films in our data set. Table 3.1 shows the most common 20 tropes and the number of films which feature them.

1	Shout-Out	54	11	Berserk Button	22
2	Deadpan Snarker	31	12	Cluster F-Bomb	19
3	Lampshade Hanging	27	13	Butt Monkey	18
4	Chekhov's Gun	27	14	Large Ham	18
5	Running Gag	26	15	Badass	18
6	Oh, Crap	25	16	Ax-Crazy	18
7	Brick Joke	24	17	Title Drop	17
8	Precision F-Strike	23	18	Word Of God	17
9	Foreshadowing	23	19	Meaningful Name	16
10	Black Comedy	22	20	Even Evil Has Standards	16

Table 3.1: The 20 most frequent TV Tropes and the number of films which feature them (out of the 86 films in the collection)

[1]www.tvtropes.org

[2]www.tvtropes.org/pmwiki/pmwiki.php/Main/TriangRelations

None of these 20 tropes describe character relationships in any way. Most of them capture plot devices, features of speech or stylistic choices. Lacking a significant number of films (all other tropes are featured in less than 16 films) which include any kind of stereotypical and easily identifiable character relationships, we were not able to use TV Tropes to evaluate the character graph representation.

3.2.2 Film similarity

Another evaluation measure considered for assessing the usefulness of character graphs was the ability to compare them in order to predict film similarity, as judged by human raters. In principle, we would expect to obtain at least some degree of accuracy, although film similarity can depend on contextual information beyond the dialogue or narrative of a film. Of course, most recommender systems determine film similarity based on users' reviews and activity and on metadata (e.g. genre, year, cast, crew) rather than the content of the films themselves [28].

Some of the reasons for which we were not able to test this hypothesis are the small size of our film corpus and the fact that the films were quite diverse, resulting in very few similarity relations between them. Furthermore, graph comparison is a difficult problem, highly specific to the domain. There are no universal solutions, but some work has been done on comparing graphs of social networks [23]. A fuller discussion of our attempts to investigate this hypothesis follows in the chapter on design and implementation (Section 4.5.2). This is because it relies on details and explanations given in subsequent sections.

Chapter 4

Design and implementation

This chapter details the components of the software pipeline implemented as part of the project to acquire and process data, construct character graphs and carry out experiments. The system was primarily programmed using Python 2.7.6. The language was chosen for the terse and flexible style of programming that it enables and for its adoption in the NLP community, having a large ecosystem of libraries and tools developed around it. Some parts concerned with parsing script files and gathering film similarity judgements from Rotten Tomatoes were implemented using F# 3.1. This choice was made on the basis of F#'s smooth integration with external data sources and ease of working with schematised data.

The structure of the Python project is shown in Figure 4.1. Directories and source files in the root directory correspond to the major areas of implementation, which can roughly be categorised as follows (by directory/file):

- **`topics`**: Extract and filter plot summary data, run Latent Dirichlet Allocation and pickle the resulting topic distributions
- `personas.py`: Define a `Persona` class, compute the persona for each character and pickle them
- `graphs.py`: Construct character graphs and implement various functions for working with them (e.g. filter nodes/edges)
- `film.py`: Define a `Film` class, which stores the character graphs and matches them with various pieces of metadata (e.g. title, year, names of characters)
- **`sentiment`**: Extract speech acts from processed film scripts and construct sentiment trajectories between pairs of characters
- **`prediction`**: Implements various predictors and tools for feature engineering and running repeated experiments with different configurations
- **`utils`**: Processing raw data, pickling and other utility functions required throughout the project

The names of the rest of the source files give a good indication of what functionality they implement and should serve as a guide for exploring the source code.

```
python
├── topics
│   ├── acquire_vocab.py
│   ├── filter_genres.py
│   └── model_topics.py
├── sentiment
│   └── trajectories.py
├── prediction
│   ├── decision_tree.py
│   ├── random_forest.py
│   ├── svm.py
│   ├── lin_regression.py
│   ├── ridge_regression.py
│   ├── regression_tree.py
│   ├── svm_regression.py
│   ├── features.py
│   └── predict.py
├── utils
│   ├── filter_summaries.py
│   ├── process_summaries.py
│   ├── process_scripts.py
│   ├── store_sentiment.py
│   └── misc.py
├── films.py
├── personas.py
├── graphs.py
├── compare.py
└── run_topic_model.py
```

Figure 4.1: General structure of the Python code written for the project

The code for constructing the Rotten Tomatoes film similarity graph was written in F#. Its structure is outlined in Figure 4.2.

```
rtgraph
├── Graph.fs
├── ApiCalls.fs
├── ToGEXF.fs
└── Main.fs
```

Figure 4.2: General structure of the F# code written for the project

4.1 The Character Graph

In order to construct the character graph for a film we need to find a meaningful representation of the relationships between characters (the nodes in our graph). For simplicity, we can start by trying to find a numerical measure, which would correspond to edge weights in an undirected weighted graph. The measure we have used in this project is the number of scene co-occurrences between two characters. Thus, the weight of an edge between two characters represents the number of scenes in which those characters appear together. As a consequence, characters which never interact in the film will be disconnected in the graph.

Apart from simplicity, choosing this measure can enable us to reliably discover the main characters of a film using a graph centrality measure. However, scene co-occurrence does not give any insights into the *nature* of the interaction between characters, but a high co-occurrence is likely to indicate that a relationship is important for the development of the story. Our approach is similar to that of Gorinski et al. [18], although they go beyond degree centrality for identifying main characters.

4.1.1 Processing ScriptBase screenplays

To ease the task of extracting the number of character to character scene co-occurrences, we restricted the set of films to the 127 which have already been processed in ScriptBase. A processed script is an XML file which, apart from the standard CoreNLP annotations, also places every sentence inside a numbered `scene` tag. It also distinguishes between stage directions and speech, annotating the name of the speaker(s) in the latter case.

Thus, it is straightforward to parse the XML document and, for each pair of characters, record the number of scenes in which they both have at least one line of speech. Note that although this approach ignores characters which might be present in a scene but do not speak, in practice we do not expect this to significantly influence any of the results.

4.1.2 Constructing character graphs

For generating and storing character graphs, we have chosen the GEXF format[1], a standard XML format for describing general network structures. As described above, we extract all the character names from the script and use them as the set of nodes in the graph. We add a weighted edge for all the recorded interactions between characters. As a result, each of the 127 films in the data set will be represented with a GEXF file.

Once the character graphs are constructed, the nodes can be annotated with the persona information. Note that although the character graphs are likely to contain a node for every character in the plot (a small variation can occur due to faults in the preprocessing of scripts), personas will be computed for 2-5 main characters. Again, due to the

[1]The GEXF file format, *www.gexf.net*

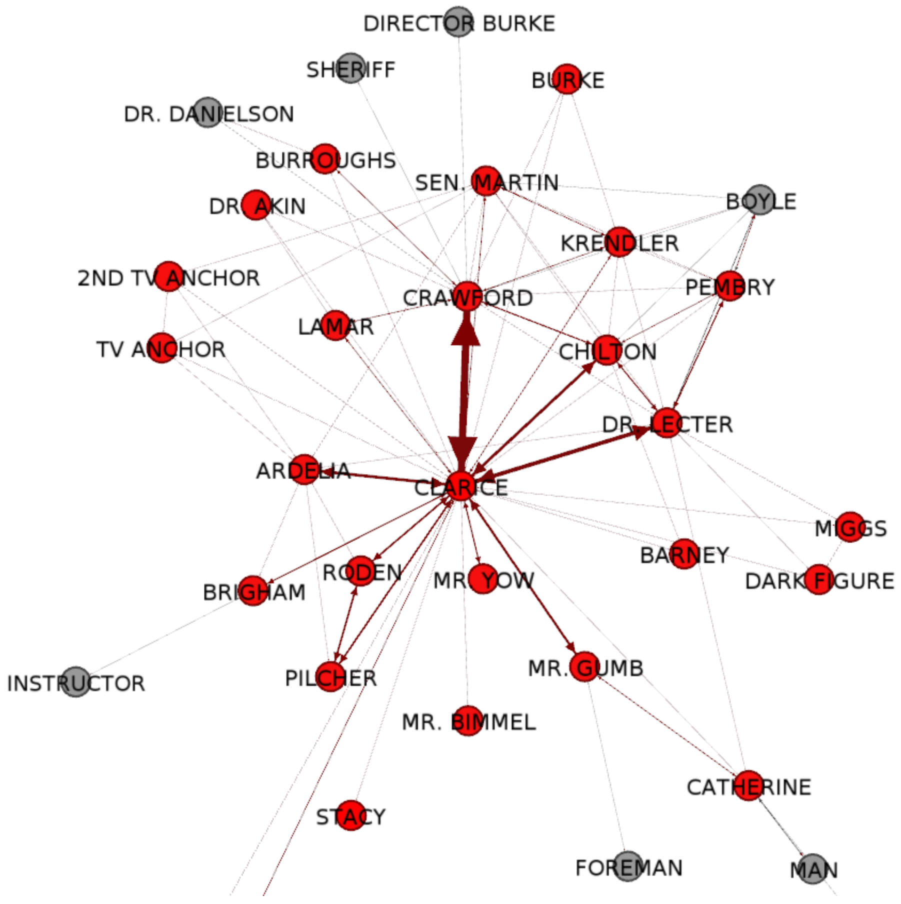


Figure 4.3: The character graph of *The Silence of the Lambs* (1991), centred around the protagonist, Clarice Starling. Red vertices are those adjacent to Clarice (characters with which she interacts). Edge width is proportional to the number of scene co-occurrences between two characters.

limitations of the standard NLP tools (most notably coreference resolution), this number can be lower, sometimes even 0. To work with annotated graphs in our project, we have used the Python `networkx` library[2], which can be used for complex graph modelling and implements many useful graph algorithms and network measures.

4.2 The Persona Model

We implemented a simplified version of the Dirichlet Persona Model introduced by Bamman et al. [5] and described in Section 2.1.1.

4.2.1 Processing the CMU script summaries

The CMU corpus provides both the raw plot summaries and a file containing all the processed summaries, which can be used as input for the pipeline described in their paper. We have designed ours to work with the same input format, enabling us to use the data from this corpus without any additional preprocessing, apart from filtering out the films for which we have not constructed a character graph. This includes the majority of films in the corpus. Out of the 127 films from ScriptBase, 96 also have a processed plot summary. The films in this intersection are the starting point in our processing pipeline.

```
30006 /m/0k6g8c:t0.1.0:verb.contact:pull:a:agent /m/0k6g8p:t2.2.0
:verb.stative:lead:a:agent /m/0k6g86:t3.0.0:verb.change:grow:a:ns
ubj /m/0k6g86:t3.2.0:verb.communication:rebuff:a:nsubj (...)
```

Figure 4.4: An entry (corresponding to a film) in the input file used for the work described in [5], which we filtered and used as input for this project.

The filtered film data file contains, for each film and each character mentioned in the plot summary, a list of words tagged as being either *agent*, *patient* or *modifier* words (as detailed in Section 2.1.1). These words are selected into one of these types on the basis of their Stanford typed dependencies, as determined using the CoreNLP parser. Each of the three types described above corresponds to a list of dependencies and all tokens tagged with those dependencies will be assigned a type. In the final input file, in addition to the word lemma, its type (agent, patient, modifier) and its dependency tag, the word tuples for each character also contain the part of speech and the WordNet supersense (which are broad semantic categories, such as *noun.shape* or *verb.motion*). As shown in Figure 4.4, each line starts with the identifier of the film, followed by a list of (colon-separated) tuples, containing this data. The first item in the tuple will either be a Freebase[3] identifier or have the format `eXX` (where `XX` stands for a 2 digit

[2]www.networkx.github.io

[3]Freebase (www.freebase.com) is a Google-owned, community-driven repository of structured data, containing numerous topics, entities and connections between them. Each entity has a unique ID.

number) for characters and other entities which could not be matched with Freebase entities. The full preprocessing pipeline, with a sample sentence as a running example is outlined in Appendix A. Note that Bamman et al. provide both the processed data and the code required to produce it[4].

4.2.2 Constructing the persona model

In this project we implement a simplified version of the Dirichlet Persona Model. The original model was implemented as a generative model. In the paper, a persona is defined as a set of 3 distributions (corresponding to agent, patient and modifier words) over topics, which are in turn distributions over words, similar to topics in Latent Dirichlet Allocation.

We have opted to use an implementation of the standard Latent Dirichlet Allocation [8] algorithm, using the `lda 0.3.2` Python package[5] in order to derive the latent topics in our vocabulary and use a simple maximum likelihood estimation method to compute each character's 3 distributions over these latent topics.

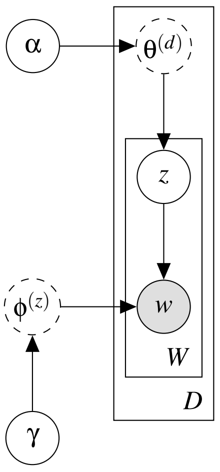


Figure 4.5: Graphical model for Latent Dirichlet Allocation (LDA)

One of the advantages of this frequentist approach is that the only parameter that we need to optimise in the model is the number of topics computed by the LDA algorithm. Since the number of films in our collection is small and largely restricted to two genres (thriller and comedy), we expect the number of meaningful topics to also be reasonably small. Therefore we have tried to manually vary this parameter and select the value which seems to intuitively give a complete coverage of topics, without having different topics which seem very close. We have used the highest probability words in each topic distribution as a guide and settled on having 10 topics (Figure 4.5).

Much like the Dirichlet Persona Model described in Section 2.1.1, LDA is a generative model which assumes that a document is created by first choosing the topics, and then sampling words from the topic distributions. Having these 10 topics distributions over

[4]www.ark.cs.cmu.edu/personas
[5]www.pypi.python.org/pypi/lda

```
1: leave meet reveal wife marry ask decide call woman want husband
2: father mother child son marry parent live brother daughter woman
3: kill shoot take man officer arrest escape kidnap detective lead
4: kill rescue escape save arrive bring reveal destroy vampire fly
5: tell be ask see say call friend talk show start mother invite
6: run get chase catch throw grab pull knock hit fall stop walk use
7: friend girl meet student play teacher end girlfriend school
8: find take follow driver travel join reach fly track locate sail
9: kill man shoot attack killer stab die dead confront attempt
10: boy win play have owner manager hero uncle narrator stooge
```

Figure 4.6: Most representative words for each of the 10 topics learned by LDA on all the comedies and thrillers in the CMU Movie Summary Corpus, ranked by probability.

words in the vocabulary, we construct, for each character, a set of 3 distributions over these topics, by multiplying the number of different occurrences of a certain word lemma with the probability of that lemma in a certain topic, then normalising. It is then easy to compare the persona of two different characters, by using any distance measure, such as Euclidean distance.

4.3 Sentiment trajectories

The sentiment trajectory of a film is the ordered sequence of speech acts, in which one (and only one) character (*the speaker*) is assumed to speak to one (and only one) other character (*the receiver*), which is different than the speaker. The sentiment of each line of dialogue is independently computed, using VADER. Each point in the sentiment trajectory is then a tuple of *(speaker, receiver, sentiment)*, which we will call a *sentiment event*.

The ScriptBase corpus includes sentiment valence annotations for each word in a script, using the AFINN-96 sentiment corpus [26], which assigns a value between -5 and +5 to 1468 English words and phrases. The sentiment of a text fragment can then be computed by summing the values of its constituents. We chose to use the VADER corpus instead because it also takes the structure of sentences into account, instead of relying exclusively on individual sentiment valences.

In order to construct a sentiment trajectory for each film in the data set, we traverse the XML script (described in Section 4.1.1), extract every scene and the names of the characters which take part and every line of dialogue, together with its speaker. The speaker of each utterance is marked in the script, but we need to identify a receiver. We will follow the assumptions introduced in [25]:

- For the first line of dialogue in the scene, assume the receiver is the next speaker.
- For all the following lines, assume the receiver is the previous speaker.

Although these assumptions (and those introduced in the first paragraph, restricting dialogue between exactly two distinct characters) are quite naïve, in practice they tend to hold reasonably well, especially since many scenes only feature 2 participants.

Another assumption that we make when using the sentiment trajectories as representations of two characters' relation is that the sentiment expressed in a speaker's utterance is directed towards the receiver. Of course, this might not always be the case. To remedy this, we could attempt to parse the typed dependencies from each sentence and only use the phrase-level sentiment from phrases which have a relevant dependency relations with the receiver (e.g. *nsubj*, *dobj*, *ref*, ...), coupled with coreference resolution to match characters with identifiers such as "you", "your", etc. However, this approach could result in far sparser sentiment trajectories. In practice, although a line of dialogue might not be directly referring to the other party, its sentiment can often be taken as an indicator of the speaker's relationship with the receiver. To minimise the impact of outliers resulting from either a failure of the sentiment analysis or the breaking of one of the assumptions introduced above, we will only use aggregate measures of the sentiment trajectory in our evaluation tasks, discussed in Chapter 5.

Once constructed, sentiment trajectories can be plotted as time series. For clarity, we will only include one character relationship per plot. Note, however, that relationships are directional, since the sentiment that each character expresses towards the other will differ. The plots shown here are illustrative of certain aspects related to film genre or the dynamic between the characters.

Figure 4.7 shows the sentiment trajectories of two thrillers. Both the main characters are allies in the story, but the protagonists of *Frozen River* have a very uneasy alliance, while those in *The Book of Eli* are sympathetic towards each other. Figure 4.8 shows the sentiment of the protagonists of two romantic comedies. One of them (*Midnight in Paris*) has a "happy ending", with the implication that the characters' relationship will continue, while the other one (*I Love You Phillip Morris*) ends on a sadder note, with the characters having a fight towards the end.

One thing to notice is that although the sentiment trajectories seem to capture the "attitude" that two characters exhibit towards each other, they are usually not enough to characterise the nature of their relationship at a more meaningful level. A positive sentiment will not be enough to distinguish between, for example, friendship, mentor-student or romantic relationships, while a negative sentiment could mean both that the characters are at odds with each other or that they are allies, but not on friendly terms. They do, however, give a gist of a film and can capture major plot points.

For example, the sentiment trajectories in Figure 4.9 illustrate the sentiment between the main characters of the romantic comedy *Chasing Amy*. In the film, Holden and Banky are two long-time friends. When the two meet Alyssa, she immediately piques Holden's interest. Although she only sees him as a friend, he pursues her throughout the film and they end up being together. Banky's distrust of Alyssa, however, becomes the main source of conflict between him and Holden. The 3 sentiment trajectories are consistent with our intuitive expectations of such a story and the feelings that the characters would express towards each other in these situations.

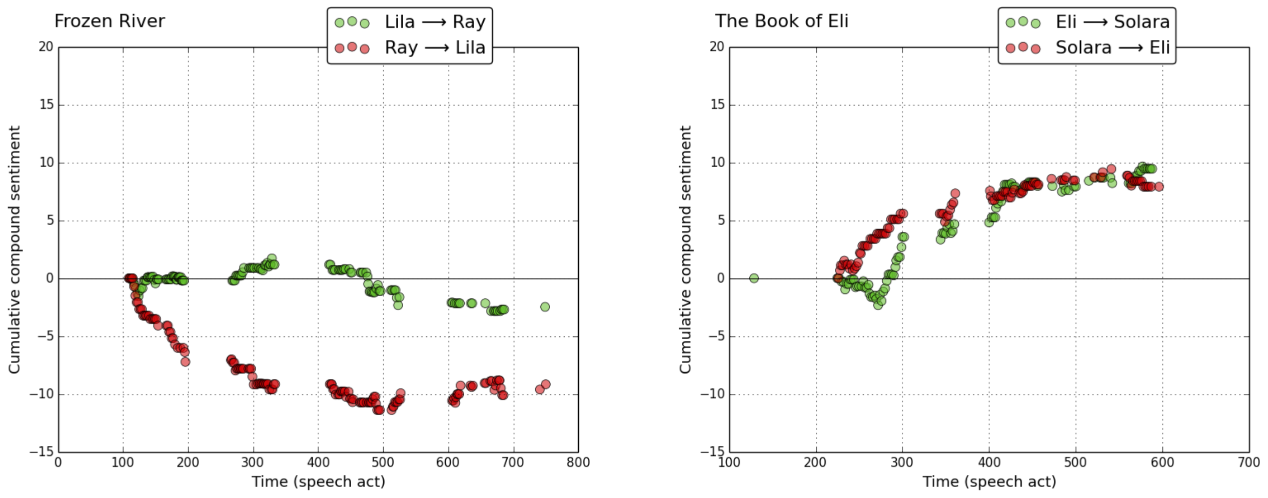


Figure 4.7: Sentiment trajectories of the two most central characters in the thrillers *Frozen River* (2008) and *The Book of Eli* (2010)

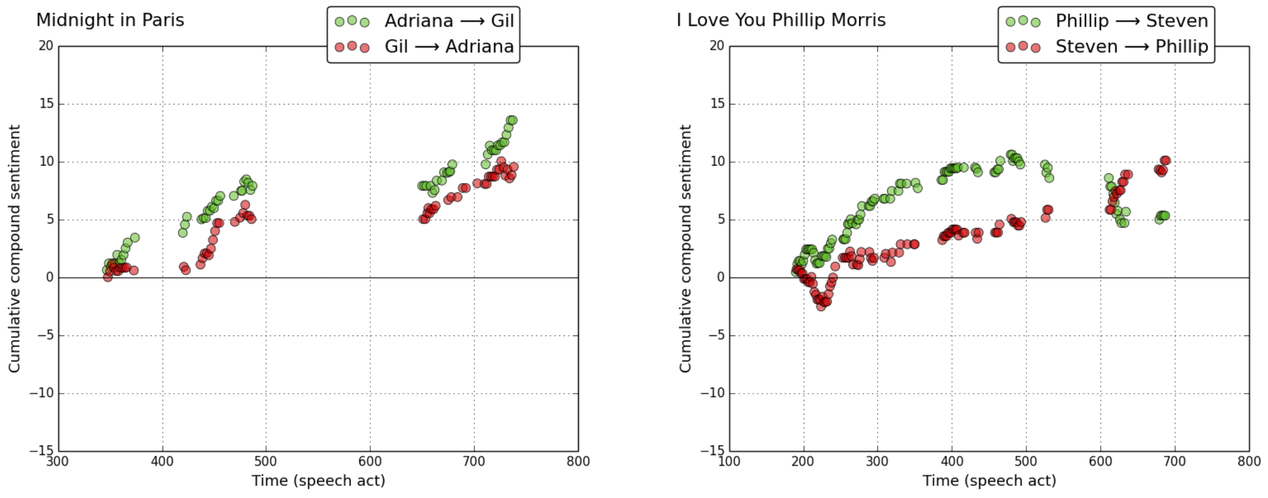


Figure 4.8: Sentiment trajectories of the two most central characters in the romantic comedies *Midnight in Paris* (2011) and *I Love You Phillip Morris* (2009)

In the examples given so far, the sentiment between the characters has largely been reciprocated (both characters expressing either positive or negative sentiment). We can also examine relationships for which that is not the case. The comedy *Analyze That* (left plot in Figure 4.10) is centred around the relationship between an organised crime boss and his psychiatrist. The humour of the films heavily relies on the contradictory and bizarre relationship between the two[6]. As it can be seen from the plot, the sentiment expressed by one character towards the other is almost always in opposition. The other film shown in Figure 4.10, *Gothika*, features two psychiatrists who are colleagues and friends. Miranda starts having paranormal experiences and is admitted in

[6]This claim is supported by different reviews, for example one for the film *Analyze This* (1999), of which *Analyze That* is a sequel, by film critic Roger Ebert (www.rogerebert.com/reviews/analyze-this-1999).

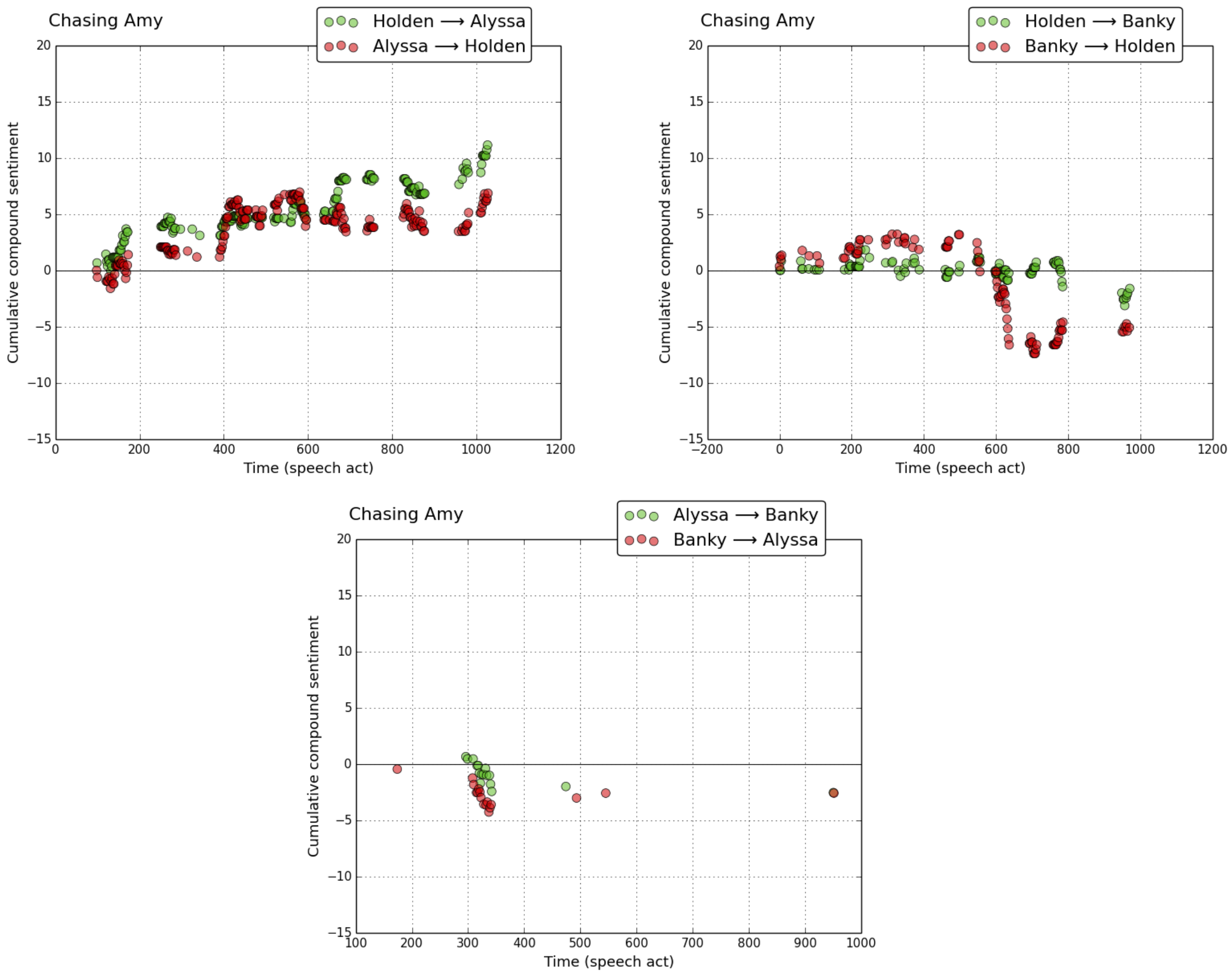


Figure 4.9: Sentiment trajectories of the three most central characters in the romantic comedy *Chasing Amy* (1997)

the hospital where they both work. Throughout the film, she tries to convince Pete that she is not mentally ill. As expected, Pete mostly expresses positive and later neutral sentiment as he tries to make Miranda aware of her situation, while Miranda is increasingly frustrated with Pete's lack of trust in her claims.

4.4 Testing the main hypothesis

In order to test the main hypothesis, a number of prediction tasks were considered, including both classification and regression problems. The tasks themselves and the results are fully detailed in Chapter 5. Here, we will outline the main machine learning techniques used, the reasons for choosing them and their implementation. The toolkit used for carrying out machine learning tasks was the *scikit-learn*[7] Python library. It was used for its implementation of the predictors described below, for training and

[7]www.scikit-learn.org/stable/

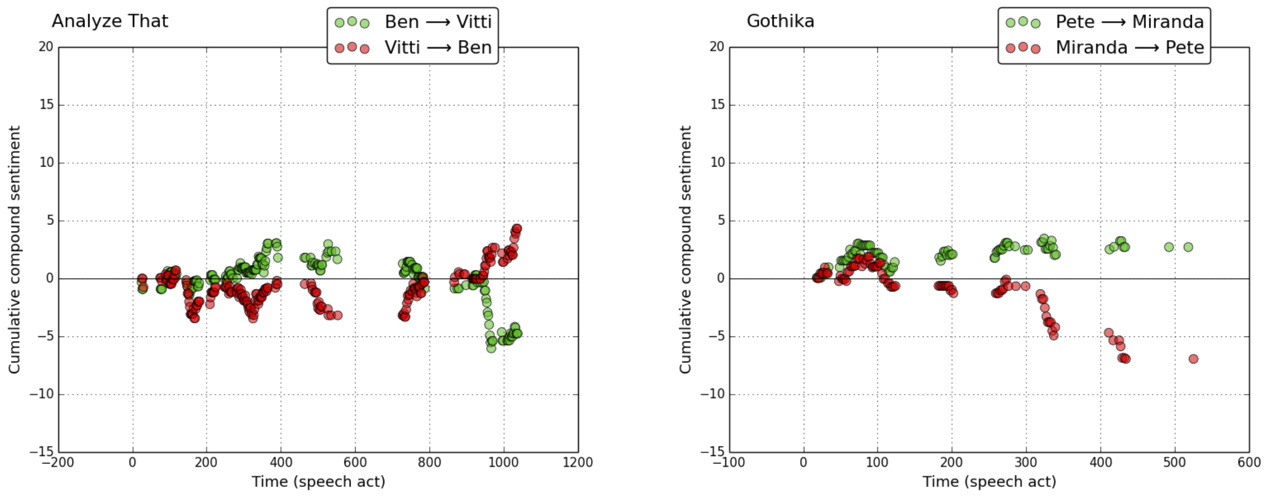


Figure 4.10: Sentiment trajectories of the two most central characters in the films *Analyze That* (2002) and *Gothika* (2003)

evaluating models using cross-validation and for generating confusion matrices. All models were evaluated on a 80% / 20% split between training and testing data, sampled randomly at each iteration.

4.4.1 Classification

We have used 3 common machine learning methods for our classification tasks. A common advantage of these methods is that they are (usually) computationally cheap to train and tend to produce good results without depending too much on optimising parameters. This enables our approach of exploring many different possibilities (detailed in Chapter 5) rather than focusing on a a small number of models and optimising their performance.

- **Decision trees** work by training a rule-based model, which repeatedly splits the data set in order to maximise a certain measure (in our case, we used both information gain and Gini impurity). They can cope with un-normalised data and are not sensitive to outliers. Another usability advantage is that trees can be visualised and their rules examined. On the other hand, they sometimes tend to overfit and produce oversized trees (in principle, every point can be classified correctly if each node corresponds to a data point). Also, they can only produce axis-aligned splits of the data (since every node is a decision of whether to follow a certain branch based on how a particular feature compares to a certain value), which might be limiting in some applications. We used the `DecisionTreeClassifier` class implemented in *scikit-learn*.
- **Random forests** [10] mitigate some of the shortcomings of decision trees. A random forest classifier will train a preset number of decision trees on different partitions of the data set and, for each new data point, output the most frequent prediction of the decision trees. This approach significantly amelio-

rates the decision trees' problems with overfitting, making random forest a viable and widely used predictors for many applications. However, it retains the decision trees' inability to generate non-axis-aligned splits. We used the `ensemble.RandomForestClassifier` class.

- **Support vector machines (SVMs)** [13] are a maximum-margin classifier, aiming to separate the data as cleanly as possible (creating the maximum margin between the class boundary and the closest data points on both sides). In addition to classifying linearly separable data, it can be used for nonlinear classification by using the "kernel trick", which can transform the problem space into a higher-dimensional one using a non-linear kernel function. In this project, we tried using the linear, polynomial, RBF (radial basis function) and sigmoid kernels, as implemented in *scikit-learn*, in the `svm` class. SVMs tend to produce reliable results, while avoiding overfitting, but they require more careful tuning than decision trees or random forests. Also, SVMs are by definition binary classifiers, meaning that in order to do multi-class predictions, a separate classifier needs to be trained for every pair of classes.

4.4.2 Regression

For regression, we have also used 4 common algorithms, 2 of which are also used in the classification tasks.

- **Linear regression** is a simple regression model which fits the parameters of each feature using ordinary least squares to minimise the differences between actual and expected output. It is computationally cheap and often useful without much tuning, but it can be too rigid since it is a linear model (note that it is linear in the parameters, meaning that nonlinear transformations can be applied to the features). It is also sensitive to outliers, which we expect to affect the results in our case. We used the `linear_model.LinearRegression` class.

- **Ridge regression** aims to improve on linear regression by enforcing a penalty on the size of the parameters, thus limiting the influence of outliers. However, the penalty parameter needs to be optimised. For this, we used the grid search method implemented in *scikit-learn* as part of the `linear_model.RidgeCV` class.

- **Decision tree regression** [11] is an adaptation of decision trees to handle continuous output values. All the points from the discussion on decision tree classification above still apply. We used the `tree.DecisionTreeRegressor` class.

- **Support vector regression** is an adaptation of SVMs to handle regression problems. There are several methods to do this implemented in *scikit-learn*, but we chose to use `svm.SVR`.

4.5 Testing alternative hypotheses

4.5.1 Scraping the TV Tropes website

The TV Tropes website does not offer an API for accessing its content. Therefore, in order to get the tropes for each film in the data set, we were required to implement a web scraper. A scraper is a piece of software which can automatically visit a large number of web pages, parse their DOM (Document Object Model) tree and extract relevant information. Scrapers can carry out more sophisticated tasks, but this was not required in our case.

We manually explored the structure of the TV Tropes website, which is quite consistent across the different pages, making our task easier. For working with the DOM inside Python, we used the Beautiful Soup library[8]. We also had to manually match some of the film titles which differ slightly between ScriptBase and TV Tropes (e.g. *Evil Dead II* and *Evil Dead 2*).

A list of the most frequent 20 tropes across the films in our data set is shown in Table 3.1, together with a discussion of why we could not exploit this data in our evaluation (Section 3.2.1). Triangular relationship of the kind shown in Figure 4.9 above (*Chasing Amy)* were the kinds of features we hoped to be able to predict. However, a very small number of films in our collection feature stereotypical 3-character relationships that would lend themselves to such a task.

4.5.2 Comparing character graphs

4.5.2.1 Direct matching based on centrality

As discussed in Section 3.2.2, graph comparison is a difficult problem which requires careful consideration of the domain. When exploring the film similarity hypothesis, we implemented a simple comparison measure that is based on matching characters from different graphs by their role in the social network. Using degree centrality, the nodes of each graph are sorted and only the first 10 nodes are kept. This parameter can in principle be experimentally adjusted, but we have set it to 10 because for every film in the collection, we could not compute a persona for any character that has rank below 10 by degree centrality. This is not surprising, since plot summaries will rarely mention characters of low importance (assuming importance in the social network translates as importance to the narrative). We have kept this filtering of nodes for all the other tasks described in the report.

Then, every pair of corresponding characters (by rank) in the two graphs is compared. Note that although a persona is a set of 3 distributions, not all characters will have a distribution defined for all 3 types (agent, patient, modifier), since the relevant word lemmas might not be present in the plot summary. Therefore, we compare the distributions which both characters have in common, by taking the norm of the difference

[8]www.crummy.com/software/BeautifulSoup

between the 2 vectors. The film similarity score is computed as the average of these differences: the lower the score, the more similar the graphs are deemed to be.

```
Star Trek V    - The Anniversary Party     (0.00167)
Ghost World    - I Love You Phillip Morris (0.00193)
Ghost World    - Copycat                   (0.00202)
Ghost World    - Clerks                    (0.00236)
Ghost World    - Miller's Crossing         (0.00257)
Ghost World    - Nick of Time              (0.00275)
Ghost World    - Sweeney Todd              (0.00338)
The Informant! - The Hangover              (0.00351)
Ghost World    - Gothika                   (0.00352)
Jaws           - Living in Oblivion        (0.00410)
```

Figure 4.11: The 10 most similar pairs of films in our collection of 96 thrillers and comedies, based on comparing personas matched by degree centrality rank

In these results, it is easy to see that the comedy *Ghost World* (2001) dominates the similarity pairings, being matched both with comedies and thrillers. Overall, the results do not seem to be very revealing. To understand why this might be the case, let us explore the example of *Ghost World*, by showing the top 10 characters and the top 3 topics for each character that has a persona defined. Each topic is described by the top 3 most probable words in that topic.

```
ENID
REBECCA [ M(join wife know)
          M(pursue criminal escape)
          M(girl son offer) ]
SEYMOUR [ P(tell take go)
          A(tell take go)
          A(meet discover leave) ]
ROBERTA [ A(give kill confront)
          A(girl son offer)
          M(girl son offer) ]
JOE
CUSTOMER
PAUL
GERROLD
JEROME
STEVEN
```

Figure 4.12: The 10 most central characters in *Ghost World* and their personas

One explanation for the dominance of *Ghost World* in the similarity results shown above is the fact that its characters seem to be typical of both romantic comedies (topics such as `(join wife know)` or `(meet discover leave)`) and thrillers (topics like `(give kill confront)`).

We suspect that one of the reasons for the weak results is the small number of films in our collection and, consequently, the small number of words in the vocabulary. This can lead to a topic clustering which does not have much discriminative power (list of topics shown in Figure 4.6). It might also be the case that, since not many personas are matched between films, these occasional matches are insufficient for obtaining meaningful results across the whole corpus.

4.5.3 Rotten Tomatoes similarity

In order to test the film similarity hypothesis, independent of the work described so far, we have also tried to construct a gold-standard collection of film similarity pairs, using data supplied by Rotten Tomatoes[9]. Their public API supports requests of the form ”What films have users deemed to be similar to this one?” and offers up to 5 results.

We have found that the results obtained by querying every film in the collection and discarding every returned film which is not part of the collection yields a very small number of pairs. Thus, we have expanded the similarity measure by constructing a graph in which films are nodes and edges represent similarity and we also allow films from outside the corpus as nodes. The similarity can then be measured as the minimum distance in the graph.

However, the limited number of films in our collection and the fact that they are quite diverse (i.e. it is difficult to find pairs for which human judgements indicate similarity) meant that we could not meaningfully assess whether character graphs can be used to predict general film similarity.

[9]www.rottentomatoes.com

Chapter 5

Results and discussion

From its onset, this project was exploratory in nature. Experimenting with various film representations and the kinds of features that they can reliably predict was one of the main goals. As detailed in Chapter 3, the main hypothesis we chose to explore is whether the character graph representation of films can be used to predict inter-character sentiment. The results presented in this chapter and the subsequent discussion will offer some insight into how this can be achieved. We will examine different ways in which persona information and character relationships can be combined. We ran a large number of experiments, exploring different transformations of the feature space and various machine learning techniques. This chapter will provide an overview of the types of prediction problems attempted, report interesting experimental results and attempt to discuss their significance.

5.1 Experimental setup and data transformations

The same experimental setup and basic data is used for all of the prediction tasks detailed below. The starting point is the set of 876 character-to-character relationships defined in our final collection of 86 films, described by the persona of each character and the weight of the edge between the nodes in the character graph. By considering relationships rather than films as our feature vectors we have increased the size of our data set ten-fold. Since our topic model has 10 topics and each persona is a set of 3 distributions over the latent topics, each feature vector will have size 61 ($2 \cdot 30$ persona variables $+1$ edge weight). Each vector will also have an associated label, depending on the prediction task, which will characterise some aspect of the sentiment between the two characters. The character relationships are directed, since we are attempting to capture the sentiment expressed by one character towards another. As a result, each pair of interacting characters will be represented by 2 feature vectors in the data set, with their persona distributions swapped.

5.1.1 Data transformations

Apart from using the the whole data set described above, we have also tried to explore ways in which the feature vectors can be transformed and data can be filtered in order to yield the most significant results. To this effect, we have designed and implemented the experimental setup in order to be able to answer questions such as:

1. Are all 3 distributions of the persona model (*agent*, *patient*, *modifier*) equally relevant?
2. Can we get better predictions is we only use characters that have a more central role in the story?
3. Should we use a uniform distribution for characters without persona information or should we exclude them altogether?
4. Is it just a small number of topics that mostly define the persona of a character or are all 10 topics useful?

And, naturally,

5. Does using the edge weight as a feature result in better predictions over simply using the persona model?

When running experiments using the system developed for this project, the user can choose to either explore a certain data space (e.g. *Make predictions using every combination of two persona distributions, rather than using all 3 of them*, which will result in 3 possible configurations), or to run the experiments on a predefined set of configurations. The next subsection explains what parameters can be specified when doing so[1].

5.1.1.1 Presentation of results

When presenting prediction results in the remainder of this chapter, they will be accompanied by the context in which they were obtained. The purpose of this section is to explain the meaning of the columns under the *Configuration* heading in the results tables:

- *Number of nodes (N):* The (maximum[2]) number of characters from each film that were used. These were selected based on their degree centrality in the character graph. Its value can either be *All* or a number from 2 to 5.
- *Filter personas: Yes* or *No* value, depending on whether nodes without persona information were filtered out, or their personas were filled in with uniform distributions.

[1]The code which facilitates this is found in the `main` method of `python/predictions/predict.py`

[2]The number could be less than the one shown in a configuration where nodes without a persona are filtered out

- *Personas used:* Shows which of the 3 distributions in a character persona were used. Can be any non-empty subset of $\{A,P,M\}$ (standing for *agent*, *patient* and *modifier*).
- *Top topics only (T):* This value can be either *All topics*, showing that the persona distributions were used unmodified, or a number n between 1 and 7, meaning that, for each distribution, the top n topics by probability were selected and their values in the feature vectors was changed to 1, while the rest were set to 0, effectively creating a *1-of-N* representation.
- *Edge used:* *Yes* or *No* value, showing whether the edge weight between two characters was added as a feature or not.

5.2 Predicting inter-character sentiment

In this section, we present the results of 3 prediction tasks: 2 classification tasks and 1 regression. The first subsection will also give more details on how the data set can be transformed and filtered, by showing results for the full data set and then gradually varying the parameters described above.

5.2.1 Predicting sentiment polarity

First, we consider the problem of predicting compound sentiment polarity, modelled as a 2-class classification task. As discussed in Section 4.4, the classifiers used are decision trees, random forests and SVMs. In the case of random forests, 20 classifiers were trained on each configuration and the best one was selected.

Sentiment polarity of a relationship is computed as the sign of the cumulative sentiment expressed by one character towards the other. All the events between the 2 characters are summed across the sentiment trajectory of a film. If the sum is negative then the relationship will be in class 0 (*negative*), otherwise, it will be in class 1 (*positive*).

For this task, the baseline we are comparing our predictors against is the proportion of positive relationships in the total number of character relationships, which is always the majority class. As can be seen in the results tables below, they usually significantly outnumber the negative sentiment relationships, making this the simplest realistic baseline to compare our predictors against (i.e. comparing to a model which classifies every data point as having class 1). Models performing better than this baseline are highlighted in the tables.

We will first try to use the entire data set of 876 character relationships, with all 3 persona distributions across all topics (Table 5.1). Decision trees do not manage to out-perform the baseline, while random forests and SVMs are narrowly more accurate than it, but the difference is never more than 4%. When using the data set in which characters with missing personas are not filtered out, but their personas are treated as uniform distributions over the latent topics, no predictor manages to beat the baseline by more than 1%. This is the case for all the subsequent prediction tasks, therefore we will skip those results. Another aspect to note is that for both random forests and SVMs, using the edge weight either improved or had no effect on the accuracy.

Configuration							
Classifier	No. nodes	Filter personas	Personas used	Top topics only	Edge used	Baseline	**Accuracy**
Decision tree	All	Yes	A, P, M	All topics	Yes	0.733	0.648
	All	Yes	A, P, M	All topics	No	0.733	0.670
	All	No	A, P, M	All topics	Yes	0.871	0.872
	All	No	A, P, M	All topics	No	0.871	0.871
Random forests	All	Yes	A, P, M	All topics	Yes	0.733	**0.773**
	All	Yes	A, P, M	All topics	No	0.733	**0.767**
	All	No	A, P, M	All topics	Yes	0.871	**0.878**
	All	No	A, P, M	All topics	No	0.871	**0.877**
SVMs	All	Yes	A, P, M	All topics	Yes	0.733	**0.761**
	All	Yes	A, P, M	All topics	No	0.733	**0.761**
	All	No	A, P, M	All topics	Yes	0.871	**0.879**
	All	No	A, P, M	All topics	No	0.871	**0.879**

Table 5.1: Sentiment polarity predictions obtained by using the entire dataset and the full persona representation.

We will now try to train the same classifiers with different configurations of the data. First, we will remove the restriction that all characters in a film need to be used. We expect this to increase accuracy because some of the minor characters have less accurate persona information since they are less likely to be mentioned in a plot summary. Furthermore, their sentiment trajectories will have significantly fewer events, limited by the number of on-screen interactions that they are involved in. We will explore the range between 2 and 5 as the maximum number of characters from each film. The nodes are selected according to their rank in the character graph, which is the reverse order of degree centrality.

Table 5.2 shows that the best results were obtained when limiting the number of characters to either 2 or 3. The results for all 3 classifiers are significantly better than those obtained when using all the characters.

Next, we will both vary the maximum number of characters in the film, as before, and let the persona include any subset of the 3 distributions. Results are shown in Table 5.3. Note that for random forests, a third row of results (highlighted) is included in order to compare the accuracy of either using or not using edge weights on the same configuration that yielded the best performance in the previous task. This shows that adding the M (*modifier*) distribution and using the edge weight have the same effect on the accuracy (an increase of 6.3%).

Finally, we will also allow the persona representations to either include all topics, or to only select the *n* most probable topics and ignore the rest. Results are shown in Table 5.4. Note that the highlighted row has been included in order to compare the accuracy of SVMs on the same number of maximum characters (same baseline) with

Configuration							
Classifier	No. nodes	Filter personas	Personas used	Top topics only	Edge used	Baseline	**Accuracy**
Decision tree	2	Yes	A, P, M	All topics	Yes	0.713	**0.750**
	2	Yes	A, P, M	All topics	No	0.713	**0.750**
Random forest	2	Yes	A, P, M	All topics	Yes	0.713	**0.875**
	3	Yes	A, P, M	All topics	No	0.699	**0.882**
SVM	3	Yes	A, P, M	All topics	Yes	0.699	**0.824**
	3	Yes	A, P, M	All topics	No	0.699	**0.824**

Table 5.2: Best sentiment polarity prediction accuracy obtained when varying the maximum number of characters for each film.

Configuration							
Classifier	No. nodes	Filter personas	Personas used	Top topics only	Edge used	Baseline	**Accuracy**
Decision tree	2	Yes	A, P, M	All topics	Yes	0.713	**0.750**
	2	Yes	A, P, M	All topics	No	0.713	**0.750**
Random forest	2	Yes	A	All topics	Yes	0.713	**0.938**
	2	Yes	A, M	All topics	No	0.713	**0.938**
	2	Yes	A	All topics	No	0.713	**0.875**
SVM	3	Yes	A, P, M	All topics	Yes	0.699	**0.824**
	3	Yes	A, P, M	All topics	No	0.699	**0.824**

Table 5.3: Best sentiment polarity prediction accuracy obtained when varying the maximum number of characters for each film and allowing the persona to include any subset of the 3 distributions it contains.

the best model from the previous task. The percentages in brackets indicate the change in accuracy between the current and the previous task, for cases where the best results are comparable (having the same baseline).

The accuracy of these models is generally better than with the previous configurations. It remains unchanged in the case of random forests, for which allowing the number of latent topics to vary did not result in an increase in accuracy. These are the best results obtained in the sentiment polarity classification tasks. For all the trained classifiers, they are significantly higher than the baseline, showing that inter-character sentiment polarity can be predicted reasonably accurate. Note that in the final configuration shown in Table 5.4, decision trees can achieve the same accuracy as random forests. Such a decision tree is pictured in Figure 5.2 (on page 52). It does not seem to drastically overfit, although some nodes do contain only 1 sample.

5.2.2 Predicting changes in sentiment

As can be seen in the graphs shown in Section 4.3, it appears that the way inter-character sentiment evolves intuitively conveys certain aspects about the nature of the narrative and the characters which take part (e.g. in a romantic film, both of the main characters tend to express positive sentiment throughout the story). This suggests an-

Configuration							
Classifier	No. nodes	Filter personas	Personas used	Top topics only	Edge used	Baseline	**Accuracy**
Decision tree	2	Yes	A	6	Yes	0.713	**0.813 (+6.3%)**
	2	Yes	A, M	3	No	0.713	**0.938**
Random forest	2	Yes	A	All topics	Yes	0.713	**0.938** (+0.0%)
	2	Yes	A, M	All topics	No	0.713	**0.938** (+0.0%)
SVM	2	Yes	A	3	Yes	0.713	**0.875**
	2	Yes	A	3	No	0.713	**0.875**
	3	Yes	A	1	–	0.699	**0.853 (+2.9%)**

Table 5.4: Best sentiment polarity prediction accuracy obtained when varying the maximum number of characters for each film, allowing the persona to include any subset of the 3 distributions it contains and varying the number of topics over which personas are distributed.

other prediction problem: can we predict how the inter-character sentiment varies over the course of the film? We model this task as a 3-class classification problem and divide the sentiment trajectory of a film in 2 halves. If, in a character relationship, the averaged sentiment expressed by the first character towards the second one in the first half of the film is smaller than the sentiment expressed in the second half by at least some threshold α, then their relationship is in class 0 (*increasing*). If the absolute difference between the 2 halves is smaller than α then it is in class 1 (*constant*). Otherwise, it is in class 3 (*decreasing*).

We have introduced the parameter α, which controls the number of data points in class 1. A value of $\alpha = 0$ will results in class 1 being empty. In setting its value, the goal will be to obtain a balanced split between the 3 classes, across all the values of N. The graph in Figure 5.1 shows the proportion of points in the majority class, across all the values of N used in the previous task, for 3 values of α.

In most (5 out of 7) cases, $\alpha = 0.07$ makes the lowest number of points to be in the majority class. Either increasing or decreasing α will generally increase this proportion. We will therefore use 0.07 when labelling the data used to train the classifiers. Their performance is shown in Table 5.5. The highest accuracy is again obtained using random forests and it is 0.867. Note that although setting α to 0.06 or 0.08 will result in a drop in performance, in both of those cases, the best accuracy still ranges between 0.75 and 0.80.

We have shown that predicting changes in inter-character sentiment is also feasible with our representation. Note that the baselines are notably smaller than in the previous task, since we sought to obtain roughly equally sized classes.

5.2.2.1 Comparison of results between the 2 classification tasks

When predicting changes in sentiment, for all 3 classifiers, the best accuracy is obtained when limiting the maximum number of nodes in a character graph (N) to 2. This is similar to the sentiment polarity classification task, in which the best results

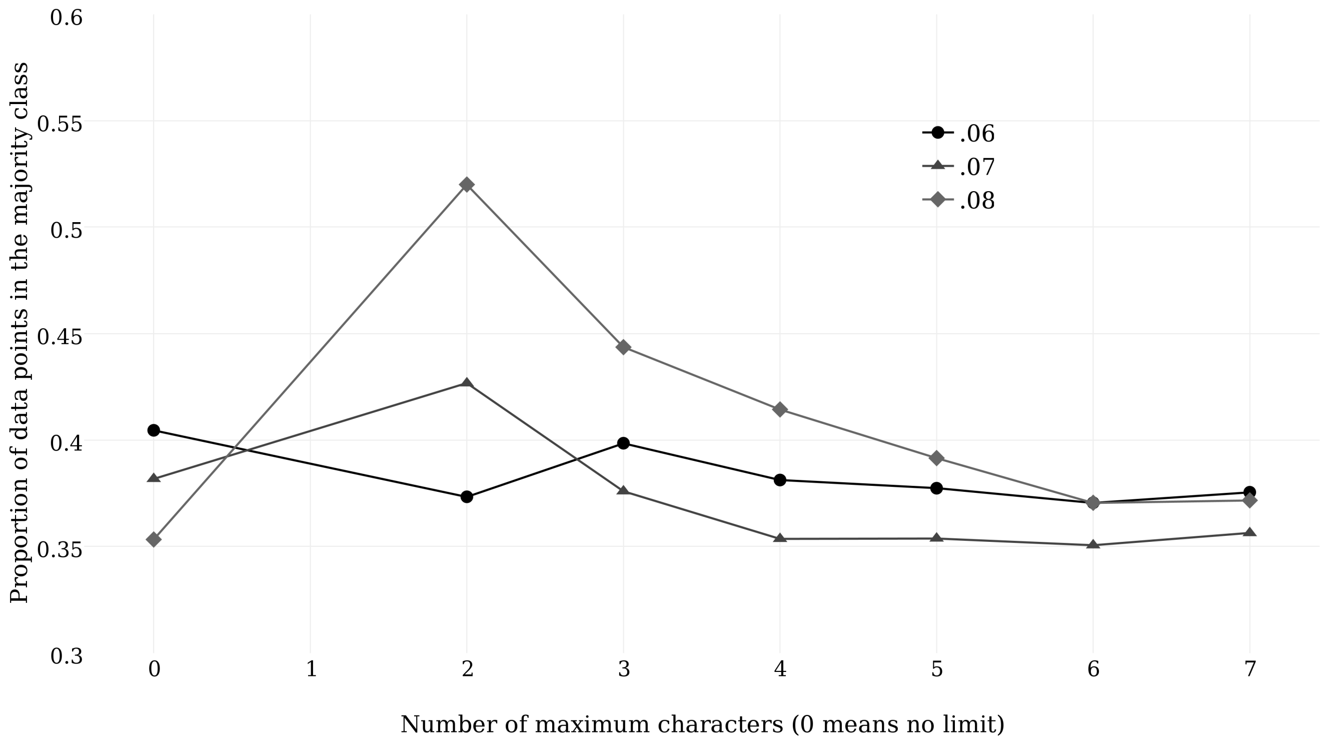


Figure 5.1: Proportion of data points in the majority class, over different values for the maximum number of characters (N), for 3 settings of the parameter α.

Configuration							
Classifier	No. nodes	Filter personas	Personas used	Top topics only	Edge used	Baseline	**Accuracy**
Decision tree	2	Yes	P, M	1	Yes	0.427	**0.600**
	2	Yes	P, M	1	No	0.427	**0.667**
Random forest	2	Yes	P	7	Yes	0.427	**0.800**
	2	Yes	A, P, M	6	No	0.427	**0.867**
SVM	2	Yes	P	4	Yes	0.427	**0.733**
	2	Yes	P	4	No	0.427	**0.733**

Table 5.5: Best sentiment variation prediction accuracy, with $\alpha = 0.07$.

were obtained when using an $N \in \{2,3\}$. We can offer a possible explanation for this bias in both of these cases. The sentiment trajectories of films will naturally include more events between the more central characters in the narrative, which can result in less erratic predictions, since the influence of outliers (e.g. from shortcoming in sentiment analysis) is limited.

Another common feature to note is the preference, in both tasks, to one of the 3 distributions in a character persona. While the choice of distributions varies for different classifiers, we can observe that in the sentiment polarity prediction task all the best-performing models were trained using (sometimes exclusively) the *agent* distribution.

This kind of preference is also observed in the sentiment variation task, but for the *patient* distribution. We can speculate that such a bias seems to show that actions that the characters do themselves will better predict the overall sentiment of a relationship, while actions which the characters experience are more likely predictors of how relationships evolve in the timeline of the story.

One more thing to note is that for both tasks and for all classifiers, using the edge weight (i.e. number of scene co-occurrences of 2 characters) as an extra feature in the feature vectors either has no effect on or decreases classification accuracy. This would seem to indicate that the persona model is enough to predict inter-character sentiment. However, as earlier discussed, the best results were obtained for values of $N \leq 3$, with nodes ranked by their degree centrality in the character graph. As a consequence, it is possible that using the edge weight only added noise to the data and the inter-character sentiment depends more on how central the characters are to the story than on an absolute measure of the frequency of their interactions. This claim can be supported by the fact that using the edge weight did not decrease accuracy when all the characters of a film were included (Table 5.1).

Another aspect which might seem surprising is the consistent better performance of random forests over SVMs, which are regarded to be a more sophisticated model. A often-cited study [12] found that, on average, random forests do tend to perform better than SVMs, when averaging performance across a wide array of problems. This is in part due to the fact that SVMs are more sensitive to the effects of using sub-optimal parameters.

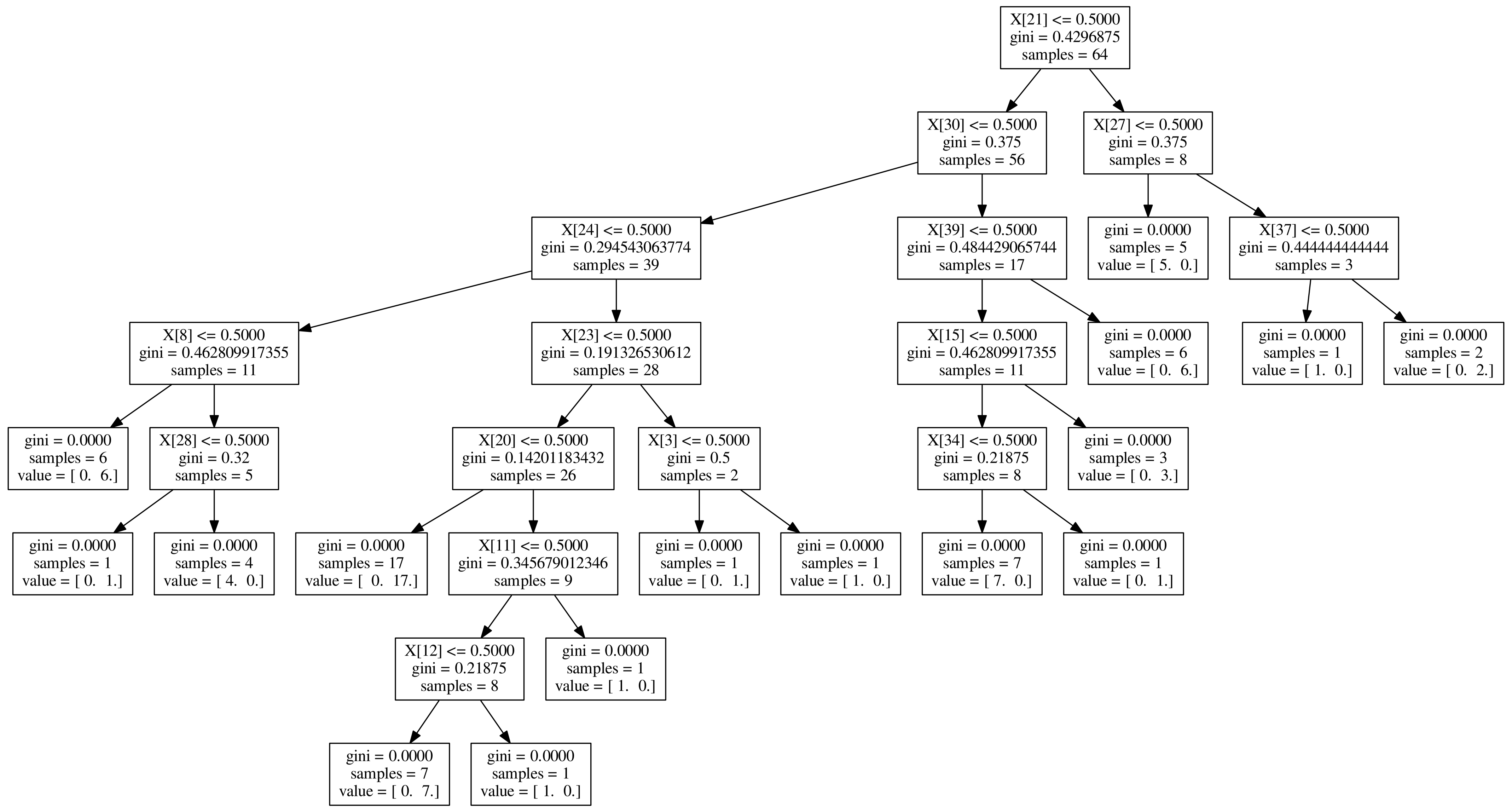
5.2.3 Predicting compound sentiment

We have modelled previous tasks as classification problems. However, inter-character sentiment is a continuous numerical value, meaning we can also explore regression tasks. The broadest such task is simply predicting the value of inter-character compound sentiment. Attempting to obtain the best possible results, we will explore the whole space of configurations introduced in the sections above. Results are shown in Table 5.6.

Configuration						
Classifier	No. nodes	Filter personas	Personas used	Top topics only	Edge used	**Mean squared error**
Linear regression	All	Yes	M	7	Yes	10.19
	All	Yes	M	7	No	10.19
Ridge regression	All	Yes	A, M	7	Yes	10.30
	All	Yes	A, M	7	No	10.28
Decision tree regression	All	Yes	A	2	Yes	10.55
	All	Yes	A	2	No	10.50
SVM regression	All	Yes	A	3	Yes	10.99
	All	Yes	A	3	No	10.99

Table 5.6: Best sentiment polarity prediction accuracy.

Prediction results are very poor, with all the predictors achieving the mean squared error of around 10.0. The value of cumulative sentiment for the films in our data set ranges between -18.0 and +20.0. Predicting sentiment value was quite an ambitious task. The reasons behind poor performance, can, in addition to those discussed for the classification task, be the small number and diversity of films used, sub-optimal parameters (particularly for ridge regression and SVM regression) or the fact that the value of sentiment can be more sensitive to features of the language (rather than a character's persona) and more easily skewed by outliers than sentiment polarity and sentiment variance.


X[21] <= 0.5000
gini = 0.4296875
samples = 64
X[30] <= 0.5000
gini = 0.375
samples = 56
X[27] <= 0.5000
gini = 0.375
samples = 8
X[24] <= 0.5000
gini = 0.294543063774
samples = 39
X[39] <= 0.5000
gini = 0.484429065744
samples = 17
gini = 0.0000
samples = 5
value = [5. 0.]
X[37] <= 0.5000
gini = 0.444444444444
samples = 3
X[8] <= 0.5000
gini = 0.462809917355
samples = 11
X[23] <= 0.5000
gini = 0.191326530612
samples = 28
X[15] <= 0.5000
gini = 0.462809917355
samples = 11
gini = 0.0000
samples = 6
value = [0. 6.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 2
value = [0. 2.]
gini = 0.0000
samples = 6
value = [0. 6.]
X[28] <= 0.5000
gini = 0.32
samples = 5
X[20] <= 0.5000
gini = 0.14201183432
samples = 26
X[3] <= 0.5000
gini = 0.5
samples = 2
X[34] <= 0.5000
gini = 0.21875
samples = 8
gini = 0.0000
samples = 3
value = [0. 3.]
gini = 0.0000
samples = 1
value = [0. 1.]
gini = 0.0000
samples = 4
value = [4. 0.]
gini = 0.0000
samples = 17
value = [0. 17.]
X[11] <= 0.5000
gini = 0.345679012346
samples = 9
gini = 0.0000
samples = 1
value = [0. 1.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 7
value = [7. 0.]
gini = 0.0000
samples = 1
value = [0. 1.]
X[12] <= 0.5000
gini = 0.21875
samples = 8
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 7
value = [0. 7.]
gini = 0.0000
samples = 1
value = [1. 0.]

Chapter 6

Conclusions and future work

We have presented a novel representation of films, as a graph which models characters and their interactions. We also gave an overview of the previous work which guided the design and implementation decisions taken in this project and discussed the details and suitability of the different methods chosen. We have constructed and evaluated the character graphs using 86 thriller and comedy films, based on their ability to predict various features of the inter-character sentiment. We ran several experiments on different character graph variations and different subsets of the data set and found the representations and the prediction tasks that achieve the greatest accuracy. Although it seems that using the graph edge weight as a feature in our representation slightly hinders prediction accuracy in many cases, we have shown the viability of our simplified Dirichlet Persona Model, adapted from the work of Bamman et al. [5].

6.1 Future work

Of course, in our exploration, we have only scratched the surface of this problem. At every step, there were many possibilities of doing things differently, which we did not have a chance to try in the scope of an honours project. A non-exhaustive list would include:

- evaluate our character graphs using the original Dirichlet Persona Model and the Persona Regression Model in addition to our simplified persona model
- spend more time optimising the parameters of machine learning models such as SVM and ridge regression, which could potentially improve their performance significantly
- use other, more established, sentiment analysis systems, such as SentiWordNet[1] or LIWC[2] and compare the results against those obtained with VADER

[1]www.sentiwordnet.isti.cnr.it
[2]www.liwc.net

- rigorously evaluate the sentiment trajectories that we constructed; in the project, we only evaluated them by exemplifying how they can convey some aspects of the narrative on a selection of films
- process more films scripts in order to test other hypotheses; although not mentioned in the report, we attempted this, but the task proved very laborious and not easily automated, requiring manual annotation of each script
- explore more variations of the data space; in particular, vary the number of topics used for Latent Dirichlet Allocation; training LDA is quite computationally expensive (especially on more than 10.000 plot summaries), so we did not attempt too many possibilities
- as an alternative to the character graph model proposed, we could have used sentiment analysis to enhance either the character personality model or the edge weight, and use a different evaluation measure
- implement a procedure for comparing character graphs; the work in [15] could be a starting point

6.2 Final thoughts

I would like to end this report with a few remarks on the difficulty of computationally analysing works of fiction, by discussing a very recent and hotly debated example. In the Introduction, we touched upon the belief that stories can, on a high level, be generalised and that the gist of many successful stories can be captured by a "shape" or skeleton. Recent work by Matthew Jockers on the Syuzhet package[3] aims to accomplish this: automatically extract the "latent structure of narrative by means of sentiment analysis". The model was used on 40.000 novels. Jockers claims that when comparing the shape of these novels, 6 or 7 distinct story archetypes emerge[4].

Although the paper presenting his approach has not yet been published (it is due to appear later this year), Syuzhet has received some media attention and has attracted discussion and criticism from researchers in the NLP and DH (digital humanities) communities, mostly expressed in exchanges of tweets and blog posts. Annie Swafford raises a number of concerns, including doubts about the accuracy of the sentence splitting technique employed and on the choice of smoothing function applied to the "emotional trajectories" of the novels[5]. She also points out the limitations of using a number of unsophisticated sentiment analysis lexicons (among others, AFINN-96, discussed in Section 4.3) when computing sentence-level sentiment, including the problem with simply summing the valences of words in a sentence, the loss of multiple

[3] www.github.com/mjockers/syuzhet

[4] Matthew Jockers. *The Rest of the Story*. Blog post. 25 February 2015. www.matthewjockers.net/2015/02/25/the-rest-of-the-story. Note that this and all the following blog posts have been accessed on 1 April 2015.

[5] Annie Swafford. *Problems with the Syuzhet Package*. Blog post. 2 March 2015. www.annieswafford.wordpress.com/2015/03/02/syuzhet/

word senses by assigning a word only one sentiment valence and the fact that these lexicons are constructed with modern English in mind, while the novels analysed with Syuzhet span a much longer period of time. Others, such as Andrew Piper, have contributed to the debate by touching on the inherent difficulty of rigorously validating or invalidating a hypothesis about subjective constructs such as sentiment or the shape of stories[6]. An interesting point (which we earlier discussed in our assessment of sentiment analysis in Section 2.1.3) is that evaluating the accuracy of a procedure in this context will invariably depend on what humans agree to be correct. This is an issue in many common NLP tasks (coreference resolution, named entity recognition, even POS tagging, to a much lower extent), where human agreement is relatively low. One person's sentiment trajectory of a story will probably not match with the other's. David Bamman tried to address this point by having five people annotate each scene of *Romeo and Juliet* with a sentiment valence between -5 and +5 via the Amazon Mechanical Turk crowdsourcing platform[7]. Agreement was high for some scenes (e.g. the balcony scene), but less so for others, especially towards the end of the play.

Beyond any technical shortcomings that might have inadvertently affected the work described in this report, the results presented here are also ultimately subject to the same validity concerns that are being discussed in this ongoing debate. We have established some correlations and demonstrated the usefulness of character graphs, but it is important to keep in mind that deeper claims about the nature of narratives might be harder to support.

[6]Andrew Piper. *Validation and Subjective Computing*. Blog post. 25 March 2015. www.txtlab.org/?p=470

[7]David Bamman. *Validity*. Blog post. 1 April 2015. http://www.davidbamman.com/?p=52

Bibliography

[1] Apoorv Agarwal and Owen Rambow. Automatic Extraction of Social Networks from Literary Text : A Case Study on Alice in Wonderland. *the Proceedings of the 6th . . .*, pages 1202–1208, 2013.

[2] Apoorv Agarwal, Owen Rambow, and Rebecca J Passonneau. Annotation scheme for social network extraction from text. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 20–28. Association for Computational Linguistics, 2010.

[3] Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 579–586. Association for Computational Linguistics, 2005.

[4] Cecilia Ovesdotter Alm and Richard Sproat. Emotional sequencing and development in fairy tales. In *Affective Computing and Intelligent Interaction*, pages 668–674. Springer, 2005.

[5] David Bamman, Brendan O'Connor, and Noah Smith. Learning Latent Personas of Film Characters. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 352–361, 2013.

[6] David Bamman, Ted Underwood, and Noah a Smith. A Bayesian Mixed Effects Model of Literary Character. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 370–379, 2014.

[7] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2012.

[9] C. Booker. *The Seven Basic Plots: Why We Tell Stories*. Bloomsbury Academic, 2004.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[12] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[14] Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. *URL http://nlp. stanford. edu/software/dependencies manual. pdf*, 2008.

[15] Micha Elsner. Character-based kernels for novelistic plot structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 634–644. Association for Computational Linguistics, 2012.

[16] David K Elson, Nicholas Dames, and Kathleen R McKeown. Extracting social networks from literary fiction. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (July):138–147, 2010.

[17] Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. *ICWSM*, 7:21, 2007.

[18] Philip John Gorinski and Mirella Lapata. Graph-based Scene Extraction for Movie Script Summarisation. *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, 2015.

[19] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[20] CJ Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[21] William Jinks. The celluloid literature: Film in the humanities. 1971.

[22] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[23] Owen Macindoe and Whitman Richards. Graph comparison using fine structure analysis. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 193–200. IEEE, 2010.

[24] Viktor Mayer-Schönberger and Kenneth Cukier. *Big data: A revolution that will transform how we live, work, and think.* Houghton Mifflin Harcourt, 2013.

[25] Eric T Nalisnick and Henry S Baird. Character-to-character sentiment analysis in shakespeares plays. pages 479–483, 2013.

[26] Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.

[27] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[28] Alan Said, Shlomo Berkovsky, and Ernesto W. De Luca. Putting things in context: Challenge on context-aware movie recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 2–6, New York, NY, USA, 2010. ACM.

[29] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[30] Raisa Varghese and M Jayasree. A survey on sentiment analysis and opinion mining. *IJRET: International Journal of Research in Engineering and Technology eISSN*, pages 312–317, 2013.

[31] K. Vonnegut and D. Simon. *A Man Without a Country*. Seven Stories Press, 2011.

[32] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics, 2005.

Appendix A

CMU Movie Summary Corpus preprocessing example

This appendix supports the explanation is Section 4.2.1. To illustrate the preprocessing steps that take the raw plot summaries and generate the input for the work described in [5] and used in this project, we will give a concrete example of a sentence in a plot summary, as it is passed through the preprocessing pipeline. For this we will use the first sentence in the Wikipedia plot summary for *The Silence of the Lambs* (1994).

We start with the unprocessed sentence:

```
Clarice Starling  is pulled from her training at the FBI Academy
at Quantico, Virginia, by Jack Crawford  of the Bureau’s Behavioral
Science Unit.
```

The sentence is tokenised, with tokens being split by spaces:

```
Clarice Starling is pulled from her training at the FBI Academy at
Quantico , Virginia , by Jack Crawford of the Bureau ’s Behavioral
Science Unit .
```

Then, tokens are lemmatised (i.e. transformed into their canonical, non-inflected form):

```
Clarice Starling be pull from she training at the FBI Academy at
Quantico , Virginia , by Jack Crawford of the Bureau ’s
Behavioral Science Unit .
```

The sentence is also annotated with part-of-speech (POS) tags:

```
NNP NNP VBZ VBN IN PRP$ NN IN DT NNP NNP IN NNP , NNP , IN NNP
NNP IN DT NNP POS NNP NNP NNP .
```

It is also annotated with the named entity information:

```
PERSON PERSON O O O O O O O ORGANIZATION ORGANIZATION O
LOCATION O LOCATION O O PERSON PERSON O O ORGANIZATION O
ORGANIZATION ORGANIZATION ORGANIZATION O
```

Stanford typed dependencies are computed:

```
[["nn",0,1],["nsubjpass",1,3],["auxpass",2,3],["poss",5,6],
["prep_from",6,3],["det",8,10],["nn",9,10],["prep_at",10,6],
["nn",12,14],["prep_at",14,3],["nn",17,18],["agent",18,3],
["det",20,21],["poss",21,25],["nn",23,25],["nn",24,25],
["prep_of",25,18]]
```

And the sentence is syntactically parsed:

```
(ROOT (S (NP (NNP Clarice) (NNP Starling)) (VP (VBZ is)
(VP (VBN pulled) (PP (IN from) (NP (NP (PRP$ her) (NN
training)) (PP (IN at) (NP (DT the) (NNP FBI) (NNP Academy)))
)) (PP (IN at) (NP (NNP Quantico) (, ,) (NNP Virginia) (, ,)))
 (PP (IN by) (NP (NP (NNP Jack) (NNP Crawford)) (PP (IN of)
 (NP (NP (DT the) (NNP Bureau) (POS ’s)) (NNP Behavioral) (NNP
 Science) (NNP Unit))))))) (. .)))
```

Finally, each token is categorised as being an *agent*, *patient* or *modifier* word based on its typed dependency (those which do not have one of the relevant dependencies are discarded). Coreference resolution is used to resolve the dependencies to the correct character. Where it is possible, character names are matched with Freebase entities, otherwise, an internal identifier is used. The final input file contains tuples of the form:

```
/m/0k6g8c:t0.1.0:verb.contact:pull:a:agent
/m/0k6g8p:t2.2.0:verb.stative:lead:a:agent
/m/0k6g86:t3.0.0:verb.change:grow:a:nsubj
/m/0k6g86:t3.2.0:verb.communication:rebuff:a:nsubj
/m/0k6g86:t5.1.0:verb.communication:call:a:nsubj
/m/0k6g86:t5.2.0:verb.communication:tell:a:nsubj
/m/0k6g86:t8.0.0:verb.possession:offer:a:nsubj
/m/0k6g86:t8.4.0:verb.emotion:detest:a:nsubj
/m/03jrpdm:t9.0.0:verb.contact:kidnap:a:nsubj
/m/0k6g8c:t9.1.0:verb.communication:authorize:a:nsubj
/m/0k6g81:t9.2.0:verb.possession:offer:a:nsubj
/m/0k6g81:t9.4.0:verb.possession:provide:a:nsubj
/m/0k6g81:t10.2.0:verb.possession:give:a:nsubj
/m/0k6g8p:t11.0.0:verb.communication:record:a:nsubj
/m/0k6g8p:t11.1.0:verb.perception:reveal:a:nsubj
/m/0k6g86:t12.0.0:verb.communication:agree:a:nsubj
/m/0k6g86:t12.3.0:verb.perception:reveal:a:nsubj
(...)
```