Linguistic **I**ssues in **L**anguage **T**echnology – **LiLT**
Submitted, October 2015

Abstract Representations of Plot Structure

Micha Elsner

Published by CSLI Publications

LiLT volume 12, issue 5 October 2015

Abstract Representations of Plot Structure

Micha Elsner, *Department of Linguistics, The Ohio State University, melsner@ling.ohio-state.edu*

Abstract

Since the 18th century, the novel has been one of the defining forms of English writing, a mainstay of popular entertainment and academic criticism. Despite its importance, however, there are few computational studies of the large-scale structure of novels—and many popular representations for discourse modeling do not work very well for novelistic texts. This paper describes a high-level representation of plot structure which tracks the frequency of mentions of different characters, topics and emotional words over time. The representation can distinguish with high accuracy between real novels and artificially permuted surrogates; characters are important for eliminating random permutations, while topics are effective at distinguishing beginnings from ends.

LiLT Volume 12, Issue 5, October 2015.
Abstract Representations of Plot Structure.

1 Introduction

The novel, one of the characteristic forms of modern English literature, poses several interesting challenges from the point of view of computational analysis. Some of these have to do with the sheer size of a novel. Unlike a newspaper or encyclopedia article, novels routinely stretch to tens or hundreds of thousands of words, covering complex sequences of interlocking characters and events. Other challenges are representational. It is clear that not all descriptions of sequences of events make for acceptable novelistic plots, but literary theorists have taken a variety of perspectives on what the defining characteristics of plot structure actually are.

This paper attempts to represent "plot structure" at a high level, abstracting away from any particular events. The aim is to capture basic concepts such as "protagonist" or "happy ending" in ways that apply across a broad range of texts. Whenever possible, the representation is constructed using lexical distribution rather than requiring text analyses (possibly error-prone) with complex NLP tools. This representation is used to create models capable of distinguishing real novels from artificially disordered texts for which plot structure is missing or incomprehensible.

This approach is motivated by the inadequate performance of current general-purpose NLP systems on novelistic text. For instance, general-purpose extractive summarizers fail to find suitable sentences (Kazantseva and Szpakowicz, 2010). The output of Microsoft Word 2008's built-in summarizer on *Pride and Prejudice* is shown in Figure 1.

> "Bingley." Elizabeth felt Jane's pleasure. "Miss Elizabeth Bennet." Elizabeth looked surprised. "FITZWILLIAM DARCY" Elizabeth was delighted. Elizabeth read on: Elizabeth smiled. "If! "Dearest Jane!

FIGURE 1 Output the built-in summarizer in Microsoft Word 2008, run on the full text of *Pride and Prejudice*; quoted by Huff (2010).

Part of the problem is that counting unigrams at the document level identifies character names, rather than themes or plot elements, as the most important content of *Pride and Prejudice*. Another issue is the extractive assumption. Not only has this summary selected the wrong sentences, but the novel may not contain a good set of "topic sentences" to begin with. Thus, rather than asking whether it is possible to summarize *Pride and Prejudice* by drawing specific sentences from within the text that say "what the story is about", summarizers may be better

off representing its structure in terms of similarities to other works. For instance, one can say *Pride and Prejudice* is a romance. This ties it to a variety of other books (including Jane Austen's other works) and sets up some basic expectations in the mind of a potential reader: the story will focus on a protagonist (probably female) who falls in love and eventually gets married. The goal of this paper is to construct an automatic function for measuring similarity between novels, as progress toward eventual summarization, search and recommendation systems.

While this study focuses on public-domain works from the 19th and early 20th centuries, these representational strategies will hopefully continue to be useful on modern texts from the Western novelistic tradition. Modern writers continue to produce vast amounts of fictional text. That includes amateur authors whose output is never formally published or curated; in 2010, over 200,000 people participated in NaNoWriMo (national novel writing month). A system able to provide summaries or other representations of these texts could aid in making them more accessible to prospective readers as well as to academics such as sociolinguists or narratologists.

The system represents a novel as a set of trajectories which describe the frequencies of various kinds of language over time. Two novels are defined as similar if their trajectories for various features tend to look alike. Several options are available for constructing such trajectories. Either the entire novel can have a single representation, or there can be features for the language associated with individual characters within it. And there are multiple possibilities for linguistic features to track. This paper describes representations based on sentiment words, the frequencies of mentions of the characters themselves, and word clusters created by Latent Dirichlet Allocation (henceforth, LDA; Blei et al., 2001). LDA groups related words, such as "song" and "melody", because they frequently occur together in the same works.

Some literary scholars claim that sentiment is central to the high-level construction of plots. Crane (2002) writes that effective plots cause us to "wish good [for some of the characters], for others ill, and depending on our inferences as to the events, we feel hope or fear... or similar emotions". Phelan and Rabinowitz (2012) extend this approach to a rhetorical theory of narrative which analyzes stories by examining the kinds of emotional and moral judgements which the author intends to elicit from the audience. On the other hand, LDA induces word clusters directly from the corpus and can therefore learn features which a general-purpose lexicon could not. The experiments discussed in this paper evaluate the strengths and weaknesses of each approach.

These representational choices are evaluated through the construc-

tion of a similarity function, or *kernel*, which measures how alike two novels are in a given feature representation. Representation quality can then be compared by drawing on an existing tradition of discourse coherence evaluation using artificial reordering experiments (Karamanis et al., 2004). In these experiments, the kernel is used to distinguish novels that have been randomly permuted, or reversed, chapter-by-chapter, from the originals.

The proposed representation and experimental setup implicitly assume that novels share a common sequential structure—an emotional/rhetorical structure for systems using sentiment features or a chronological sequence for LDA. Deviations from these structural principles are a common way to create suspense or draw attention to the narrative as an artifact. In a 19th-century example, the long flashback in *The Tenant of Wildfell Hall* breaks the chronological sequence to reveal the details of a character's mysterious past. Post-modern novels often involve even more complex structures, such as intertwined or nested narratives (for instance, David Foster Wallace's *Infinite Jest* or Milorad Pavić's *Dictionary of the Khazars*). Computational linguists have devoted some effort to representing such texts (Ryan, 1991) and disentangling their narrative threads (Wallace, 2012). The representation proposed here is unlikely to work well with texts which violate its core assumption of a sequential structure, but might still be helpful in representing single threads extracted by such a technique.

The experimental results indicate that character-based systems and single-trajectory systems which represent the whole novel at once have complementary strengths. Character frequency is most effective for distinguishing randomly permuted novels from originals, while a single trajectory of LDA features is most effective for reversals. This implies that tracking the course of events (as in many previous models) is indeed important in making sure the plot sequence points in the correct direction, from beginning to end. For instance, the rise in suspense or exciting sentiment at the end of a mystery or adventure story is a property of the story as a whole, not an emotion felt only by the protagonist. It is also important, however, to track the kinds of characters who appear in a work, and so help distinguish a coherent plot structure from a randomized one; minor characters' marriage occurs throughout Jane Austen's novels, but the protagonist only gets married at the end.

This paper builds on previous work (Elsner, 2012), which proposed the initial building blocks of the representation used here, but expands on it in several ways. First, this work tracks a wider range of lexical features, using a more sophisticated sentiment lexicon as well as LDA topics. Second, when comparing characters from one novel to charac-

ters from another, creating an explicit one-to-one map of corresponding characters (*symmetrization*) improves upon the previous technique of comparing all pairs. The results, especially the use of symmetrization, are substantially better than those reported in (Elsner, 2012); the best (one-neighbor) classification of randomly permuted novels increases from 62% to 82% and of reversals from 52% to 89%. Further comparisons are given in subsection 6.2.

2 Related work

2.1 High-level representations

Previous attempts to represent plot structure automatically have largely focused on understanding stories sentence-by-sentence. A few have attempted to construct high-level representations as in this paper; many of these have settled on characters as an important representational primitive.

The work by Coll Ardanuy and Sporleder (2014) is the closest to what is presented here. They too use both static and time-varying character similarity features to represent texts as graphs with characters as nodes. They use these graphs to define a text similarity metric, and cluster texts by author and genre. Unlike this work, however, they gather features over the network as a whole (for instance, the proportion of male characters) rather than trying to define text similarity by building up from character similarities. Thus, the feature sets considered in their work are quite different from those used here.

Elson et al. (2010) also build social networks for characters in novels, which they use to evaluate several questions in literary theory. As in this work, they begin by identifying characters using coreference resolution on mentions. They construct the social network based on conversation structure (O'Keefe et al., 2012, Elson et al., 2010); the experiments here use the simpler, but less precise, heuristic of co-frequency counts. Their work, however, does not use time-varying features, but collapses the novel over time, producing a static picture of a dynamic system.

Bamman et al. (2014) use a mixed-effects model to infer latent character types from the text of a large set of novels. Like Elson et al. (2010), they have a feature set that is not time-varying. They build upon previous work by Bamman et al. (2013) who use a Bayesian model to learn a set of character roles such as PROTAGONIST, LOVE INTEREST and BEST FRIEND for movie characters, but using metadata rather than scripts directly. They evaluate against a set of preregistered hypotheses.

Alm and Sproat (2005), on the other hand, produce an explicitly temporal structure, a time-varying curve of emotional language over

time, which they call an emotional trajectory. Alm and Sproat (2005) rely on hand-annotation of the trajectory. They produce a single trajectory per story (or several stories). Subsequent work on sentiment analysis of fiction and literary texts has retained the single-trajectory assumption, while focusing on enriching the set of sentiments used by the system and improving techniques for detecting them. Volkova et al. (2010), for instance, describe a protocol for hand-annotation of sentiment in fairy tales which allows non-experts to achieve high agreement.

Mohammad and Turney (2010) construct a large emotional lexicon using Mechanical Turk crowd-sourcing, which is used in the systems presented here. In Mohammad (2011, 2012), it is used to construct emotional trajectories for a few literary works such as *Hamlet*, but, apart from a broad corpus-level comparison between novels and fairy tales, no effort is made to evaluate these systematically. Ang (2012) creates trajectory plots using Mohammad and Turney's (2010) emotional categories, and evaluates them as part of a toolkit for writers. In a survey, writers find the tool interesting, although they point out that its inferred sentiments can be inaccurate due to negation and other effects of context.[1] This paper evaluates both traditional trajectory systems which produce a single trajectory per work, and multiple trajectories, one per character. It finds that both can be effective at capturing different aspects of plot structure.

An alternative to tracking sentiment distributions is tracking topics—words which occur together in context. LDA (Blei et al., 2001), the now-standard topic model, has been used in a variety of analyses in the digital humanities (Salway and Herman, 2011). LDA groups the words of a corpus into semantic "topics" by considering the set of documents in which they co-occur. The vector of topic frequency counts within a document can then be used as a coarse-grained approximation of its content.

Although the frequencies of LDA topics naturally vary throughout the course of a long text, LDA does not directly model temporal variation. Subsequent work proposes a variety of Bayesian topic models which make more sophisticated use of document metadata, including sequence ordering (Blei and Lafferty, 2006, Kim and Sudderth, 2011). These papers tend to evaluate using a combination of held-out likelihood and eyeball, often on corpora of scientific journal articles—an approach criticized by Chang et al. (2009) and Mimno and Blei (2011).

[1] Sentiment analysis systems which work on whole clauses or sentences are an active research area (Socher et al., 2011, Yessenalina and Cardie, 2011, among others); to my knowledge, such systems have not been used to construct emotional trajectory models.

The emperor rules the kingdom. The kingdom holds on to the emperor. The emperor rides out of the kingdom. The kingdom speaks out against the emperor. The emperor lies.

FIGURE 2 Story generated by an event-based model with coherence reranking (McIntyre and Lapata 2010).

How these representations compare to sentiment or other features on novelistic text is an open question.

This paper uses the simpler method of running standard LDA and measuring temporal patterns on the resulting topics. While more sophisticated methods do have published implementations, they tend to be slower, less stable and less scalable than LDA. Since the results in this paper show that topic model features are useful in capturing a global beginning-to-end temporal structure, evaluating these more complex topic models is a promising direction for future work.

2.2 Event-based representations

In contrast to these abstract representations are models that stay closer to the level of specific events extracted from sentences in the text. Such models have usually been applied to shorter fictional narratives such as fables. Many of these follow from narrative schema extraction (Chambers and Jurafsky, 2009), which attempts to learn representations for events in news stories. These representations are similar to Schankian scripts (Schank and Abelson, 1977); they are networks of events in temporal sequence, with slots for specific actors. For instance, "*terrorist* attacks *target*" can be followed by "*terrorist* being arrested".

Similar representations for fiction were investigated by McIntyre and Lapata (2009), who use them to generate short fables. In later work (McIntyre and Lapata, 2010), they add a coherence component to ensure smooth sentence-to-sentence transitions. Figure 2 shows a sample output. The lack of global "plot" structure is an important shortcoming of this work; while the generated stories describe reasonable sequences of events, the stories do not seem to raise or resolve any central conflict.

Similar narrative models are considered by Li et al. (2012), who learn event networks from a corpus of short texts elicited through crowdsourcing. These texts focus on specific events such as bank robberies or dates. Importantly, although the goal of this work is to learn knowledge that may be useful in constructing or understanding narratives, the elicited texts themselves are not fictional narratives intended to entertain but simple descriptions; thus the models capture sequences

of events without necessarily producing a satisfying plot structure.

Finlayson (2009) also produces event network models, following a Proppian structuralist analysis of story structure (Propp, 1968). It has been evaluated against hand-annotated Russian folktales and a small corpus of Shakespeare's plays.

Early work by Lehnert (1981) proposes a model which represents emotion and goal information alongside events. This *plot-unit* model treats positive and negative sentiment as primitives, and builds up a plot as a set of actions which result from, and cause, characters to feel good or bad. For instance, a *retaliation* has the form: "Because Y's [action] caused a [negative state] for X, X [acted] to cause a [negative state] for Y". Early implementations relied heavily on hand-engineered domain knowledge, and thus did not generalize well. AESOP (Goyal et al., 2010) attempts to modernize this representation by learning which verbs cause positive or negative states automatically. A similar goal-based representation is learned in O'Neill and Riedl's (2011) work, which performs story generation by abductive plan inference. These representations are sophisticated, but brittle; AESOP's accuracy is poor even for short fables, and the authors conclude that the approach is unlikely to be scalable. Elson et al. (2010) present Scheherezade, a system for human annotation of AESOP-like goal structures; a small annotated corpus is available (Elson and McKeown, 2010). However, extending this expensive hands-on annotation task to novelistic texts is likely to be prohibitively time-consuming.

Kazantseva and Szpakowicz (2010), while relying on sentence-based information, is somewhat different in its objectives. Rather than modeling plot structure, it attempts to produce "spoiler-free" summaries which exclude plot detail while incorporating character and setting description, using features such as stative verbs ("stand, know, be located") to find appropriate sentences. They analyze the performance of summarization systems on fictional text and find that conventional systems are deeply inadequate; this serves as further motivation for our own work.

2.3 Kernels, symmetrization and parameter optimization

The systems presented in this paper are *kernel functions*, a standard tool in machine learning for feature-based classification and regression (Bishop, 2006, chapter 6). A kernel is a similarity function $k(X, Y) \geq 0$, with 0 representing minimal similarity. A valid kernel represents an inner product in some feature space ϕ, so that $k(X, Y) = \phi(X) \cdot \phi(Y)$. The *linear* kernel takes ϕ as the identity and is simply a dot product.

More complex kernels have been proposed for the case where X and

Y are structured objects such as graphs (Vishwanathan et al., 2010). A common method of extending a simple kernel to an object with many substructures is to avail oneself of the convolution theorem (Haussler, 1999) which applies the simple kernel to all pairs of substructures and sums the result.

$$K(X,Y) = \sum_{u \in X} \sum_{v \in Y} k(u,v) \tag{1}$$

Other methods have been proposed, although not all of them result in theoretically valid kernels. The approach here follows the work of Boughorbel et al. (2004) in computer vision, who compute a mapping between X and Y and only evaluate $k(u,v)$ for matched pairs:

$$k_{match}(X,Y) = \max_{\text{matching } F: u \leftrightarrow v} \sum_{u \in X} k(u, F(u)) \tag{2}$$

Such a function can no longer be represented as a linear product, since it includes a maximization operator. However, it can still be used to provide positive similarities. Because these functions remain similar to kernels in practice, the paper will continue to describe them as such.

Additional intuition for this approach comes from machine translation (MT), where the use of a one-to-one mapping in word alignment is known as *symmetrization*. Symmetrization is a common method for improving word alignments. Suppose that an alignment between two sentences indicates that word u translates as v; when aligning in reverse, v should become u again. In this work, the same holds true for characters—character u from work X (Elizabeth Bennet from *Pride and Prejudice*) should correspond only to character v from work Y (Jane from *Jane Eyre*). If Elizabeth took on the roles of a whole set of characters, this would be a point of dissimilarity between the two novels. The simplest technique for computing a symmetric alignment is to run independent one-to-many models in each direction and take the intersection. MT researchers have improved upon this by computing a matching explicitly (Matusov et al., 2004) or training the component models directly to encourage agreement (Liang et al., 2006).

When the problem of interest involves multiple sets of features, it may be useful to set kernel parameters weighing the different features to optimize performance. This task is sometimes considered as *multiple kernel learning* (Gnen and Alpaydın, 2011); a variety of methods have been used. In this paper, kernel parameters are optimized using rank learning: attempting to make pairs of training instances which should be similar score higher under k than less similar ones. The ranking protocol was designed following Feng and Hirst (2012), who used a similar procedure to optimize parameters in a model of document coherence.

The specific rank learner used is SVM-rank (Joachims, 2006), which solves an optimization problem based on ordinal regression to approximately minimize the number of pairs ranked in the wrong order.

2.4 Evaluation by reordering

The experiments presented here test the kernel similarity function by challenging it to distinguish novels from artificial "negative examples". These are created from real texts by permuting the order of the chapters. This procedure originated in the discourse coherence literature (Karamanis et al., 2004, Barzilay and Lapata, 2005), in which it is assumed that most reorderings of a text cause a loss of coherence and are therefore suitable as negative examples. Post (2011) uses a similar idea to evaluate a model of grammaticality.

The use of artificial negative examples has both strengths and weaknesses. On one hand, it is highly replicable and objective in domains where annotation would be expensive and unreliable. In this case, human annotators would have to decide which real novels are more or less similar to one another. On an artificial task, systems can be evaluated for at least basic effectiveness without this kind of resource.

On the other hand, performance on artificial reordering tasks can fail to correlate with performance on more realistic tasks (Elsner et al., 2007). Random permutations can cause particular problems, since they often create local structures (for instance, a rarely mentioned minor character who appears in two widely separated chapters) which are uncommon in real documents. The use of novels in reverse order is intended to guard against this to some degree, because reversals destroy the global plot structure of the novel while preserving its local consistency. In any case, failure to correlate is primarily a problem with well-performing systems; systems which fail badly on tests with artificial documents rarely succeed on more complex ones.

Another potential testing strategy is that of Bamman et al. (2014), who make a series of pre-registered hypotheses on the similarity of characters by the same author or of the same type. For example, Elizabeth Bennet is more similar to Elinor Dashwood than Allan Quatermain ("Austenian protagonists should resemble each other more than they resemble a grizzled hunter"). Such tests are more realistic, since they are directly based on human intuition. However, they require input from a literary scholar and do not transfer easily between corpora. They are also unsuitable for training a machine learning algorithm, since that requires a large set of training instances to optimize against, while the set of hypotheses created by a literary scholar is typically small. Thus, this evaluation strategy seems complementary to the use of artificial

examples. Here, an evaluation in terms of literary hypotheses is left for future work.

3 Creating representations

As stated, the overall goal of this paper is to design systems for measuring the similarity of plot structures by comparing the frequency of linguistic features over time. This section discusses in detail the basic operations used to compute frequency-over-time trajectories for sentiment words, LDA topics and characters in novels. Two kinds of representations are constructed. *Single-trajectory* representations track the frequency of different lexical features across the narrative as a whole. In *character-trajectory* representations, on the other hand, each character in the novel has their own set of associated trajectories, indicating the frequency of appearance and the frequency of different lexical features associated with them.

The corpus used in all experiments consists of 50 novels, listed in the appendix. All novels are downloaded from the Project Gutenberg Website (www.gutenberg.org) in raw text form; the Gutenberg header and footer are stripped, as are introductory and concluding material by editors, critics or publishers. Since machine learning is used to optimize system parameters, the data are divided into a training set of 20 novels, used to establish the parameters, and a test set of 30 novels, reserved for evaluation.

3.1 Single trajectories

Most of the representations described here measure lexical frequency over time. Such systems require a particular lexicon of relevant words. This study uses two such lexicons: Mohammad and Turney's (2010) crowdsourced sentiment lexicon, and topics from LDA (Blei et al., 2001). These word frequency systems represent a text as the count, for each chapter, of how often words from a particular lexical category appear, normalized by the total number of word tokens from the lexicon. For instance, a system using the category "anger" would represent a 10-chapter novel as a 10-element list, with the first element being the percentage of emotion words in chapter 1 in the "angry" category.

Mohammad and Turney's (2010) lexicon is a list of words, each annotated for potential associations with various emotions by Amazon Mechanical Turk workers. The version used here contains 14273 words; the lexicon recognizes 8 different basic emotions (anger, anticipation, disgust, fear, joy, sadness, surprise and trust) and two umbrella cat-

Topic	word 1	...				
0	man	reply	lady	gentleman	boy	head
1	love	life	heart	world	thought	soul
2	mother	father	woman	make	year	day
3	return	person	time	receive	give	place
4	character	world	men	feeling	opinion	mind
5	hand	eye	face	voice	speak	word
6	form	light	woman	air	pass	beauty
7	thing	make	letter	time	write	read
8	room	house	door	night	time	day
9	men	foot	man	round	horse	time

TABLE 1 Most frequent words for LDA topics in our corpus.

egories, positive and negative, for a total of 10.[2] As with any purely lexical sentiment resource, it will mis-classify texts which are context-specific, specific to some sense of the word, or negated by their semantic context ("not happy", "wish to be happy").

The LDA topics are computed with Mallet (McCallum, 2002), stripping stop words and specifying 10 topics (to keep parity with the number of emotions). Topic counts are computed from the output list of topic indicator variables z. Table 1 shows the most common words grouped in each topical category. In many cases the topics are quite interpretable; topic 1 ("love, life, heart, soul") seems to involve romantic feeling, while topic 8 ("room, house, door") covers setting details. But the clustering is very coarse; less frequent words associated with topic 1 include non-romantic feeling words like "fear, hope, death".

3.2 Character trajectories

A character-trajectory representation has a trajectory for each character in the narrative. For instance, a system using character frequency as its only feature represents a 10-chapter novel with three characters as a bundle of three 10-element lists. Each list represents the normalized frequency with which the associated character occurs in each chapter. A system using both character frequency and anger represents the novel as three bundles of two 10-element lists. Each bundle contains the character frequencies in one list, and the frequency with which the character is associated with "anger" words in the other.

To count how often a character appears over time, the system must first compute a canonical list of the characters who appear in the text, bearing in mind that one character may be called by many names. This

[2]The paper subsequently refers to all 10 of these as "emotion" or "sentiment" categories.

computation follows Bhattacharya and Getoor (2005) in extracting a list of proper names and performing cross-document coreference resolution using a series of filters which cluster them together. A related framework is described in (Coll Ardanuy and Sporleder, 2014).

The system begins by detecting proper nouns and discarding those which occur fewer than 5 times. Identical mentions longer than two words are merged into a single entity, so that for example all mentions of "George Osborne" are taken to refer to the same person. Next, each name is assigned a gender—masculine, feminine or neuter—using a list of gendered titles, then a list of male and female first names from the 1990 US census. Mentions are then merged when each is longer than one word, the genders do not clash, and first and last names are consistent (Charniak, 2001). This step would cluster "George Osborne", "Mr. Osborne" and "Mr. George Osborne". Single-word mentions are then merged, either with matching multiword mentions if they appear in the same paragraph, or else with the multi-word mention that occurs in the most paragraphs—so when "George" appears close to "George Osborne", the two refer to the same person. Finally, mentions are discared if they still have not been assigned a non-neuter gender, or if they match synset *location.n.01* in WordNet (Miller et al., 1993); these are likely to be place names like "London", which are proper noun phrases but not characters.

These coreference heuristics still have some difficulty in coping with the variety of names used by characters in 19th-century novels. These stories often contain characters who are related to one another (and thus share a last name); characters refer to one another by nickname; titles—and even names—can change over time (due to marriage, military promotion and so on). For instance, in Thackeray's *Vanity Fair*, Mr. (John) Osborne has a son, George Osborne, initially with the title Master, then Mr., but eventually rising to the rank of army captain. George, in turn, marries (making his wife *Mrs.* George Osborne) and has a son, who is inconveniently named George Osborne. By the end of the book, this son is himself known as Mr. Osborne. Table 2 shows how our system tries to resolve this confusion; since it insists that titles be consistent, it produces a somewhat excessive list of characters. This is one of the harder cases in the corpus, however, and in general the results look sensible, although there is no annotated novelistic coreference corpus with which to validate them.

To represent a novel using character frequency, the system preprocesses the text by splitting it into paragraphs. The coreference heuristics are used to decide which characters appear in each paragraph. The frequency of a character in a chapter is defined as the number of para-

Character (longest name)	gender	count
Mrs. George Osborne	F	662
Georgy Osborne	N	344
Capt. George Osborne	M	153
Mr. Osborne	M	146
Miss Jane Osborne	F	75
Master George	M	8
Mr. George	M	7
Lt. Osborne	M	7

TABLE 2 Names (over frequency cap) of characters named "George" or "Osborne" detected in *Vanity Fair*. "Mr. Osborne" can refer to multiple characters. "Capt. Osborne" sometimes refers to the same person as "Mr. Osborne". "Georgy" is missing from the name list and fails to be assigned a gender, so it is discarded from consideration in character-based systems.

graphs in which they appear, divided by the total number of character appearances.

Character-specific lexical trajectories are computed by counting lexicon items—sentiment or LDA—in paragraphs featuring a specific character. Such trajectories naturally reflect overall character frequency, since there can be more instances of particular words if there are more paragraphs overall. To remove this correlation so that the character-frequency features are not redundant, they are normalized per character rather than with reference to the whole text. To compute this normalized score, the count is first set to 0 if the character frequency for the chapter is less than .05 (so that dividing by a small number does not inflate very uncertain statistics). Then the count is divided by the total number of lexicon items appearing in paragraphs of the chapter which mention the specific character (rather than dividing by the number of lexicon items in the chapter overall).

All trajectories are smoothed using a moving average with a window size of 10, forcing them to vary smoothly over time:

$$x'(t) = \frac{1}{\sum_{i=t-10}^{t+10} |t-i|} \sum_{i=t-10}^{t+10} |t-i| x(i) \tag{3}$$

After smoothing, each trajectory is projected onto a fixed basis of 50 points using linear interpolation. This enables fast comparison of trajectories, because it yields a fixed-length discrete representation of each one.

4 Kernel-based measurements

After preprocessing, a novel is represented as a set of time-varying trajectories, each representing the proportional frequency of some feature (e.g., "anger" words) in each chapter. In single-trajectory systems, there is one set of trajectories for the entire novel, with a trajectory for each lexical feature. In character-based or hybrid character/lexicon systems, characters have their own associated trajectories. A similarity function for representations of this type can be defined as a kernel function $k(X, Y)$ which will be large when novels X and Y are similar and small when they are not.

The function k for this representation relies on a function c which compares trajectories for a single feature. The system uses the simple dot product (the linear kernel function),[3] defining the similarity of a pair of trajectories u and v (e.g. "anger" over time in two novels) as:

$$c(u, v) = u \bullet v \tag{4}$$

To combine trajectories for multiple features (various emotions or topics), the total similarity k is defined as a weighted sum controlled by a parameter vector θ which represents the relative weight assigned to each one in the overall similarity function:

$$k(u, v; \theta) = \sum_{\text{coord } i} \theta_i c(u_i, v_i) \tag{5}$$

This function k will be called the "single-trajectory plot kernel" since it can be used to evaluate similarity for representations which do not include per-character trajectories.

We cannot use k for representations with character features, because there is no obvious way to determine which pairs of characters should be compared with function c. For instance, should the system use k to compare Elizabeth from *Pride and Prejudice* to Jane from *Jane Eyre*, or to Rochester, or to someone else? One standard approach is to define $k^{char}(X, Y)$ using the convolution theorem (Haussler, 1999, Equation 1), which compares each character u from X to all the characters v from Y:

$$k^{char}(X, Y) = \sum_{u \in X} \sum_{v \in Y} k(u, v; \theta) \tag{6}$$

[3] Another common choice would be the normalized cosine, a standard measurement in information theory. In development, however, the normalized cosine caused problems, because it assigns high similarity to pairs of curves which have relatively small values throughout. Such small values are generally uninteresting—every novel has rare characters, but this does not represent any deep similarity in plot.

4.1 Symmetrization

Intuitively, the construction of k^{char} from c is intended to find two novels more similar if characters from one correspond to the characters from the other in a way that makes sense. For instance, *Pride and Prejudice* is like *Jane Eyre* in that they both have a female protagonist, a male love interest, and so forth. The convolution theorem does not restrict itself to sensible alignments, however. It counts many-to-many alignments (as if to allow a single character from *Pride and Prejudice* to stand in for all the characters from *Jane Eyre* at once), and to make things worse, it sums over all the alignments, so that many fairly tenuous comparisons can "gang up" to render a pair of texts more similar than a single good comparison.

As discussed in section 2.3, the use of a matching can improve performance in such a system by forcing each character to map to only one other character. The mapping gives each *Pride and Prejudice* character a "best equivalent" character in *Jane Eyre* (ignoring "left-over" characters), e.g., Elizabeth mapping to Jane, Darcy to Rochester, and so forth. Following Matusov et al. (2004), the system computes an optimal bipartite matching between characters with the Hungarian algorithm, which works in polynomial time.

$$k^{char}_{symm}(X,Y) = \max_{\text{matching } F: u \leftrightarrow v} \sum_{u \in X} k(u, F(u)) \tag{7}$$

4.2 Parameter estimation

The function k described in Equation 5 is controlled by a parameter vector θ which controls the relative weight assigned to similarity with respect to each feature. Since some of the feature sets used here are relatively large (10 emotions and 10 LDA dimensions), some method for automatic parameter tuning is desirable. This section describes a training procedure, although its results show that it displays certain ambivalence.

The procedure, following Feng and Hirst (2012), is based on rankings. Since the system will be evaluated by using it to detect aberrant novels, the training objective should not depend on the absolute similarity values assigned to each novel pair, but on the difference between the scores $k(X,Y)$ and $k(X,Y')$ where Y' is some kind of aberrant text.

As in any learning task, a training set of novels $X_{1\ldots n}$ is used to fix the parameters. For each X_i, the training program constructs aberrant texts $X'_{i,1\ldots k}$, and assigns a rank to each aberrant ordering based on its dissimilarity to the original text. The two choices used for X' correspond to the two settings for synthetic-data experiments reported

below: *random permutations* and *reversals*. When training to optimize discrimination of random permutations, the system computes $k = 10$ random orderings of each training novel and ranks them by edit distance from the identity permutation.[4] When training to optimize reversals, it uses a single X'_1, the reversed novel, for each training novel.

Training instances are generated for each ordered pair X_i, X_j. The SVM-rank learner (Joachims, 2006) attempts to find parameters such that the original novel X_i is more similar to the real novel X_j than it is to the least aberrant permutation $X'_{j,1}$, more similar to $X'_{j,1}$ than to $X'_{j,2}$, and so forth.[5]

$$k(X_i, X_j) > k(X_i, X'_{j,1}) > k(X_i, X'_{j,2}) > \ldots > k(X_i, X'_{j,k})$$

As an example, consider the ordered pair *Pride and Prejudice* and *Jane Eyre* as X_i, X_j.[6] The system produces chapter-by-chapter permutations, ranked by edit distance from the original (schematically, these might be called *Jaen yrEe* and *naeE yJr*) and then attempts to achieve the ranking:

$$k(PP, Jane\,Eyre) > k(PP, Jaen\,yrEe) > k(PP, naeE\,yJr)$$

Optimization is somewhat more complicated a symmetrized system, because the matching acts as a latent variable, rendering the optimization non-convex (Yu and Joachims, 2009). The system employs an EM-like coordinate ascent procedure; it begins with an initial classifier, solves for the matching, and then reoptimizes the classifier. The character frequency features yield good results in the experiments discussed below, so the initial weights are set to *(character-frequency-gender-matched: 1, character-frequency-gender-mismatched: 1)*. In development experiments, running only a single iteration leads to the best results, but this may be a function of the small size of the training set.

5 Experimental setup

In the experiments in this section, the system is used to distinguish real novels in the test set (see the appendix) from artificial surrogates produced by permuting them. Two conditions are reported: *random permutations* and *reversals*. In each case, the permutations are performed chapter-by-chapter.

The experiments consider pairwise classifications in which the system is given access to a single training novel x along with a test pair

[4] Edit distance was one of the better-performing dissimilarity metrics in (Feng and Hirst, 2012).

[5] The SVM-rank learner uses a hinge loss function and a linear kernel.

[6] The pair *Jane Eyre* as X_i and *Pride and Prejudice* as X_j is a separate instance of the training set.

(y, y_{perm}), and asked to decide whether y or y_{perm} is the original. y is only selected as the original if $k(x, y) > k(x, y_{perm})$. Since this is a binary forced choice, a random baseline would score 50%.

The systems to be tested vary in three dimensions. They may compute a *single trajectory* of values for the entire novel or *character trajectories* for every character. They may use *sentiment* features or *LDA* features (or, in the case of character trajectories, they may use only the *frequency* with which each character appears). Finally, for character trajectory systems only, they may compare representations using a *symmetrized* or *unsymmetrized* kernel. The details of single and character-based representations appear in sections 3.1 and 3.2. The feature sets are described in section 3.1. Single-trajectory representations are always compared with the function k (Equation 5); the symmetrized kernel is k^{char}_{symm} (Equation 7) and the unsymmetrized one is k^{char} (Equation 6), both described in section 2.3.

There are 30 test novels, and thus $30 \times 29 = 870$ pairwise comparisons. To avoid over-analyzing miniscule differences in results due only to luck, a statistical test must be used to indicate when the gap between two systems is statistically significant. A variety of common tests cannot be used since they assume different test trials to be independent and identically distributed. That is not the case here since each test novel y participates in multiple comparisons, with different novels standing in as x. Instead, significance is assessed using a Monte Carlo permutation test with 100000 random permutations.[7] Differences between systems are reported as significant if they reach the $p < .05$ level—that is, if the probability that they are due only to chance is assessed as less than 5%.

6 Results

6.1 Accuracy scores

The accuracy scores of various single-trajectory systems appear in Table 3, and those of character-trajectory systems in Table 4. For instance, the first row of Table 3 shows that a representation based on "anger" as a proportion of all sentiment words in a chapter makes it possible to distinguish from the originals 74% of randomly permuted novels in the development set. Only 64% of reversed novels can be distinguished. Results in the test data are somewhat different, with 64% of permuted novels distinguishable and 62% of reversals. The last row of the table

[7] For kernels which can be represented as inner products, the *Maximum Mean Discrepancy* test (Gretton et al., 2007) is more suitable but, as discussed in section 2.3, the *max* operator in the symmetric kernel means that this condition does not hold.

	Dev. Order	Dev. Reverse	Order	Reverse
Anger	74	64	64	62
Anticipation	51	47	55	55
Disgust	70	51	62	53
Fear	66	59	71	65
Joy	43	52	51	51
Sadness	67	61	61	64
Surprise	40	66	60	55
Trust	43	73	60	53
Negative	63	51	65	60
Positive	40	63	60	65
Sentiment	65	66	57	65
LDA	78	83	67	$89\dagger^{sentiment}$
Sentiment/LDA	80	83	71	$88\dagger^{sentiment}$

TABLE 3 Development and test accuracy (%) for various single-trajectory systems using Equation 5. † (only computed on test) indicates a significant difference ($p < .05$).

	Order	Rev.
Character freq.	61	50
Character freq. (symm.)	$81\dagger^{freq,lda-symm}$	52
Char. freq./sentiment	53	50
Char. freq./sentiment (symm.)	$81\dagger^{sentiment,lda-symm}$	54
Char. freq./LDA	57	$59\dagger^{freq,sent}$
Char. freq./LDA (symm.)	$72\dagger^{lda}$	56
Freq./sentiment/LDA	55	57
Freq./sentiment/LDA (symm.)	$82\dagger^{combo,lda-symm}$	57

TABLE 4 Test accuracy (%) for character-trajectory systems. Non-symmetrized systems use Equation 6, symmetrized ones use Equation 7. † indicates a significant difference ($p < .05$).

		Order	Rev.
	Best (Elsner, 2012)	62	52
1	Single-traj. combined	$71\dagger^{4}$	$88\dagger^{3,4}$
2	Single-traj. LDA	67	89
3	Char.-traj. combined (symm.)	$82\dagger^{4}$	57
4	Char.-traj. LDA	57	59

TABLE 5 Comparison of best systems by accuracy (%) using single trajectory and character trajectory systems, with significance tests.

shows that a system combining all sentiment and LDA features (with learned weights) distinguishes 71% of randomly permuted novels in the test set and 88% of reversed ones from the original, and that it is significantly better than a system using only sentiment.

Comparing the results in Table 3 for single features (above the line) with those for learned combinations (below) shows that while training is generally effective, its results are not particularly impressive. This seems to reflect several issues. One is that novels are quite heterogenous, and parameters from the training set do not always work well in testing. For instance, on the development set, *trust* is a good indicator for only 43% of the random permutations but is correct for 73% of the reversals. On the test set, it is 60% accurate for random permutations and 53% for reversals.

Another issue is that, for mathematical reasons, the learning system aims only approximately to maximize the number of correctly ranked pairs in the training set.[8] This means that trained systems are not guaranteed to outperform their components, even on the training data. For instance, the learned *sentiment* trajectory system scores 65% on reversals, no better than *positive* or *fear* on their own.

Nonetheless, learning-based systems are capable of effective parameter tuning, especially using the LDA features. The LDA single-trajectory system, optimized for *reverse* permutations, scores 89%. Using uniform weights instead of optimization (not shown in Table 3), the result is significantly worse at 82%. Where the learning methods fail to improve over single-feature systems, it seems likely that the critical problem is insufficient training data.

Now, Table 4. Systems with character-specific trajectories perform worse on reversals than those using single trajectories. While the LDA-based single-trajectory system can distinguish these at the 89% level, the corresponding character trajectory system with LDA features scores only 59%. Higher performance, however, is possible for random orderings, on which simply comparing character frequencies scores 61%.

Using the symmetrization-by-matching technique (Equation 7) improves this ordering result substantially. All matching-based systems are significantly better at random orders than their un-symmetrized counterparts. Frequency alone is effective in 81%; sentiment and the combined model perform comparably, although they do not improve, possibly because the development set is too small to get good parameter

[8]The approximation used is described fully by Joachims (2006). In development experiments using a maximum entropy system, which uses a different approximation to the training error (Bishop, 2006, chapter 4), the trained system actually performed *worse* on the training set than a system with uniform weights.

estimates.

The results across different representations are reported in Table 5. It is clear that a single trajectory is better for reversals, with scores of 89% using LDA versus 59% for a character-trajectory system. Character-based systems are numerically better for orderings (82% using all features versus 71% for a single-trajectory system), although the difference does not reach significance. LDA features are more effective than sentiment overall, and combining the two feature sets seems to add relatively little—compare row 1 to 2 and row 3 to 4.

6.2 Comparison to (Elsner, 2012)

The earlier system presented in Elsner (2012) differed from this one in several ways. A few differences are minor: the name resolution algorithm presented here uses a better gender heuristic, and the definition of the basic trajectory comparison c (Equation 4) has been simplified by dropping a bag-of-unigrams feature set which appears to have little effect on performance. The basic character frequency system with these changes in place (Table 4) scores 61% on random orderings. This is comparable to the 60% reported by Elsner (2012) for a character-based kernel, suggesting that these simplifications have little effect.

The major additions are the use of Mohammad and Turney's (2010) sentiment lexicon rather than that in (Wilson et al., 2005), the use of LDA features, and symmetrization. Table 5 shows the results of the best earlier system described in (Elsner, 2012). That system is outperformed by the best systems described here, scoring 62% on orderings versus 82% with symmetrized character features and 52% on reversals versus 89% with LDA.

There is also a major subtraction. Elsner (2012) presented a *second-order* system; it takes character relationships into account, and outperforms the character-to-character systems. Once the new features have been incorporated, the second-order system performs worse than the basic systems. In development experiments (not shown in Table 5), adding the second-order relationship features to a system using character frequencies and symmetrization decreases the best ordering result from 81% to 78%. This suggests that although relationship information gives some accurate cues to structure, it can also be misleading and will need to be incorporated into realistic systems with care.

Finally, Elsner (2012) described a highly ineffective single-trajectory system (not shown in Table 5) which performed essentially at chance. He concluded that such systems were inferior to those with character information. This appears to be untrue in general. That single-trajectory system used as its only feature the proportion of "subjective" words

from Wilson et al. (2005). The results in Table 3 show single-trajectory that systems can be quite effective when given the right feature set. Better sentiment features score 57% on ordering and 65% on reversals, and LDA scores 67% on orderings and 89% on reversals.

7 Conclusions

Analysis of the synthetic permutation-discrimination task reveals several interesting facts. First, while using a larger sentiment lexicon definitely improves over a smaller one, sentiment words on their own are still a cue to plot structure less effective than LDA topics, especially for reversals. This suggests that the plot of a 19th-century novel is in fact tied closely to domain events such as, e.g., marriages—more than to sentiment patterns like happiness—and that the domain-based cues are particularly useful in detecting beginnings and endings.

On the other hand, symmetrized character frequency is, on its own, overwhelmingly the best indicator for distinguishing random permutations from real texts. That LDA models do not do especially well on this case indicates that the middle sections of a novel vary widely in terms of domain events (middle sections vary in their inclusion of events like travel, marriages of minor characters, or illness and death). What is more likely to stay constant is the way in which the narrative directs its focus toward main characters, while introducing minor characters who can attain importance for shorter periods before fading back into the background.

Constructing an explicit matching between characters is helpful for character-based systems, regardless of the feature set. Matchings allow the system to check that the two texts not only incorporate a similar set of emotions or actions, but also partition them among different characters in similar ways.

Character-specific emotions or LDA topics, however, do not seem to improve results by sharpening the model's ability to distinguish between character roles, at least as far as random permutations are concerned; on reversals, they improve significantly over character frequency alone, but the effect is not large. This suggests that, even in a character-based system, such features capture whole-story trends involving beginnings and ends. In other words, character frequency alone does a good job in distinguishing "choppy" from "fluent" patterns. All the same, specialized lexical frequencies (whether measured over the whole text or per-character) are better representations of the overall plot arc.

It seems possible that similarity measurement systems of this type

may eventually be useful in search and recommendation systems for fiction. This, however, would require additional resources beyond those presented here. The disappointing performance from parameter tuning suggests that larger datasets for artificial reordering might be useful, since they could allow the system to use its existing feature set more productively, or to work effectively with more features. Even so, while artificial tasks are likely to remain helpful for cheaply ruling out bad representational choices during development, they are probably not sufficient to fine-tune a system that must measure similarity between pairs of real novels rather than between real and reordered ones. Training data for this scenario are likely to require effort from human annotators. Future work with a practical recommendation system may help to clarify how well systems like the current work can benefit the production of a truly effective system for novelistic texts.

Acknowledgments

I am grateful to the editors of this volume, especially Stan Szpakowicz, and to three anonymous reviewers for their comments and suggestions. The discussion of related work owes much to suggestions by Diane Litman, Janice Wiebe and Robyn Warhol.

References

Alm, Cecilia Ovesdotter and Richard Sproat. 2005. Emotional Sequencing and Development in Fairy Tales. In *ACII*, pages 668–674.

Ang, Robert. 2012. *The Writer's Toolkit*. Master's thesis, University of Edinburgh.

Bamman, David, Brendan O'Connor, and Noah A. Smith. 2013. Learning Latent Personas of Film Characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361. Sofia, Bulgaria: Association for Computational Linguistics.

Bamman, David, Ted Underwood, and Noah A. Smith. 2014. A Bayesian Mixed Effects Model of Literary Character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379. Baltimore, Maryland: Association for Computational Linguistics.

Barzilay, Regina and Mirella Lapata. 2005. Modeling Local Coherence: an Entity-Based Approach . In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Bhattacharya, Indrajit and Lise Getoor. 2005. Relational clustering for multi-type entity resolution. In *Proceedings of the 4th international workshop on Multi-relational mining*, MRDM '05, pages 3–12. New York, NY, USA: ACM. ISBN 1-59593-212-7.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. ISBN 0387310738.

Blei, David, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3:2003.

Blei, David M. and John D. Lafferty. 2006. Dynamic Topic Models. In *ICML*.

Boughorbel, Sabri, Jean-Philippe Tarel, and Francois Fleuret. 2004. Non-Mercer Kernels for SVM Object Recognition. In *BMVC*, pages 1–10.

Chambers, Nathanael and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and their Participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610. Suntec, Singapore: Association for Computational Linguistics.

Chang, Jonathan, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *Neural Information Processing Systems*.

Charniak, Eugene. 2001. Unsupervised learning of name structure from coreference data. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics (NACL-01)*.

Coll Ardanuy, Mariona and Caroline Sporleder. 2014. Structure-based Clustering of Novels. In *Proceedings of Computational Linguistics for Literature (CLFL)*. Gothenburg, Sweden.

Crane, R.S. 2002. The Concept of Plot and the Plot of Tom Jones. In B. Richardson, ed., *Narrative dynamics : essays on time, plot, closure, and frames*. The Ohio State University Press.

Elsner, Micha. 2012. 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. Avignon, France: Association for Computational Linguistics.

Elsner, Micha, Joseph Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *Proceedings of HLT-NAACL '07*.

Elson, David, Nicholas Dames, and Kathleen McKeown. 2010. Extracting Social Networks from Literary Fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147. Uppsala, Sweden: Association for Computational Linguistics.

Elson, David K. and Kathleen R. McKeown. 2010. Building a Bank of Semantically Encoded Narratives. In N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, eds., *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Feng, Vanessa Wei and Graeme Hirst. 2012. Extending the Entity-based Coherence Model with Multiple Ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 315–324. Avignon, France: Association for Computational Linguistics.

Finlayson, Mark A. 2009. Deriving narrative morphologies via analogical story merging. In *New Frontiers in Analogy Research: Proceedings of the Second International Conference on Analogy*, pages 127–136. Sofia, Bulgaria: New Bulgarian University Press.

Goyal, Amit, Ellen Riloff, and Hal Daume III. 2010. Automatically Producing Plot Unit Representations for Narrative Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 77–86. Cambridge, MA: Association for Computational Linguistics.

Gretton, Arthur, Karsten M. Borgwardt, Malte Rasch, Bernhard Schlkopf, and Alexander J. Smola. 2007. A Kernel Method for the Two-Sample-Problem. In B. Schlkopf, J. Platt, and T. Hoffman, eds., *Advances in Neural Information Processing Systems 19*, pages 513–520. Cambridge, MA: MIT Press.

Gnen, Mehmet and Ethem Alpaydın. 2011. Multiple kernel learning algorithms. *The Journal of Machine Learning Research* 12:2211–2268.

Haussler, David. 1999. Convolution Kernels on Discrete Structures. Tech. Rep. UCSC-CRL-99-10, Computer Science Department, UC Santa Cruz.

Huff, Jason. 2010. *Autosummarize*. McNally Jackson Books. http://jasonhuff.com/projects/autosummarize/.

Joachims, Thorsten. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Karamanis, Nikiforos, Massimo Poesio, Chris Mellish, and Jon Oberlander. 2004. Evaluating Centering-Based Metrics of Coherence. In *ACL*, pages 391–398.

Kazantseva, Anna and Stan Szpakowicz. 2010. Summarizing short stories. *Computational Linguistics* pages 71–109.

Kim, Dae Il and Erik B. Sudderth. 2011. The Doubly Correlated Nonparametric Topic Model. In *NIPS*, pages 1980–1988.

Lehnert, Wendy. 1981. Plot Units and Narrative Summarization. *Cognitive Science* 4:293–331.

Li, Boyang, Stephen Lee-Urban, Darren Scott Appling, and Mark O Riedl. 2012. Crowdsourcing narrative intelligence. *Advances in Cognitive Systems* 2:25–42.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Matusov, Evgeny, Richard Zens, and Hermann Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 219. Association for Computational Linguistics.

McCallum, Andrew. 2002. MALLET: A Machine Learning for Language Toolkit.

McIntyre, Neil and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 217–225. Association for Computational Linguistics.

McIntyre, Neil and Mirella Lapata. 2010. Plot Induction and Evolutionary Search for Story Generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572. Uppsala, Sweden: Association for Computational Linguistics.

Miller, G., A.R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1993. Introduction to WordNet: an on-line lexical database. Tech. rep., Princeton University.

Mimno, David and David Blei. 2011. Bayesian Checking for Topic Models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 227–237. Edinburgh, Scotland, UK.: Association for Computational Linguistics.

Mohammad, Saif. 2011. From Once Upon a Time to Happily Ever After: Tracking Emotions in Novels and Fairy Tales. In *Proceedings of the 5th*

ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities, pages 105–114. Portland, OR, USA: Association for Computational Linguistics.

Mohammad, Saif and Peter Turney. 2010. Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Los Angeles, CA: Association for Computational Linguistics.

Mohammad, Saif M. 2012. From once upon a time to happily ever after: Tracking emotions in mail and books . *Decision Support Systems* 53(4):730 – 741.

O'Keefe, Timothy, Silvia Pareti, James R. Curran, Irena Koprinska, and Matthew Honnibal. 2012. A Sequence Labelling Approach to Quote Attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 790–799. Jeju Island, Korea: Association for Computational Linguistics.

O'Neill, Brian and Mark Riedl. 2011. Toward a computational framework of suspense and dramatic arc. In *Affective Computing and Intelligent Interaction*, pages 246–255. Springer.

Phelan, James and Peter J. Rabinowitz. 2012. Narrative as Rhetoric. In D. Herman, J. Phelan, P. J. Rabinowitz, B. Richardson, and R. Warhol, eds., *Narrative Theory*. The Ohio State University Press.

Post, Matt. 2011. Judging Grammaticality with Tree Substitution Grammar Derivations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 217–222. Portland, Oregon, USA: Association for Computational Linguistics.

Propp, Vladimir. 1968. *Morphology of the Folktale*. University of Texas Press, 2nd edn.

Ryan, Marie-Laure. 1991. *Possible worlds, artificial intelligence and narrative theory*. Bloomington: Indiana University Press.

Salway, Andrew and David Herman. 2011. Digital Corpora as Theory-building Resource. In R. Page and B. Thomas, eds., *New Narratives: Stories and Storytelling in the Digital Age*, pages 120–137. University of Nebraska.

Schank, Rogert and Robert Abelson. 1977. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ.: Lawrence Erlbaum Associates.

Socher, Richard, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Edinburgh, Scotland, UK.: Association for Computational Linguistics.

Vishwanathan, S. V. N., Nicol N. Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *The Journal of Machine Learning Research* 11:1201–1242.

Volkova, Ekaterina P., Betty Mohler, Detmar Meurers, Dale Gerdemann, and Heinrich H. Blthoff. 2010. Emotional Perception of Fairy Tales: Achieving Agreement in Emotion Annotation of Text. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 98–106. Los Angeles, CA: Association for Computational Linguistics.

Wallace, Byron. 2012. Multiple Narrative Disentanglement: Unraveling Infinite Jest. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10. Montréal, Canada: Association for Computational Linguistics.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354. Vancouver, British Columbia, Canada: Association for Computational Linguistics.

Yessenalina, Ainur and Claire Cardie. 2011. Compositional Matrix-Space Models for Sentiment Analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Edinburgh, Scotland, UK.: Association for Computational Linguistics.

Yu, Chun-Nam John and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1169–1176. New York, NY, USA: ACM. ISBN 978-1-60558-516-1.

Appendix

19th century novels used in our study

Development set (20 works)	
Austen	*Emma, Mansfield Park, Northanger Abbey, Persuasion, Pride and Prejudice, Sense and Sensibility*
Blackmore	*Lorna Doone*
Brontë, Emily	*Wuthering Heights*
Burney	*Cecilia* (1782)
Carey	*Heriot's Choice*
Caird	*The Daughters of Danaus*
Cholmondeley	*Red Pottage*
Conrad	*Lord Jim*
Corelli	*A Romance of Two Worlds, The Sorrows of Satan*
Hardy	*Tess of the D'Urbervilles*
James	*The Ambassadors*
Scott	*Ivanhoe*
Yonge	*The Heir of Redclyffe*
Test set (30 works)	
Braddon	*Aurora Floyd*
Brontë, Anne	*The Tenant of Wildfell Hall*
Brontë, Charlotte	*Jane Eyre, Villette*
Bulwer-Lytton	*Zanoni*
Disraeli	*Coningsby, Tancred*
Edgeworth	*The Absentee, Belinda, Helen*
Eliot	*Adam Bede, Daniel Deronda, Middlemarch*
Gaskell	*Mary Barton, North and South*
Gissing	*In the Year of Jubilee, New Grub Street*
Hardy	*Far From the Madding Crowd, Jude the Obscure, Return of the Native, Under the Greenwood Tree*
James	*The Wings of the Dove*
Meredith	*The Egoist, The Ordeal of Richard Feverel*
Scott	*The Bride of Lammermoor*
Thackeray	*History of Henry Esmond, History of Pendennis, Vanity Fair*
Trollope	*Doctor Thorne*