Multiple Narrative Disentanglement:
identification of narrative topics and clusters in *Infinite Jest*

Maggie Engler, Brett Harvey — December 16, 2016

I. Introduction

Since ancient times, humans have told stories. We have used them to explain natural phenomena, memorialize events, and express ideas; it is perhaps our most natural form of communication. However, many stories do not fit easily into the standard linear narrative structure. More complex texts usually have multiple narrative threads interwoven, like novels that switch perspective, histories that are broad in scope, or even everyday conversations that jump around from topic to topic.

We are interested in performing multiple narrative disentanglement by segmenting these texts and reconstructing the separate narrative threads. In practice, multiple narrative disentanglement could be useful for automatic plot summarization, thematic reorganization of a chronological text, and other literary analysis. It could help to derive new understanding of the text and its composition, including which characters or locations are most strongly associated with each narrative.

There are a number of possible approaches to this task, drawing from both supervised and unsupervised learning. We explore several different techniques and demonstrate them on *Infinite Jest*, the nearly 1100-page magnum opus of the late David Foster Wallace. *The New York Times* wrote of the novel that "Its size and complexity make it forbidding and esoteric," making it an ideal candidate for multiple narrative disentanglement.

II. Related Work

The term and concept of multiple narrative disentanglement was introduced in 2012 by Byron Wallace, then a doctoral student at Tufts University [1]. In addition to introducing the task, Wallace describes basic principles of narratology theory and applies them to extract entities from the text, then uses latent Dirichlet allocation (LDA) for the disentanglement. Wallace also made available a manually annotated dataset, including the markup of *Infinite Jest* and its corresponding narrative tags that we used in our supervised learning approaches.

A related task, chat disentanglement, was introduced in 2006 by a group from Hong Kong University of Science and Technology [2] in order to detect threads in dynamic text message streams. The group describes the task as similar to topic detection, with a few distinctions, most notably that the temporal aspect of the message stream data plays a large role. They take an unsupervised approach, comparing the success of several different clustering algorithms on their message representations. Another group, from Brown University, frames chat disentanglement as a graph partitioning problem [3].

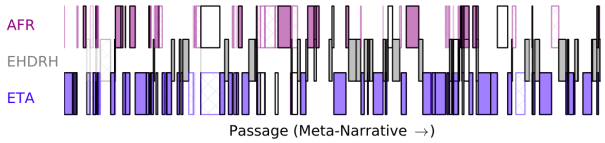

Fig. 1. This timeline graphic generated by Byron Wallace illustrates how three of the most central narrative threads of *Infinite Jest* interweave throughout the novel. [1]

However, narrative and chat disentanglement are not exactly the same, because narratives may intersect, requiring multi-label classification, and narratives are more hierarchical. Taking advantage of this structure, a group from Stanford University [4] present an integration of narrative event chains and narrative schemas into the unsupervised learning algorithm as a means of improving performance.

III. Data and Features

The annotated *Infinite Jest* data file obtained from Byron Wallace was in plaintext format with HTML-style tags denoting the beginning and ending of segments belong to particular narratives. For example, an excerpt about Hal Incandenza, one of the major characters, might be tagged with the label INC, for the Incandenza family narrative thread, like so:

> <INC> Hal some weeks back had acquiesced to Lyles diagnosis that Hal finds Ingersoll this smart soft caustic kid, with a big soft eyebrowless face and unwrinkled thumb-joints, with the runty, cuddled look of a Mamas boy from way back, a quick intelligence he squanders on an insatiable need to advance some impression of himself that the kid so repels Hal because Hal sees in the kid certain parts of himself he cant or wont accept. </INC>

As a preprocessing step, we traverse the text file and create a dictionary which maps segments to their appropriate labels. For feature extraction, we use a bag-of-words model on the segments, creating a matrix of word frequencies and a corresponding vector of labels. We randomly shuffle the rows of the matrix, each representing one example, and then split them into a training set and a test set.

We also create labels for our training set by picking a tag for each example based on the frequency of occurrence of each tag. For example, if we wish to classify our segments into 10 different classes, we search for the 10 most common tags and for each segment we choose one of these tags if it applies to the given segment, otherwise the given label is none.

IV. Supervised Methods

Using these feature vectors and labels, we trained both a Naïve Bayes classifier and a Support Vector Machine and compared their performance on the test set. Recall that the Naïve Bayes algorithm makes the very strong (but apparently rather accurate) assumption that the words in a segment are conditionally independent given a tag (label). The tag probabilities are modeled via a multinomial and thus, performing the Naïve Bayes algorithm amounts to maximizing the joint likelihood

$$\prod_{i=1}^{m} p(x^{(i)}, y^{(i)}).$$

Also recall that via the representer theorem, we can think of any discriminative algorithm as a choice of loss and a choice of kernel. For our SVM, we use the hinge loss $L(z, y) = [1 - zy]_+$ and the radial basis function for our kernel. Thus, our Support Vector Machine algorithm amounts to minimizing the regularized risk

$$J_\lambda(\alpha) = \frac{1}{m}\sum_{i=1}^{m}\left[1 - y^{(i)}K^{(i)^T}\alpha\right]_+ + \frac{\lambda}{2}\alpha^T K\alpha,$$

where K is the radial basis function kernel, α is a vector of weights, and λ is a tunable hyperparameter.

The Naïve Bayes classifier results in a slightly better generalization error for every feature extraction method we investigated (especially when using hard counts for the bag of words). In particular, using the named entities feature extraction method resulted in a test error of about 12%. Given the simplicity of our model, we found this to be quite impressive. Quantitatively comparing the various feature models in a supervised setting proved to be quite useful in determining which feature models to use for our unsupervised methods. Below, we describe the modifications we made to our original bag-of-words feature model in order to improve performance.

A. Increased Number of Training Examples

Though this did not result in an explicit modification to our feature model, one of the potential issues we initially identified with this particular task was that the tagged segments, used as training examples, numbered only about four hundred. In fact, in narrative disentanglement, it is expected to have a fairly small number of segments, even despite the length of *Infinite Jest*. Because the tagged segments were labeled by hand, and made up the entire text corpus, we werent able to generate any new segments, so to increase the number of training examples, we randomly sampled substrings of 1000-character length from our existing segments. We tagged these newly sampled segments with the appropriate tags based on the segment from which they originated.

Unfortunately, this approach did not improve our test error rates at all. We hypothesized that this might be the case since the vectorized versions of our examples in the small sample size case could be expressed as a linear combination of several examples from the large sample size case.

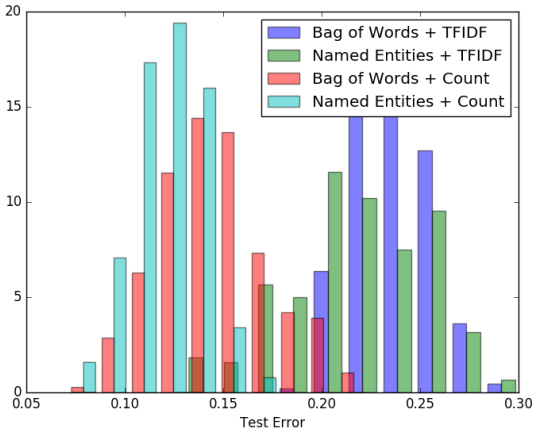

Fig. 2. Each histogram shows the test error of our Naïve Bayes classifier over 250 trials using the different types of feature extraction indicated by the legend. Most notably, using the named entities feature extraction with a hard count resulted in a median test error of 12%.

B. Stemming

By performing stemming on the text corpus of *Infinite Jest*, we hoped to eliminate some of the redundancies in the data, such as every conjugation of a single verb being represented in the models as a distinct word. Therefore, we used a Lancaster stemmer from the Natural Language Toolkit [5] to remove the morphological endings of words in the text, while leaving the tags unchanged. We selected a Lancaster stemmer rather than a Porter stemmer or Snowball stemmer because the Lancaster stemmer is more aggressive in its stemming than either of the other two, resulting in a lower number of unique words. The original text had 30,000 unique words, and the Lancaster stemming algorithm performed best at reducing the set of unique words to a set of unique word stems.

C. Named Entity Recognition

To eliminate even more of the uninteresting text from the corpus, we also implemented Named Entity Recognition (NER) on the corpus of *Infinite Jest*. To perform NER, we first extracted all of the named entities from the text, using the Natural Language Toolkit package for Python, and saved the named entities in a list. Then, when we parsed the text corpus to create our segments for training and testing, we removed all terms that did not belong to the list of named entities. Finally, we applied a Vector Space Model in two different ways, both with each dimension corresponding to a named entity. In the first we used a Count Vectorizer, so that each training example becomes a vector such that the value at index i is the count of the ith named entity in that training example. In the second, we used a Term-Frequency Inverse Document-Frequency (TFIDF) Vectorizer, where the counts in each training example are normalized by the total number of occurrences of each corresponding named entity.

D. Results

Above, in Figure 2, we present several overlaid histograms of test errors for the four different methods of feature extrac-

Feature Model	NB Error	SVM Error
Regular + Count	$14.7 \pm 2.7\%$	$23.4 \pm 2.1\%$
Regular + TFIDF	$23.5 \pm 1.9\%$	$23.6 \pm 1.9\%$
Stemmed + Count	$14.4 \pm 3.0\%$	$23.6 \pm 2.2\%$
Stemmed + TFIDF	$23.8 \pm 2.0\%$	$23.6 \pm 1.9\%$
Named Entity + Count	$12.2 \pm 1.8\%$	$20.2 \pm 2.0\%$
Named Entity + TFIDF	$22.1 \pm 3.7\%$	$23.7 \pm 1.9\%$

Fig. 3. Table showing test errors for Naïve Bayes and SVM on various methods of feature extraction. Test errors for the cases where we artificially increased the number of samples are not shown since they were not substantially different.

tion indicated in the legend. The bimodal distribution shows the relative improvement of a word-frequency method over a term-frequency/inverse document-frequency (TFIDF) method. We had assumed that TFIDF would produce the better test error, but since our results show otherwise, it is possible that segment length is not independent of the particular narrative thread. Moreover, the small difference in peaks in the two modes of the distribution show how using named entity extraction improved our test error. Our mean test errors and their spreads are listed in the table below (Figure 3) for both the NB and SVM classifiers for all feature models.

V. Unsupervised Methods

Using the same bag of words model, we also investigated unsupervised methods, in particular Latent Dirichlet Allocation (LDA) and k-means clustering.

A. Latent Dirichlet Allocation

Latent Dirichlet Allocation is an unsupervised learning technique used to discover topics from groups of documents. The model regards each document as a mixture of topics, with each topic distributed according to a Dirichlet prior distribution. In effect, LDA maximizes the total probability for each document (or segment, in our case) according to

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\varphi_{Z_{j,t}}),$$

where W represents the words in the documents, Z represents the topic each word is associated with, θ represents the distribution of topics in the document, and α and β are vectors of weights. In this way we can determine each documents unique mixture, which results in a document-topic matrix with one row for each segment. Our implementation of LDA is from the Python package `scikit-learn` [6].

When we performed Latent Dirichlet Allocation on our data, we made several interesting qualitative observations. Before removing common words and named entities, LDA seems to separate segments not into their narrative threads, but conflates male characters and female characters because of the prevalence of gendered pronouns. For example, five top words from one topic are "she", "her," "Joelle," "Avril", and "moms", while the words from another topic are "he," "his," "(Don) Gately", "(Hugh) Steeply," and "we". We achieved much better results for our purposes once we extracted the named entities using the same procedure from our supervised learning methods. The table of top 10 words for each "topic" found by LDA are listed below in Figure 4. Topic 1 aligns perfectly with the tag ETA, for Enfield Tennis Academy, as all names are characters who live at the academy. Topic 2 similarly matches the tag EHDRH, for Ennet House Drug and Alcohol Recovery House; many of these terms refer to characters who live at the house, and others are closely related to residents. The third topic, Topic 3, is not as clear—there is some overlap with the other topics, and also some miscellaneous characters who dont clearly belong to any of the larger narratives, such as Poor Tony Krause, a drag queen. This "topic" might correspond to the k-means cluster with the greatest spread, as the shared features of the group seem to not be as strong.

Topic 1	Topic 2	Topic 3
Hal	Gately	Mario
Pemulis	Orin	Schtitt
Mario	Lenz	Hal
Stice	Joelle	Blott
Wayne	Green	Poor
Incandenza	House	Tony
Schacht	Marathe	Millicent
Tavis	Himself	Kent
DeLint	Madame	Ingersoll
Schtitt	Steeply	Jim

Fig. 4. Table showing the topic organization output by our application of LDA to our dataset. Clearly, Topic 1 is the most well-defined since every common word belonging to this topic is associated with the Enfield Tennis Academy. Topics 2 and 3 are less well-defined.

Quantitatively, we evaluated our LDA model by using the document-topic matrix as an input to our Nave Bayes classifier. It performed with an average error rate of 0.31 as compared to 0.24 with the original labels, suggesting that most of the features of the original dataset were captured by the LDA model.

B. k-Means Clustering

The k-means clustering algorithm is an unsupervised learning algorithm designed to cluster the data into k clusters, minimizing the distance from each data point to the centroid of its cluster. In mathematical terms, the objective of the algorithm is to minimize

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

over set S, by initializing the centroids each to some random data point from S, labeling each point by its closest centroid, and iterating by calculating the new centroid of each cluster. We used the implementation of k-means from the Python package `scikit-learn` [6].

To evaluate our k-means clusters quantitatively, we gave each cluster found by the algorithm the closest corresponding tag. We then used the tags assigned by k-means as the

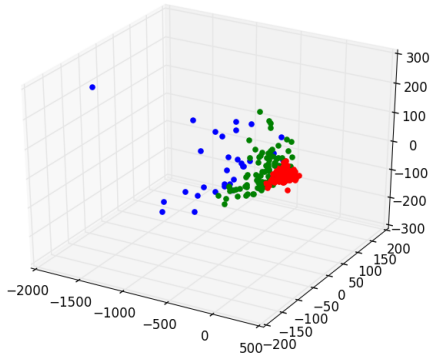

Fig. 5. Graph of the three clusters that the k-means algorithm determined, using a bag-of-words model for feature extraction. We used Primary Component Analysis (PCA), with the axes representing the first through third primary components

labels for our training data and tested them again using the Naïve Bayes classifier for comparison. K-means did not do particularly well with the bag-of-words model, with an average error rate of 0.53 over 100 trials, as compared to 0.24 test error with the original labels (although still significantly better than the expected error for random guessing, 0.66). Its possible that the algorithm suffered from some of the same challenges as LDA. Once we extracted the named entities and performed the same test, the k-means test error reached as low as 0.2 with an average of 0.32, outperforming random assignment by 100% but not quite reaching the generalization error of the original training set labels. As illustrated in Figure 5, we observed that some of the clusters were not as tightly clustered as others, so it is likely that some of these data points are misclassified by the k-means algorithm, which would induce additional error in classification.

VI. Conclusion

Although we achieved fairly good rates of success in reassembling the narrative threads, the highest being 88% on average with a Naïve Bayes classifier on the extracted named entities, we identified many possible areas of improvement and further exploration. Given more time, for higher performance on our existing models, we would like to tune our models by trying different parameters—for example, perhaps a different kernel would reduce the error of the Support Vector Machine. We could also compare other learning algorithms, such as neural networks, to the ones that we used. Instead, much of our work was focused on improving the features selected from the data we had.

Of course, to explore narrative disentanglement more fully, we would need to look at more data and datasets, particularly in different genres. We have identified many potential applications for the task, such as rearranging history textbooks by theme, or reconstructing personal narratives and meaningful events using transcripts from spoken therapy sessions. With more tagged data and refinement of techniques, multiple narrative disentanglement could become a meaningful tool in natural language processing.

References

[1] B. C. Wallace, "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*, pp. 1–10, Association for Computational Linguistics, 2012.

[2] D. Shen, Q. Yang, J.-T. Sun, and Z. Chen, "Thread detection in dynamic text message streams," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 35–42, ACM, 2006.

[3] M. Elsner and E. Charniak, "Disentangling chat," *Computational Linguistics*, vol. 36, no. 3, pp. 389–409, 2010.

[4] N. Chambers and D. Jurafsky, "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: Volume 2-Volume 2*, pp. 602–610, Association for Computational Linguistics, 2009.

[5] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.