Visual Summarization of Fairy Tales through Automated Tagging of Beats

Dustin Janatpour - djanatpour@stanford.edu

Abstract

We present a framework for taking a fictional text which has been broken into a sequence of "beats"–overlapping chunks of a few sentences at a time which span the entirety of that text– and automatically applying labels to characterize those beats. Armed with such labels as positivity/negativity, romance, suspense, friendliness, and confrontration, the system can produce a visualization which serves as a summary of the story; these visual "fingerprints" can be used to understand the nature of a given story at a glance. The system is agnostic to the features and classifiers used to produce the labellings, and with minimal extension can be made to present the results of a beat-by-beat topic model. Here, we give a worked example of the system's application to the analysis of Grimm's Fairy Tales using a number of simple feature sets and machine learning techniques.

1 Introduction

There is considerable interest in producing systems which can automatically summarize or tag documents for quicker at-a-glance information consumption. The former goal is to condense texts into a form which captures the majority of content but omits much detail. The latter is to produce a small number of keywords which can be used to easily index, categorize, or understand the essential concepts covered in a text. A shared goal of such systems is to give a human reader a strong idea of what a text is about with a minimum of information.

Narrative works, unlike documents in general, have an internal temporal structure which can be leveraged in this summarization process. Narratives are often described as having "arcs," which can be thought of as the ups and downs of various factors in the story, such as positivity, romance, or suspense.

It is this property of temporality in narrative works which makes them lend themselves well to being summarized visually. In particular, a system which can characterize each moment of a work, rather than just a work as a whole, provides a unique means of visually representing a linear narrative as the interaction of various degrees of genre/plot building blocks such as romance and suspense. To avoid confusion with the mathematical notion of moments, we will refer to a short segment of a narrative which can be so characterized as a "beat" (as in a dramatic beat), and now have a clearly defined task: to produce a system which can, for each beat in a narrative work, provide a short list of properties from a set of possible properties, and then use the resulting sequence of property lists to visually represent the work.

The parallel task to the assembly of this system is the selection of models and algorithms which perform this characterization. There are myriad ways to featurize a beat, as a beat is simple a short (3 sentences for the purpose of this paper) piece of text. Under any such feature function, there are a number of algorithms which lend themselves to the task: it can be framed as simple classification of each property as present or not-present, in which case the full range of binary classification algorithms are available. That is the approach taken in this paper, and a number of simple classfication algorithms including Perceptrons and Naive Bayes are used and evaluated to demonstrate how the viualizer operates. However, it is important to note that any algorithm which can produce a list of properties is appropriate. In particular, Latent Dirichlet Allocation has theoretical value for this task and is discussed in the section on future work.

2 Background

Automatic summarization is a central task in computational semantics, and one which, as it is improved, will add greatly to humans' capacity to rapidly access the information stored in natural language docu-

ments[1]. Automated tagging and topic modeling have likewise been extensively studied as they increase the efficacy and efficiency of information retrieval as well as recommender systems[2].

Looking outside the field of NLP research, we find that the description of documents, in particular those which contain linear narratives, as a sequence of moments with quantitative properties is not novel. Indeed, a common practice in grade school discussions of stories is to draw a curve of tension over time: it starts low at the exposition, then rises with the rising action until it peaks at the climax and falls again into the resolution.

Robin D. Laws, a writer and designer of games, gives an extended treatment of this notion in his text, Hamlet's Hit Points[3]. In this text, he uses Shakespeare's Hamlet and number of other examples to demonstrate the use of dramatic beats for good narrative design. He treats each work as a series of beats which can either improve, worsen, or keep stable the situation of the central characters. At the end of each such analysis, we are left with a curve which serves as the literal arc of each story.

While such curves don't explain what happens in the story, they do give a strong indication of the story's tone over time. Hamlet, for example, finds his situation getting slowly worse over time until he comes up with a scheme to have his vengeance, after which point there is a slight improvement until Claudius and brutal irony conspire to give him a bad ending. The plot of the story cannot be described thoroughly in the curve, but its dramatic arc can.

Fairy tales, with their linear narratives, lend themselves well to such representations. We go a step beyond just the representation of "better" or "worse" by attempting to assign the presence or absence of a number of dramatic properties common across fairy tales; this effort forms the basis of our data collection.

3 Dataset

The data include labelled and unlabelled beats extracted from Project Gutenberg's HTML edition of Grimm's Fairy Tales. We used 61 tales and from these produced 3381 three-sentence beats, with a one-sentence step (so a beat can overlap with up to four others). The unlabelled beats were taken out of context and presented along with the titles of their origin stories to human labelers using a console interface. The interface asked if the human believed the beat was "positive, negative, scary/suspenseful, confrontational, romantic, or friendly"; it could be any combination of those things. Note that under this scheme a beat can be both positive and negative, or it can be neither.

We labelled 500 examples in this way. The labelled examples span 59 of the stories.

Label	# in 500 Examples
Positive	248
Negative	241
Confrontational	284
Suspenseful	257
Romantic	124
Friendly	187

An example, with its label, may look as follows:

Title THE ROBBER BRIDEGROOM

Beat Then the girl told her father all that had happened.\nThe day came that had been fixed for the marriage. The bridegroom arrived and also a large company of guests, for the miller had taken care to invite all his friends and relations.

Labels r(omantic), f(riendly)

These data are messy and, due to being handlabelled and sparse, serve only as a starting point. Still, they are effective for demonstrating the value of our system and elucidate a number of challenges that must be overcome if it is to be more effective.

4 Models and Algorithms

At a high level, our desired input is a story and our output is a sequence of lists of properties corresponding to each beat in that story (ultimately, a visual representation of that list). However, we have fixed pre- and post-processing steps: the story will be broken into three-sentence beats which will be marked with the title, and the set of property lists generated for a story will be ordered and assembled into a visual representation. Thus, when considering models and algorithms, we need only concern ourselves with ways to represent a beat and algorithms which can act on those representations to determine the presence or absence of the properties discussed in the dataset section.

The discussion here assumes a classifier-based approach; a broader investigation would include a rule-based system as well as continuous models such as true Bayesian models.

[1] Chris Manning, CS224N Lecture 18: Computational Semantics, 2013.

[2] Spence Green, CS224N Lecture 16: Topic Models, 2013.

[3] Robin D. Laws, Hamlet's Hit Points. Published by Gameplaywright Press, 2010.

4.1 Models and Features

The representations of a beat come down to the feature function used to turn the beat into something classifiable. Here, we only consider two feature functions: the bag-of-words model, and a subspace of it, the bag-of-adjectives model. Others considered but not yet tested include higher-order n-grams as well as dependency parse features to draw attention to verbs.

4.1.1 Bag of Words

This feature function simply removes stop words and punctuation and performs a count of remaining words. Such a model posits that a human classifying a beat will at least give some weight to the mere presence or absence of some words; "love" may imply romance, "sad" may imply negativity, and "anticipation" may imply suspense. A concern with such a model is that it cannot capture negation (we need at least bigrams to do so), and that it already produces a very high-dimensional space relative to the number of the training examples, which are in turn very sparse due to their brevity.

4.1.2 Bag of Adjectives

To combat the issues associated with the high dimensionality of the bag of words approach, we try an alternative by limiting our features to counts of adjectives using a list of about 8000 adjectives. This restricts the posible number of dimensions; however, we find that in the resulting space the vectors are still very sparse.

4.1.3 Other Features (Untested)

Any feature function which maps from text to a map of counts (include 0 or 1 for indicator features) would work. Thus, the set of possible features is as unbounded as it is for other NLP problems. The brevity of beats means that in future work it will better to investigate the features used for very short texts, such as Twitter posts, than those that are used for e-mails (as in the case of the bag of words).

Another consideration is that different features could be computed for different properties being analyzed. This especially would make sense when converting manually crafted rules into indicator features; we might want to know if "he loves her" when analyzing romance but not care about it for analyzing friendship.

4.2 Algorithms

The algorithms tested are very simple and include only a baseline classifier, a binary Perceptron, and a Naive Bayes Classifier. In every case, the presence or absence of each of the six properties is considered independently, so we are able to use the binary versions of each classifier. This is not necessary in general–algorithms which return a set of properties are perfectly usable, though they are not tested here.

4.2.1 Baseline

The baseline classifier simply counts the most common value for each property and assigns all examples to have the associated labels. A problem with this is that it can assign all examples to a label grouping that doesn't exist; if a majority are friendly, a majority are negative, and a majority are romantic, all beats will be declared friendly, negative, and romantic even if none of them are actually all three of these. Fortunately, it is just a baseline.

4.2.2 Perceptron

The Perceptron algorithm is a basic linear classifier which is looks for a separating hyperplane associated to each of the six properties. Unlike a support vector machine, the perceptron does not maximize the margin around this plane, so it is not very robust to new examples. In addition, using the perceptron supposes that the examples are linearly separable in the feature space, which is not a very good theoretical assumption. However, due to the sparseness of the training data, it seems that they are at least linearly separable in the training step (we see the problems with this during evaluation).

4.2.3 Naive Bayes

The Naive Bayes classifier assumes that each feature is independent and drawn from either a presence or absence distribution. For example, we compute the probability that we will see "love" in romantic beats and the probability that we will see in non-romantic beats. Given these feature probabilities and the prior distributions of each class, we classify new examples by comparing the probability of seeing its features given its membership in each class weighted by the prior probability of belonging to that class.

The theoretical justification is that, when a person reads a beat, they come to it with a context of what it means for a beat to be positive, friendly, and so on, then ask themselves if it is likely that they would see what they are seeing if this beat in fact belongs to each of those categories. The independence assumption is surely incorrect for many cases–we are unlikely to produce a collection of independent features unless we first run another algorithm to deliberately transform the feature space–but the simplicity of the algorithm and its clear mapping to human intuition makes it appealing.

5 Results and Evaluation

The high dimensionality of the feature spaces used relative to the length and number of training examples seems to have resulted in low stability and substantial overfitting. We first give a quantitative report of results for various combinations of features, then show some examples for qualitative analysis. Finally, we show the visual output of our system to demonstrate its value in spite of the performance shortcomings.

Note the properties are p(ositive), n(egative), c(onfrontational), s(uspensful), f(riendly), and r(omantic), and precision is given as 1.0 for no positives reported.

5.1 Comparison of Algorithms, Trained on 300, Tested on Another 100

Baseline, Train

	P	R	F1
p	1.0	0.0	0.0
n	.52	1.0	.68
c	.59	1.0	.74
s	.52	1.0	.68
r	1.0	0.0	0.0
f	1.0	0.0	0.0

Baseline, Dev

	P	R	F1
p	1.0	0.0	0.0
n	.46	1.0	.63
c	.57	1.0	.73
s	.49	1.0	.66
r	1.0	0.0	0.0
f	1.0	0.0	0.0

Perceptron, Bag of Words, Train

	P	R	F1
p	1.0	.97	.99
n	1.0	1.0	1.0
c	1.0	1.0	1.0
s	1.0	1.0	1.0
r	1.0	.97	.99
f	.99	1.0	1.0

Perceptron, Bag of Words, Dev

	P	R	F1
p	.67	.54	.59
n	.52	.59	.55
c	.61	.72	.66
s	.51	.63	.56
r	.43	.43	.43
f	.41	.34	.37

Naive Bayes, Bag of Words, Train

	P	R	F1
p	.94	.99	.96
n	.99	.97	.98
c	.98	.97	.97
s	.97	.97	.97
r	.94	.91	.92
f	.93	.98	.96

Naive Bayes, Bag of Words, Dev

	P	R	F1
p	.72	.55	.63
n	.6	.72	.65
c	.59	.72	.65
s	.55	.65	.59
r	.5	.48	.49
f	.4	.29	.33

Naive Bayes, Bag of Adjectives, Train

	P	R	F1
p	.83	.91	.87
n	.84	.85	.84
c	.83	.91	.87
s	.82	.85	.83
r	.89	.55	.68
f	.87	.68	.77

Naive Bayes, Bag of Adjective, Dev

	P	R	F1
p	.66	.66	.66
n	.58	.54	.56
c	.57	.72	.64
s	.48	.59	.53
r	.4	.35	.37
f	.35	.34	.35

As we can see, both the perceptron and Naive Bayes with bag of words features have serious problems of overfitting. They do too well on the training sets and poorly on the dev sets, although Naive Bayes outperforms the baseline in most cases. Switching to bag of adjectives features does not alleviate the issue much; it appears that under bag of adjective features, the training set is is no longer easy to separate.

5.2 Comparison - 10-fold Cross validation over 500 Examples

Here, we simply compare Naive Bayes and the baseline.

	Baseline	Bayes, Bag of Words
p	.512	.627
n	.467	.63
c	.583	.57
s	.515	.605
r	.76	.805
f	.635	.605

Naive Bayes does a little bit worse for some categories and a lot better for others. Again, it seems

that Naive Bayes is overfitting; to alleviate this issue, we will need to look to other feature spaces or use algorithms that can be regularized.

5.3 Specific Examples

We look at a few examples from the tale "The Goose-Girl" and posit about why the algorithm concluded what it did.

5.3.1 Romantic?

Title THE GOOSE-GIRL

Beat And it was very lucky for her that she did so, for when she had done the king ordered royal clothes to be put upon her, and gazed on her with wonder, she was so beautiful. Then he called his son and told him that he had only a false bride; for that she was merely a waiting-maid, while the true bride stood by. And the young king rejoiced when he saw her beauty, and heard how meek and patient she had been; and without saying anything to the false bride, the king ordered a great feast to be got ready for all his court.

Labels c, p, r

This section is not really romantic in the story, but we have the words "beautful" and "bride" all over. The classifier doesn't notice the issue of it being a "false bride" since we don't have bigram features.

5.3.2 Not Romantic?

Title THE GOOSE-GIRL

Beat And the young king rejoiced when he saw her beauty, and heard how meek and patient she had been; and without saying anything to the false bride, the king ordered a great feast to be got ready for all his court. The bridegroom sat at the top, with the false princess on one side, and the true one on the other; but nobody knew her again, for her beauty was quite dazzling to their eyes; and she did not seem at all like the little goose-girl, now that she had her brilliant dress on. \nWhen they had eaten and drank, and were very merry, the old king said he would tell them a tale.

Labels f, p

Given that the previous beat was declared romantic, it is interesting that this one is not.

5.3.3 C, N, S?

Title THE GOOSE-GIRL

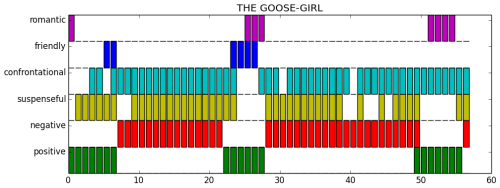
Beat When they had eaten and drank, and were very merry, the old king said he would tell them a tale. So he began, and told all the story of the princess, as if it was one that he had once heard; and he asked the true waiting-maid what she thought ought to be done to anyone who would behave thus. 'Nothing better,' said this false bride, 'than that she should be thrown into a cask stuck round with sharp nails, and that two white horses should be put to it, and should drag it from street to street till she was dead"

Labels c, n, s

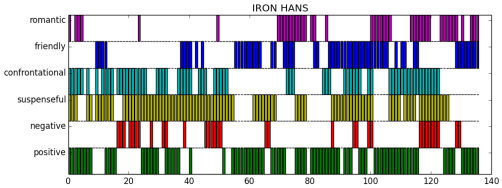
This passage is certainly negative and scary, and discusses conflict. However, when we look to other examples, we find that many, many beats are classified as c, n, s for no apparent reason. The reason under Naive Bayes, however, is that the priors suggest most moments in fairy tales are confrontational, negative, and scary–something we learn to be the case about Grimm's tales.

5.4 Visualization

Here is the visualization of the Goose-Girl using Naive Bayes trained with the bag of words features and the 500 labelled examples:



We can see that a large number of moments are considered c, n, s, ever though there are glimpses of positivity and romance. In contrast, the tale of Iron Hans appears much richer:



Outputs for all 61 of the fairy tales from Grimm are included in the Outputs directory of the submission file.

6 Future Work

To make full use of this system, much additional work will be to be done both in investigating feature functions and algorithms.

In feature functions, we need to look for features that do better for short pieces of text. Since there is a rich body of early research into Twitter mining, we may look there for examples.

In the algorithm space, SVMs with L2 regularization may be the best short term solution to the overfitting problem. With linear features, SVMs don't provide a very linguistically rich model; thus, training an SVM that is linguistically interesting will be totally reliant on the selection of the right features.

A theoretically interesting approach is to use Latent Dirichlet Allocation, with the justification that each beat is made up of dramatic properties in the same way that documents can be considered to be made out of topics.

7 References

7.1 Manning

Chris Manning, CS224N Lecture 18: Computational Semantics, 2013.

7.2 Green

Spence Green, CS224N Lecture 16: Topic Models, 2013.

7.3 Laws

Robin D. Laws, Hamlet's Hit Points. Published by Gameplaywright Press, 2010.