Storytelling with Adjustable Narrator Styles and Sentiments

Boyang Li, Mohini Thakkar, Yijie Wang, and Mark O. Riedl

School of Interactive Computing, Georgia Institute of Technology
{boyangli, mthakkar, yijiewang, riedl}@gatech.edu

Abstract. Most storytelling systems to date rely on manually coded knowledge, the cost of which usually restricts such systems to operate within a few domains where knowledge has been engineered. *Open Story Generation* systems are capable of learning knowledge necessary for telling stories in a given domain. In this paper, we describe a technique that generates and communicates stories in language with diverse styles and sentiments based on automatically learned narrative knowledge. Diversity in storytelling style may facilitate different communicative goals and focalization in narratives. Our approach learns from large-scale data sets such as the Google N-Gram Corpus and Project Gutenberg books in addition to crowdsourced stories to instill storytelling agents with linguistic and social behavioral knowledge. A user study shows our algorithm strongly agrees with human judgment on the interestingness, conciseness, and sentiments of the generated stories and outperforms existing algorithms.

1 Introduction

Narrative Intelligence (NI), or the ability to craft, tell, understand, and respond to stories, is considered a hallmark of human intelligence and an effective communication method. It follows that Narrative Intelligence is important for Artificial Intelligence that aims to simulate human intelligence or communicate effectively with humans. In this paper, we focus on computational NI systems that can generate and tell stories.

A significant challenge in building NI systems is the knowledge intensive nature of NI. To date, most computational systems purported to demonstrate NI are reliant on substantial amount of manually coded knowledge, whose availability is limited by the time and financial cost associated with knowledge engineering. Consequently, most systems are designed to operate in only a few micro-worlds where knowledge is available. For example, an automated story generator may be told about the characters and environment of Little Red Riding Hood; that system can tell a large variety of stories about the given set of characters and topic, but no stories about other characters or topics.

Open Story Generation systems (e.g. [5, 16]) have been proposed in order to tackle the challenge of generating and telling stories in any domain. Such systems can learn the needed knowledge for story generation and storytelling without *a priori* knowledge engineering about a particular domain. We previously described an open story generation system, Scheherazade [5], which uses crowdsourcing to construct a commonsense understanding about how to perform everyday activities such as going to a restaurant or going to a movie theater. Given a topic, the system learns what it needs to generate a story about the topic. However, the system does not reason about how to *tell* a story, or how to translate a sequence of abstract events into natural language.

A story may be told to achieve communicative goals, such as to entertain, to motivate, or to simply report facts. Different goals may require narrators to adopt different storytelling styles. Additionally, the narrative technique of focalization involves describing the same events from different characters' perspectives, possibly with opposing sentiments (cf. [2,18]). As a first step, we tackle the problem of creating different storytelling styles for Open Story Generation. Style parameters are learned from large data sets including the Google N-Gram corpus [8] and books from Project Gutenberg (`www.gutenberg.org`). We offer methods to tune the storytelling with different levels of details, fictional language, and sentiments. Our user study indicates our algorithm strongly agrees with human readers' intuition of linguistic styles and sentiments, and outperforms existing methods.

1.1 Background and Related Work

Story generation and interactive narrative have a long history. See Gervás [3] and Riedl & Bulitko [12] for overviews. Several Open Story Generation systems have been proposed before. The SayAnything system [16] generates stories from snippets of natural language mined from weblogs. McIntyre & Lapata [7] learn temporally ordered schema from fairy tales, merge schema into plot graphs, and use a genetic algorithm to maximize the coherence of generated stories. Crowdsourcing has been proposed as a means for overcoming the knowledge bottleneck. Sina *et al.* [14] use case-based reasoning to modify crowdsourced semi-structured stories to create alibi for virtual suspects in training. None of these approaches explicitly model discourse or generate different narration styles.

The work in this paper builds off our previous work on the Scheherazade system [4, 5], which learns the structure of events in a given situation from crowdsourced exemplar stories describing that situation. As opposed to other story generation systems, Scheherazade is a just-in-time learner; if the system does not know the structure of a situation when it is called for, it attempts to learn what it needs to know from a crowd of people on the Web. This results in a script-like knowledge structure, called a *plot graph*. The graph contains events that can be expected to occur, temporal ordering relations between events, and mutual exclusions between events that create branching alternatives.

The learning of the plot graph proceeds in four steps [4, 5]. After exemplar stories about a social situation are crowdsourced from Amazon Mechanical Turk (AMT), the learning starts by creating clusters of sentences of similar semantic meaning from different exemplar stories. Each cluster becomes an event in the plot graph. In order to reduce the difficulty in natural language processing, crowd workers from AMT have been asked to use simple language, i.e., using one sentence with a single verb to describe one event, avoiding pronouns, etc. The second and third steps identify the temporal precedences and mutual exclusions between events. The final step identifies optional events. Story generation in SCHEHERAZADE is the process of selecting a linear sequence of events that do not violate any temporal or mutual exclusion relations in the script [5]. However, telling the generated story in natural language with different storytelling styles has not been previously realized.

Focalization in narration refers to telling stories from different viewpoints (e.g. of an omniscient entity or any story character; cf. [2]), potentially requiring multiple narration styles. Most computational implementations focus on plot events [11, 18] instead of linguistic variations. Curveship [10] generates focalized text based on manually coded knowledge. Our work directly addresses the problem of diverse language use by implied or explicit narrators.

Automatic generation of distinct linguistic pragmatics for narration has also been studied. The PERSONAGE system [6] maps the Big Five psychological model to a large number of linguistic parameters. Rishes *et al.* [13] used PERSONAGE to create different tellings of stories generated from a semantic representation consisting of events and character intentions. The generated linguistic styles differ mostly in aspects independent of content, such as in the use of swear words, exclamation marks and shuttering. Instead of generating from symbolic representations with precise semantic meaning, we select from existing sentences that are similar but not strictly synonymous to describe an event (i.e. sentences may differ in content). We consider parameters directly related to word choices: degree of details, fictionality, and sentiments.

2 Storytelling with Different Styles and Sentiments

This section describes the process of telling stories in natural language with a variety of personal styles. The architecture for story generation and communication is shown in Figure 1. Plot graph learning is typically an offline process that incrementally constructs a knowledge base of models of social situations, from which stories can be generated [4, 5]. A story is generated as one possible total-ordered sequence of events that respect all constraints in the plot graph. The discourse planning stage selects some interesting events from the complete sequence to be told, which is beyond the scope of this paper and not used in the evaluation. This paper focuses on the last stage of the architecture: describing the selected events with personal styles and affects, which we explain below.

Recall that each event in the learned plot graph is a cluster of natural language descriptions of similar meaning. Given a generated story, or a complete, linear sequence of events, natural language text is generated by selecting the sentence from each cluster that best matches the intended narration style. We describe two criteria for selecting individual sentences: (1) the interestingness of the text and (2) the sentiment of the text. We aim to create a diverse set of storytelling styles that may be suitable for different occasions. For example,

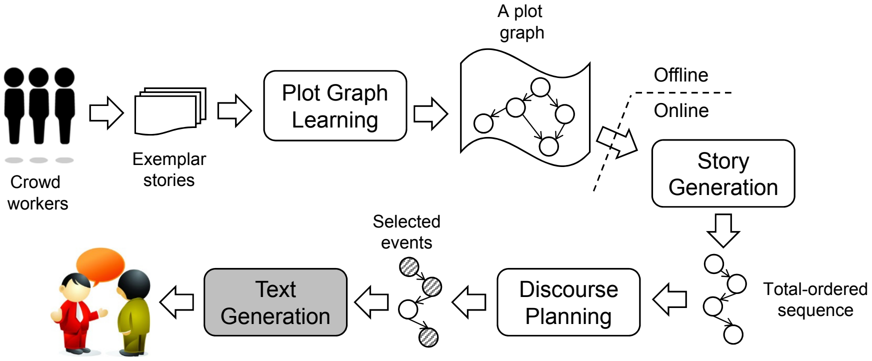

Fig. 1. The system pipeline.

some narrators or story characters may speak very succinctly, whereas others can recall vivid details. A positive tone may be used if the narrator wants to cheer up the audience; a negative tone may be suitable for horror stories and so on. We also present a Viterbi-style algorithm that considers preferences on individual sentences and inter-sentence connections to produce coherent textual realizations.

We performed a second round of crowdsourcing to obtain a variety of event descriptions that can reflect different narration styles and sentiments. The originally crowdsourced exemplar stories were written in simple sentences that help to simplify natural language processing [4]. Thus, only simple event descriptions were available for selection. The second round of crowdsourcing asked AMT workers to provide "interesting" event descriptions. For $1, workers wrote detailed descriptions for each event in a given story; each description may contain more than one sentences. We allowed workers to interpret "interesting" however they wanted, though we suggested that they describe characters' intentions, facial expressions, and actions. Each worker saw a complete sequence of events to make sure they understand the story context. We accepted all stories that describe the events we provide, and did not perform a manual check of interestingness.

2.1 Textual Interestingness

We investigate two aspects of language that affect the interestingness of stories. The first is the amount of details provided, and the second is the degree that the story language resembles the language used in fictions. We model the amount of details with the probability of a sentence in English, since Information Theory suggests a less likely sentence contains more information (i.e. more details). We compute the probability of an English word as its frequency in the Google N-Gram corpus. Due to the large size of the corpus, these frequencies approximate word probabilities in general English. We compute the probability of a sentence using the bag-of-word model, where the probability of sentence S containing words $w_1, w_2, \ldots, w_k$, each appearing $x_1, x_2, \ldots, x_k$ times is

$$P(S) = \frac{\left(\sum_{i=1}^{k} x_i\right)!}{\prod_{i=1}^{k}(x_i!)} \prod_{i=1}^{k} P(w_i)^{x_i} \tag{1}$$

where $P(w_i)$ is the probability of word w_i. For our purpose, the average frequency over the 10-year period of 1991 to 2000 in the "English 2012" corpus is used. Stop words are removed before computation.

We further consider the style of language as how much it resembles fictional novels. The language used in fictions has distinctive word choices as fictions tend to accurately describe actions (e.g. "snatch" instead of "take") and emotions, and make less use of formal words (e.g. "facility", "presentation"). If a word appears more frequently in fiction books than in all books, we can presume that its use creates a sense that the story is being told in a literary manner. Therefore, the *fictionality* of a word w is the ratio

$$f_w = P_{\text{fic}}(w)/P(w) \tag{2}$$

where $P(w)$ is the probability of a word computed previously and $P_{\text{fic}}(w)$ is the probability of a word appearing in the "English Fiction 2012" corpus from the Google N-Gram corpus. The fictionality of a sentence is aggregated from fictionality values of individual words as an exponentiated average:

$$\text{fic}(S) = \frac{\sum_{w \in W} \exp(\alpha f_w)}{\text{card}(W)} \tag{3}$$

where W is the multiset of words in sentence S, and $\text{card}(W)$ is its cardinality. α is a scaling parameter. The exponential function puts more weights on words with higher fictionality, so that a few highly fictional words are not canceled off by many words with low fictionality.

Table 1 shows some example sentences. We observe that the most probable sentence (MostProb) usually provides a good summary for the event. The most fictional (MostFic) sentence usually contains more subjective emotions and character intentions, whereas the least probable (LeastProb) sentence is usually longer and contains more objective details. We balance and combine the MostFic and the LeastProb criteria by using the harmonic mean in order to create the sentence with most interesting details (MID). Let us denote the ranks of each sentence under the LeastProb and the MostFic criteria as r_{LP} and r_{MF} respectively. For example, the least probable sentence has $r_{\text{LP}} = 1$, and the second most fictional has $r_{\text{MF}} = 2$. The harmonic mean rank r_{MID} is computed as $2\, r_{\text{LP}}\, r_{\text{MF}}/(r_{\text{LP}} + r_{\text{MF}})$. The sentence with the lowest r_{MID} is picked as the one with the most interesting details.

2.2 Textual Sentiments

Stories may be told with positive or negative sentiment. To detect sentiments of sentences in each event cluster, we construct a sentiment dictionary called

Example event 1: Sally puts money in bag

- MostProb: Sally put $1,000,000 in a bag.
- LeastProb: Sally put the money in the bag, and collected the money from the 2 tellers next to her.
- MostFic: Sally quickly and nervously stuffed the money into the bag.
- MID: Sally quickly and nervously stuffed the money into the bag.
- Positive: Sally continued to cooperate, putting the money into the bag as ordered.
- Negative: Sally's hands were trembling as she put the money in the bag.

Example event 2: John drives away

- MostProb: John drove away.
- LeastProb: John pulled out of the parking lot and accelerated, thinking over which route would make it easier to evade any police cars that might come along.
- MostFic: John sped away, hoping to get distance between him and the cops.
- MID: John sped away, hoping to get distance between him and the cops.
- Positive: As the stoplight turned green and the daily traffic began to move, John drove away.
- Negative: John slammed the truck door and, with tires screaming, he pulled out of the parking space and drove away.

Table 1. Example Sentences Selected with the Probability, Fictionality, and Sentiment Criteria

Smooth SentiWordNet (SSWN). SSWN builds off SentiWordNet [1], which tags each synset (word sense) in WordNet [9] with three values: positivity, negativity, and objectiveness, the three summing to 1. SentiWordNet was produced by propagating known sentiments of a few seed words along connections between words in WordNet to provide good coverage, but this automatic approach can produce many erroneous values, resulting in unreliable sentiment judgments. Smooth SentiWordNet uses an unsupervised, corpus-based technique to correct errors found in the original library and expand its coverage beyond words appearing in WordNet. The intuition behind SSWN is that words that are nearby should share similar sentiments, and words closer should have a stronger influence than words farther away. We take sentiment values from SWN and "smooth" the values based on word location using Gaussian kernel functions, in order to alleviate errors and further expand the coverage.

We perform smoothing with a corpus of 9108 English books from Project Gutenberg that are labeled as fiction. These books are tagged with parts of speech (POS) with the Stanford POS Tagger [17]. Each pair of word and POS is considered a unique word. For every word we want to compute sentiment value for, we consider a neighborhood of 100 words, 50 to its left and 50 to its right. The target word is at position 0 and denoted as w_0. The words to its immediate left and right are at position -1 and 1, and so forth. The positions of these words are included in the index set N. For word w_i at position $i \in N$, its influence at position j is modeled with a Gaussian kernel function g_i: $g_i(j) = \exp\left(-(i-j)^2/d\right)$, where parameter d determines how fast the function diminishes with distance, and is empirically set to 32. Only nouns, verbs, adjectives and adverbs in complete sentences can influence the target word.

(...the first 7 sentences omitted)
When it was his turn, John, wearing his Obama mask, approached the counter.
Sally saw Obama standing in front of her and she felt her whole body tense up as her worst nightmare seemed to be coming true.
Once Sally began to run, John pulled out the gun and directed it at the bank guard.
John wore a stern stare as he pointed the gun at Sally.
Sally saw the gun and instantly screamed before she could stop herself.
John told her she had one minute to get the money and shook the gun at her.
John gave Sally a bag to put the banks money in.
John struggled to stuff the money in his satchel.
Sally was quietly sobbing as John grabbed the bag full of money.
John strode quickly from the bank and got into his car tossing the money bag on the seat beside him.
John pulled out of the parking lot and accelerated, thinking over which route would make it easier to evade any police cars that might come along.

Table 2. An example partial story with most interesting details. The first 7 sentences in the story are omitted for space reasons.

In the each neighborhood that the target word w_0 appears, its sentiment s_{w_0} is computed as a weighted average of all kernel functions at position 0:

$$s_{w_0} = \frac{\sum_{i \in N} s^{\mathrm{swn}}_{w_i} g_i(0)}{\sum_{i \in N} g_i(0)} \tag{4}$$

where $s^{swn}_{w_i}$ is the sentiment value from SentiWordNet, i.e. the difference between the positive and negative polarity. The SentiWordNet value for the target word w_0 has no influence on itself, i.e. $0 \notin N$. As a word can appear multiple times in different neighborhoods, the final sentiment value for w_0 is the average over all neighborhoods it appears in. We aggregate sentiments of individual words in sentence S, again using the exponential average:

$$\mathrm{sentiment}(S) = \frac{\sum_{w \in V} \mathrm{sign}\,(s_w)\ \exp\,(\beta |s_w|)}{\mathrm{card}(V)} \tag{5}$$

where $\mathrm{card}(V)$ is the cardinality of the multiset V, containing only nouns, verbs, adjectives or adverbs in sentence S. β is a scaling parameter. The exponential function ensures that words expressing strong sentiments are weighted more heavily than words with weak sentiments.

We selected a subset of English words that are of interest to our task. The exemplar stories in two previously crowdsourced social situations—dating at the movie theater and bank robbery—contain 1001 unique nouns, verbs, adverbs and adjectives. We selected highly influential adjectives and adverbs from their direct neighbors, producing a total of 7559 words. We normalize the raw values produced by smoothing, so that 1 percentile and 99 percentile of the values fall in the range of $[-1, 1]$, to account for outliers.[1] Table 1 shows some of the most

[1] The list of books can be downloaded at `http://www.cc.gatech.edu/~bli46/SBG/list.txt`. The resulted dictionary is at: `http://www.cc.gatech.edu/~bli46/SBG/dic.txt`.

positive and most negative sentences. We find the results to reflect the valences of individual words. Although this approach works most of the time, there are cases such as sarcasm where the sentiment of a sentence could be the opposite of that of individual words. SSWN is evaluated in Section 3.

2.3 Connecting Sentences

For each event, we can find individual sentences ranked highest for any criterion or combinations of criteria using the harmonic mean. However, this selection does not consider the coherence between sentences and may results in incoherent texts due to two major problems: (1) previously mentioned objects can suddenly disappear and previously unmentioned objects can appear, and (2) a sentence can repeat actions in the previous sentence. To address this problem, we propose a Viterbi-style algorithm, which considers both selection criteria for individual sentences and the connection between sentences.

In a hidden Markov model, the Viterbi algorithm generates a sequence of hidden variables that best explains a sequence of observed random variables. The algorithm relies on two things specified in a hidden Markov model: One, the probabilities of a hidden variable generating any observation. That is, the observation indicates preference over the values of a hidden variable. Two, the probabilities of any hidden variable transiting to the hidden variable in the next time slice. That is, we have preferences over pairs of values for adjacent variables. The Viterbi algorithm generates the best hidden variable sequence by considering the two set of preferences at the same time.

Our problem is similar as we want to find the highest scored sentence sequence based on preferences over sentences in each event cluster, as well as preferences on how adjacent sentences connect. For this paper, we do not consider connection between non-adjacent sentences. Specifically, we score the connection between any two sentences s_i, s_j as $\text{score}(s_i, s_j) = \log\left((sn(i,j)+1)/(sv(i,j)+1)\right)$, where $sn(i,j)$ is the number of nouns shared by the two sentences, and $sv(i,j)$ is the number of verbs shared by the two sentences. Similarly, we score individual sentences as the reciprocal of their ranks according to any selection criterion $c : \text{score}(s_i) = 1/\text{rank}_c(s_i)$.

Our algorithm is shown as Algorithm 1. The BestSeqEndingIn function is recursive, because in order to find the best sequence ending in a given sentence s_i^j from the j^{th} event cluster c_j, we need to consider the scores of best sequences ending in every sentence from the previous cluster c_{j-1}, in addition to the connection between every sentence from cluster c_{j-1} and s_i^j. Due to the Markov property, we do not need to consider clusters $c_1, \ldots, c_{j-2}$. We can then iterate over every sentence from cluster c_j to find the best sequence ending in cluster c_j. A dynamic programming approach can be used to store every sequence ending in every sentence from every cluster and their scores. For a sequence of n clusters and m sentences in each cluster, the time and space complexity are $O(m^2 n)$ and $O(mn)$. An example partial story is shown in Table 2.

Algorithm 1 Generation of Story Text

```
function GenerateText(event sequence $\langle c_1, c_2, \ldots, c_n \rangle$)
    for each sentence $s_k \in \{s_1, s_2, \ldots, s_m\}$ in event cluster $c_n$ do
        $(seq_k, \text{score}(seq_k)) \leftarrow$ BestSeqEndingIn$(s_k, c_n)$
    end for
    return the highest scored sequence from $seq_1, seq_2, \ldots, seq_m$
end function

function BestSeqEndingIn$(s_i, c_j)$
    for each sentence $s_p \in \{s_1, s_2, \ldots, s_m\}$ in event cluster $c_{j-1}$ do
        $(seq_p, \text{score}(seq_p)) \leftarrow$ BestSeqEndingIn$(s_p, c_{j-1})$        ▷ stored previously
        $new\_seq_p \leftarrow seq_p + s_i$
        $\text{score}(new\_seq_q) \leftarrow \text{score}(seq_q) + \text{score}(s_q, s_i) + \text{score}(s_i)$
    end for
    $best\_seq \leftarrow$ the highest scored sequence from $new\_seq_1, \ldots, new\_seq_m$
    return $(best\_seq, \text{score}(best\_seq))$
end function
```

	Movie Date	Bank Robbery
# Stories	20	10
# Sentences	470	210
# Words per sentence	14.53	13.7
# Verbs per sentence	2.36	2.6

Table 3. Statistics of crowdsourced interesting stories

3 Evaluation

We performed a user study to test if the results of our algorithm agree with human intuition. We investigated two social situations: dating at a movie theater and bank robbery. In addition to the originally crowdsourced exemplar stories, we crowdsourced interesting stories using procedures described in Section 2. Some statistics of these stories are shown in Table 3. Some of these sentences can be seen in prior examples showing least probable and most fictional sentences for particular clusters (the most probably sentence typically comes from the original, simplified language exemplars).

With the newly crowdsourced sentences added, for each situation we generate two groups of stories with the Viterbi-style algorithm with different sentence selection criteria. We do not perform discourse planning to avoid confounding factors. The first group includes stories generated from the most interesting details (MID) criterion, the most probable (MostProb) criterion, and a story where we use the MID criterion but penalize long sentences. After reading the stories, participants are asked to select the most interesting story, the most detailed story and the most concise story. Our hypothesis is that human readers will select the MID story as containing the most details and the most interesting, and the MostProb story as the most concise. We set α to 12. The second group of

	Participant Agreement %	
Test	Movie Date	Bank Robbery
Most Concise Story	$90.38^{\S}$	75.00^{*}
Most Detailed Story	$97.92^{\S}$	$100.00^{\S}$
Most Interesting Story	$88.46^{\S}$	$80.77^{\S}$
Positive/Negative Stories	$86.54^{\S}$	$92.31^{\S}$

Table 4. Participant agreement with our algorithm. § denotes $p < 0.0001$. * denotes $p < 0.0005$.

stories include a story containing the sentences with the most positive sentiment from each event, and a story containing the sentences with the most negative sentiment. We set β to 16 and 2 for the movie data and bank robbery situation respectively. After reading the second group, participants are asked to select a positive and a negative story. We hypothesize human readers will agree with the algorithm's sentiment judgments.

A total of 52 undergraduate, masters, and doctoral students participated in our study. Table 4 shows the percentage of human participants that agree with the algorithm. All results are predominantly positive and consistent with our hypothesis, strongly indicating our algorithm can capture the human intuition of interestingness, conciseness, and sentiments. We use one-tailed hypothesis testing based on the multinomial/binomial distribution and find the results to be extremely statistically significantly above a random baseline.

However, it is arguably easier to detect the sentiment of an entire story than to detect the sentiment of individual sentences, because in a story, a few sentences labeled with wrong sentiments mixed with many correctly labeled sentences can be overlooked by human readers. To further evaluate our sentiment detection algorithm, we perform a sentence-level comparison. We first take out the top 3 most positive sentence and top 3 most negative sentences from 45 event clusters in both situations. One positive and one negative sentences are randomly selected from the top 3, and shown to participants, who labeled one sentence as positive and the other as negative. In total, 52 participants performed 4678 evaluations of 265 unique pairs of sentences. The results are shown in Table 5 . Overall, 70.76% of participants' decisions agree with our algorithm. The majority opinion on each pairs of sentences agree with our algorithm for 80.75% of the time.

We further compare our algorithm with SentiWordNet. We replaced word sentiments in Equation 5 with values directly taken from SentiWordNet, and label a sentence as positive if its sentiment is higher than the median sentence in a cluster, and negative if lower. Results are matched against the participants' decisions. We tuned β to maximize performance. We also compare SSNW with the technique by Socher *et al.* [15] from Stanford University, which directly labels a sentence as positive, negative or neutral. The results are summarized in Table 5. SSWN outperform SWN by a margin of 11.16% to 16.22%, and outperform Socher *et al.* by 34.85% to 41.5%, although Socher *et al.*'s algorithm targets

	Participant Agreement %		
Test	Smooth SWN	SentiWordNet	Socher et al.
Sentence Sentiments	70.76	59.60§	35.91§
Sentence Sentiments by Majority Vote	80.75	64.53§	39.25§

Table 5. Comparing word sentiment values from SentiWordNet and the values computed by our smoothing technique. § denotes $p < 0.0001$.

movie reviews and has not been tuned on our data set. A Chi-Square test shows the difference between conditions are extremely statistically significant.

4 Discussion and Conclusions

Open Story Generation systems can learn necessary knowledge to generate stories about unknown situations. However, these systems have not considered how to tell the generated story in natural language with different styles. Such a capability is useful for achieving different communicative goals and for projecting a story to perspectives of story characters. For example, a story with mostly objective details is suitable for conveying information, whereas interesting stories tend to describe characters' subjective feelings. A positive tone may be used to cheer up the audience, or to describe things from a cheerful character's perspective. As a first step toward solving these problems, we discuss Open Storytelling with different styles, such as attention to detail, fictionality of language, and sentiments. Our technique employ the same knowledge structure learned by Open Story Generation systems and large data sets including the Google N-Gram Corpus and Project Gutenberg. We develop a method for selecting interesting event descriptions and build a sentiment dictionary called Smooth SentiWordNet by smoothing out errors in sentiment values obtained from SentiWordNet. Our user study with 52 participants reveals that corpus-based techniques can achieve recognizably different natural language styles for storytelling. Future work will investigate newer fiction corpora, such as weblogs labeled as stories, than Project Gutenberg, which may not fully reflect the language use of this day.

Our storytelling techniques help to overcome the authoring bottleneck for storytelling systems by learning from data sets consisting of crowdsourced exemplar stories, the Google N-Gram Corpus, and books from Project Gutenberg, Building off existing work [4, 5], the effort presented in this paper moves the state of the art towards the vision of computational systems capable of telling an unlimited number of stories about an unlimited number of social situations with minimum human intervention.

5 Acknowledgments

We gratefully acknowledge DARPA for supporting this research under Grant D11AP00270, and Stephen Lee-Urban and Rania Hodhod for valuable inputs.

References

1. Baccianella, S., Esuli, A., Sebastani, F.: SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In: The 7th Conference on International Language Resources and Evaluation (2010)
2. Bae, B.C., Cheong, Y.G., Young, R.M.: Automated story generation with multiple internal focalization. In: 2011 IEEE Conference on Computational Intelligence and Games. pp. 211–218 (2011)
3. Gervás, P.: Computational approaches to storytelling and creativity. AI Magazine 30, 49–62 (2009)
4. Li, B., Lee-Urban, S., Appling, D., Riedl, M.: Crowdsourcing narrative intelligence. Advances in Cognitive Systems 2 (2012)
5. Li, B., Lee-Urban, S., Johnston, G., Riedl, M.: Story generation with crowdsourced plot graphs. In: The 27th AAAI Conference on Artificial Intelligence (2013)
6. Mairesse, F., Walker, M.: Towards personality-based user adaptation: Psychologically informed stylistic language generation. User Modeling and User-Adapted Interaction 20, 227–278 (2010)
7. McIntyre, N., Lapata, M.: Plot induction and evolutionary search for story generation. In: The 48th Annual Meeting of the Association for Computational Linguistics. pp. 1562–1572 (2010)
8. Michel, J.B., Shen, Y., Aiden, A., Veres, A., Gray, M., Brockman, W., The Google Books Team, Pickett, J., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M., Aiden, E.: Quantitative analysis of culture using millions of digitized books. Science 331, 176–182 (2011)
9. Miller, G.: WordNet: A lexical database for English. Communications of the ACM 38, 39–41 (1995)
10. Montfort., N.: Generating narrative variation in interactive fiction. Ph.D. thesis, University of Pennsylvania (2007)
11. Porteous, J., Cavazza, M., Charles, F.: Narrative generation through characters point of view. In: The SIGCHI Conference on Human Factors in Computing Systems (2010)
12. Riedl, M.O., Bulitko, V.: Interactive narrative: An intelligent systems approach. AI Magazine 34, 67–77 (2013)
13. Rishes, E., Lukin, S., Elson, D., Walker, M.: Generating different story tellings from semantic representations of narrative. In: The 6th International Conference on Interactive Storytelling (2013)
14. Sina, S., Rosenfeld, A., Kraus, S.: Generating content for scenario-based serious-games using crowdsourcing. In: The 28th AAAI Conference on Artificial Intelligence (2014)
15. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: The Conference on Empirical Methods in Natural Language Processing (2013)
16. Swanson, R., Gordon: Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling. ACM Transactions on Interactive Intelligent Systems 2, 1–35 (2012)
17. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: The NAACL-HLT Conference (2003)
18. Zhu, J., Ontañón, S., Lewter, B.: Representing game characters' inner worlds through narrative perspectives. In: The 6th International Conference on Foundations of Digital Games. pp. 204–210 (2011)