Learning to Tell Tales: A Data-driven Approach to Story Generation

Neil McIntyre and **Mirella Lapata**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, UK
n.d.mcintyre@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Computational story telling has sparked great interest in artificial intelligence, partly because of its relevance to educational and gaming applications. Traditionally, story generators rely on a large repository of background knowledge containing information about the story plot and its characters. This information is detailed and usually hand crafted. In this paper we propose a data-driven approach for generating short children's stories that does not require extensive manual involvement. We create an end-to-end system that realizes the various components of the generation pipeline stochastically. Our system follows a generate-and-and-rank approach where the space of multiple candidate stories is pruned by considering whether they are plausible, interesting, and coherent.

1 Introduction

Recent years have witnessed increased interest in the use of interactive language technology in educational and entertainment applications. Computational story telling could play a key role in these applications by effectively engaging learners and assisting them in creating a story. It could also allow teachers to generate stories on demand that suit their classes' needs. And enhance the entertainment value of role-playing games[1]. The majority of these games come with a set of pre-specified plots that the players must act out. Ideally, the plot should adapt dynamically in response to the players' actions.

Computational story telling has a longstanding tradition in the field of artificial intelligence. Early work has been largely inspired by Propp's (1968) typology of narrative structure. Propp identified in Russian fairy tales a small number of recurring units (e.g., the hero is defeated, the villain causes harm) and rules that could be used to describe their relation (e.g., the hero is pursued and the rescued). Story grammars (Thorndyke, 1977) were initially used to capture Propp's high-level plot elements and character interactions. A large body of more recent work views story generation as a form of agent-based planning (Theune et al., 2003; Fass, 2002; Oinonen et al., 2006). The agents act as characters with a list of goals. They form plans of action and try to fulfill them. Interesting stories emerge as agents' plans interact and cause failures and possible replanning.

Perhaps the biggest challenge faced by computational story generators is the amount of world knowledge required to create compelling stories. A hypothetical system must have information about the characters involved, how they interact, what their goals are, and how they influence their environment. Furthermore, all this information must be complete and error-free if it is to be used as input to a planning algorithm. Traditionally, this knowledge is created by hand, and must be recreated for different domains. Even the simple task of adding a new character requires a whole new set of action descriptions and goals.

A second challenge concerns the generation task itself and the creation of stories characterized by high-quality prose. Most story generation systems focus on generating plot outlines, without considering the actual linguistic structures found in the stories they are trying to mimic (but see Callaway and Lester 2002 for a notable exception). In fact, there seems to be little common ground between story generation and natural language generation (NLG), despite extensive research in both fields. The NLG process (Reiter and Dale, 2000) is often viewed as a pipeline consisting of content planning (selecting and structuring the story's content), microplanning (sentence ag-

[1]A role-playing game (RPG) is a game in which the participants assume the roles of fictional characters and act out an adventure.

gregation, generation of referring expressions, lexical choice), and surface realization (agreement, verb-subject ordering). However, story generation systems typically operate in two phases: (a) creating a plot for the story and (b) transforming it into text (often by means of template-based NLG).

In this paper we address both challenges facing computational story telling. We propose a data-driven approach to story generation that does not require extensive manual involvement. Our goal is to create stories automatically by leveraging knowledge inherent in corpora. Stories within the same genre (e.g., fairy tales, parables) typically have similar structure, characters, events, and vocabularies. It is precisely this type of information we wish to extract and quantify. Of course, building a database of characters and their actions is merely the first step towards creating an automatic story generator. The latter must be able to select which information to include in the story, in what order to present it, how to convert it into English.

Recent work in natural language generation has seen the development of learning methods for realizing each of these tasks automatically without much hand coding. For example, Duboue and McKeown (2002) and Barzilay and Lapata (2005) propose to learn a content planner from a parallel corpus. Mellish et al. (1998) advocate stochastic search methods for document structuring. Stent et al. (2004) learn how to combine the syntactic structure of elementary speech acts into one or more sentences from a corpus of good and bad examples. And Knight and Hatzivassiloglou (1995) use a language model for selecting a fluent sentence among the vast number of surface realizations corresponding to a single semantic representation. Although successful on their own, these methods have not been yet integrated together into an end-to-end probabilistic system. Our work attempts to do this for the story generation task, while bridging the gap between story generators and NLG systems.

Our generator operates over predicate-argument and predicate-predicate co-occurrence statistics gathered from corpora. These are used to produce a large set of candidate stories which are subsequently ranked based on their interestingness and coherence. The top-ranked candidate is selected for presentation and verbalized using a language model interfaced with RealPro (Lavoie and Rambow, 1997), a text generation engine. This generate-and-rank architecture circumvents the complexity of traditional generation systems, where numerous, often conflicting constraints, have to be encoded during development in order to produce a single high-quality output.

This is a fat hen.
The hen has a nest in the box.
She has eggs in the nest.
A cat sees the nest, and can get the eggs.

The sun will soon set.
The cows are on their way to the barn.
One old cow has a bell on her neck.
She sees the dog, but she will not run.
The dog is kind to the cows.

Figure 1: Children's stories from McGuffey's Eclectic Primer Reader; it contains primary reading matter to be used in the first year of school work.

As a proof of concept we initially focus on children's stories (see Figure 1 for an example). These stories exhibit several recurrent patterns and are thus amenable to a data-driven approach. Although they have limited vocabulary and non-elaborate syntax, they nevertheless present challenges at almost all stages of the generation process. Also from a practical point of view, children's stories have great potential for educational applications (Robertson and Good, 2003). For instance, the system we describe could serve as an assistant to a person who wants suggestions as to what could happen next in a story. In the remainder of this paper, we first describe the components of our story generator (Section 2) and explain how these are interfaced with our story ranker (Section 3). Next, we present the resources and evaluation methodology used in our experiments (Section 4) and discuss our results (Section 5).

2 The Story Generator

As common in previous work (e.g., Shim and Kim 2002), we assume that our generator operates in an interactive context. Specifically, the user supplies the topic of the story and its desired length. By topic we mean the entities (or characters) around which the story will revolve. These can be a list of nouns such as *dog* and *duck* or a sentence, such as *the dog chases the duck*. The generator next constructs several possible stories involving these entities by consulting a knowledge base containing information about dogs and ducks (e.g., dogs bark, ducks swim) and their interactions (e.g., dogs chase ducks, ducks love dogs). We conceptualize

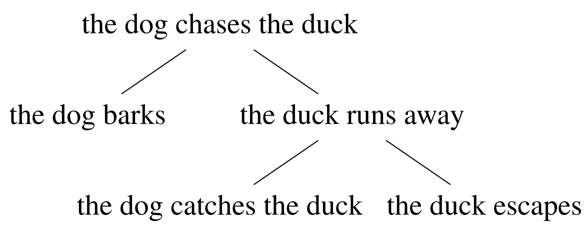

Figure 2: Example of a simplified story tree.

dog:SUBJ:bark	whistle:OBJ:dog
dog:SUBJ:bite	treat:OBJ:dog
dog:SUBJ:see	give:OBJ:dog
dog:SUBJ:like	have: OBJ:dog
hungry:ADJ:dog	lovely:ADJ:dog

Table 1: Relations for the noun *dog* with high *MI* scores (SUBJ is a shorthand for subject-of, OBJ for object-of and ADJ for adjective-of).

the story generation process as a tree (see Figure 2) whose levels represent different story lengths. For example, a tree of depth 3 will only generate stories with three sentences. The tree encodes many stories efficiently, the nodes correspond to different sentences and there is no sibling order (the tree in Figure 2 can generate three stories). Each sentence in the tree has a score. Story generation amounts to traversing the tree and selecting the nodes with the highest score

Specifically, our story generator applies two distinct search procedures. Although we are ultimately searching for the best overall story at the document level, we must also find the most suitable sentences that can be generated from the knowledge base (see Figure 4). The space of possible stories can increase dramatically depending on the size of the knowledge base so that an exhaustive tree search becomes computationally prohibitive. Fortunately, we can use beam search to prune low-scoring sentences and the stories they generate. For example, we may prefer sentences describing actions that are common for their characters. We also apply two additional criteria in selecting good stories, namely whether they are coherent and interesting. At each depth in the tree we maintain the N-best stories. Once we reach the required length, the highest scoring story is presented to the user. In the following we describe the components of our system in more detail.

2.1 Content Planning

As mentioned earlier our generator has access to a knowledge base recording entities and their interactions. These are essentially predicate argument structures extracted from a corpus. In our experiments this knowledge base was created using the RASP relational parser (Briscoe and Carroll, 2002). We collected all verb-subject, verb-object, verb-adverb, and noun-adjective relations from the parser's output and scored them with the mutual information-based metric proposed in Lin (1998):

$$MI = \ln \left(\frac{\| w, r, w' \| \times \| *, r, * \|}{\| w, r, * \| \times \| *, r, w' \|} \right) \qquad (1)$$

where w and w' are two words with relation type r. $*$ denotes all words in that particular relation and $\| w, r, w' \|$ represents the number of times w, r, w' occurred in the corpus. These *MI* scores are used to inform the generation system about likely entity relationships at the sentence level. Table 1 shows high scoring relations for the noun *dog* extracted from the corpus used in our experiments (see Section 4 for details).

Note that *MI* weighs binary relations which in some cases may be likely on their own without making sense in a ternary relation. For instance, although both dog:SUBJ:run and president:OBJ:run are probable we may not want to create the sentence *"The dog runs for president"*. Ditransitive verbs pose a similar problem, where two incongruent objects may appear together (the sentence *John gives an apple to the highway* is semantically odd, whereas *John gives an apple to the teacher* would be fine). To help reduce these problems, we need to estimate the likelihood of ternary relations. We therefore calculate the conditional probability:

$$p(a_1, a_2 \mid s, v) = \frac{\| s, v, a_1, a_2 \|}{\| s, v, *, * \|} \qquad (2)$$

where s is the subject of verb v, a_1 is the first argument of v and $a2$ is the second argument of v and $v, s, a_1 \neq \varepsilon$. When a verb takes two arguments, we first consult (2), to see if the combination is likely before backing off to (1).

The knowledge base described above can only inform the generation system about relationships on the sentence level. However, a story created simply by concatenating sentences in isolation will often be incoherent. Investigations into the interpretation of narrative discourse (Asher and Lascarides, 2003) have shown that lexical information plays an important role in determining

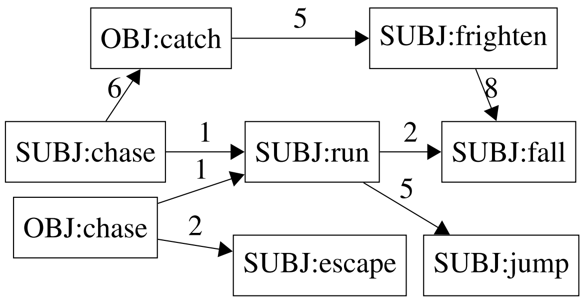

Figure 3: Graph encoding (partially ordered) chains of events

the discourse relations between propositions. Although we don't have an explicit model of rhetorical relations and their effects on sentence ordering, we capture the lexical inter-dependencies between sentences by focusing on events (verbs) and their precedence relationships in the corpus. For every entity in our training corpus we extract event chains similar to those proposed by Chambers and Jurafsky (2008). Specifically, we identify the events every entity relates to and record their (partial) order. We assume that verbs sharing the same arguments are more likely to be semantically related than verbs with no arguments in common. For example, if we know that someone steals and then runs, we may expect the next action to be that they hide or that they are caught.

In order to track entities and their associated events throughout a text, we first resolve entity mentions using OpenNLP[2]. The list of events performed by co-referring entities and their grammatical relation (i.e., subject or object) are subsequently stored in a graph. The edges between event nodes are scored using the *MI* equation given in (1). A fragment of the action graph is shown in Figure 3 (for simplicity, the edges in the example are weighted with co-occurrence frequencies). Contrary to Chambers and Jurafsky (2008) we do not learn *global* narrative chains over an entire corpus. Currently, we consider *local* chains of length two and three (i.e., chains of two or three events sharing grammatical arguments). The generator consults the graph when selecting a verb for an entity. It will favor verbs that are part of an event chain (e.g., SUBJ:chase $\rightarrow$ SUBJ:run $\rightarrow$ SUBJ:fall in Figure 3). This way, the search space is effectively pruned as finding a suitable verb in the current sentence is influenced by the choice of verb in the next sentence.

[2]See http://opennlp.sourceforge.net/.

2.2 Sentence Planning

So far we have described how we gather knowledge about entities and their interactions, which must be subsequently combined into a sentence. The backbone of our sentence planner is a grammar with subcategorization information which we collected from the lexicon created by Korhonen and Briscoe (2006) and the COMLEX dictionary (Grishman et al., 1994). The grammar rules act as templates. They each take a verb as their head and propose ways of filling its argument slots. This means that when generating a story, the choice of verb will affect the structure of the sentence. The subcategorization templates are weighted by their probability of occurrence in the reference dictionaries. This allows the system to prefer less elaborate grammatical structures. The grammar rules were converted to a format compatible with our surface realizer (see Section 2.3) and include information pertaining to mood, agreement, argument role, etc.

Our sentence planner aggregates together information from the knowledge base, without however generating referring expressions. Although this would be a natural extension, we initially wanted to assess whether the stochastic approach advocated here is feasible at all, before venturing towards more ambitious components.

2.3 Surface Realization

The surface realization process is performed by RealPro (Lavoie and Rambow (1997)). The system takes an abstract sentence representation and transforms it into English. There are several grammatical issues that will affect the final realization of the sentence. For nouns we must decide whether they are singular or plural, whether they are preceded by a definite or indefinite article or with no article at all. Adverbs can either be pre-verbal or post-verbal. There is also the issue of selecting an appropriate tense for our generated sentences, however, we simply assume all sentences are in the present tense. Since we do not know a priori which of these parameters will result in a grammatical sentence, we generate all possible combinations and select the most likely one according to a language model. We used the SRI toolkit to train a trigram language model on the British National Corpus, with interpolated Kneser-Ney smoothing and perplexity as the scoring metric for the generated sentences.

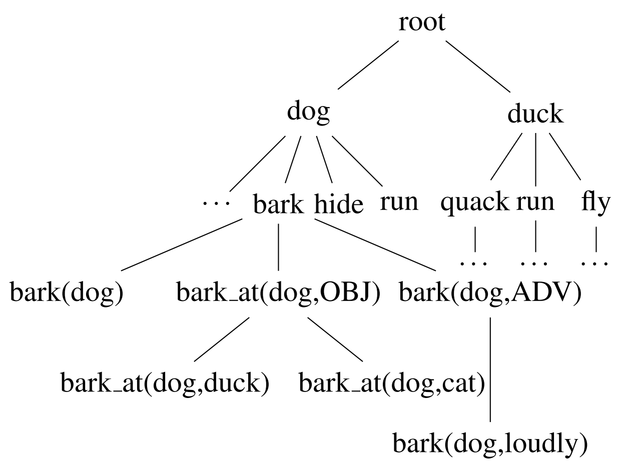

Figure 4: Simplified generation example for the input sentence *the dog chases the duck*.

2.4 Sentence Generation Example

It is best to illustrate the generation procedure with a simple example (see Figure 4). Given the sentence *the dog chases the duck* as input, our generator assumes that either *dog* or *duck* will be the subject of the following sentence. This is a somewhat simplistic attempt at generating coherent stories. Centering (Grosz et al., 1995) and other discourse theories argue that topical entities are likely to appear in prominent syntactic positions such as subject or object. Next, we select verbs from the knowledge base that take the words *duck* and *dog* as their subject (e.g., *bark, run, fly*). Our beam search procedure will reduce the list of verbs to a small subset by giving preference to those that are likely to follow *chase* and have *duck* and *dog* as their subjects or objects.

The sentence planner gives a set of possible frames for these verbs which may introduce additional entities (see Figure 4). For example, *bark* can be intransitive or take an object or adverbial complement. We select an object for *bark*, by retrieving from the knowledge base the set of objects it co-occurs with. Our surface realizer will take structures like "bark(dog,loudly)", "bark_at(dog,cat)", "bark_at(dog,duck)" and generate the sentences *the dog barks loudly*, *the dog barks at the cat* and *the dog barks at the duck*. This procedure is repeated to create a list of possible candidates for the third sentence, and so on.

As Figure 4 illustrates, there are many candidate sentences for each entity. In default of generating all of these exhaustively, our system utilizes the *MI* scores from the knowledge base to guide the search. So, at each choice point in the generation process, e.g., when selecting a verb for an entity or a frame for a verb, we consider the *N* best alternatives assuming that these are most likely to appear in a good story.

3 Story Ranking

We have so far described most modules of our story generator, save one important component, namely the story ranker. As explained earlier, our generator produces stories stochastically, by relying on co-occurrence frequencies collected from the training corpus. However, there is no guarantee that these stories will be interesting or coherent. Engaging stories have some element of surprise and originality in them (Turner, 1994). Our stories may simply contain a list of actions typically performed by the story characters. Or in the worst case, actions that make no sense when collated together.

Ideally, we would like to be able to discern interesting stories from tedious ones. Another important consideration is their coherence. We have to ensure that the discourse smoothly transitions from one topic to the next. To remedy this, we developed two ranking functions that assess the candidate stories based on their interest and coherence. Following previous work (Stent et al., 2004; Barzilay and Lapata, 2007) we learn these ranking functions from training data (i.e., stories labeled with numeric values for interestingness and coherence).

Interest Model A stumbling block to assessing how interesting a story may be, is that the very notion of interestingness is subjective and not very well understood. Although people can judge fairly reliably whether they like or dislike a story, they have more difficulty isolating what exactly makes it interesting. Furthermore, there are virtually no empirical studies investigating the linguistic (surface level) correlates of interestingness. We therefore conducted an experiment where we asked participants to rate a set of human authored stories in terms of interest. Our stories were Aesop's fables since they resemble the stories we wish to generate. They are fairly short (average length was 3.7 sentences) and with a few characters. We asked participants to judge 40 fables on a set of criteria: plot, events, characters, coherence and interest (using a 5-point rating scale). The fables were split into 5 sets of 8; each participant was randomly assigned one of the 5 sets to judge. We obtained ratings (440 in total) from 55 participants, using the WebExp[3] experimental software.

We next investigated if easily observable syntactic and lexical features were correlated with interest. Participants gave the fables an average interest rating of 3.05. For each story we extracted the number of tokens and types for nouns, verbs, adverbs and adjectives as well as the number of verb-subject and verb-object relations. Using the MRC Psycholinguistic database[4] tokens were also annotated along the following dimensions: number of letters (NLET), number of phonemes (NPHON), number of syllables (NSYL), written frequency in the Brown corpus (Kucera and Francis 1967; K-F-FREQ), number of categories in the Brown corpus (K-F-NCATS), number of samples in the Brown corpus (K-F-NSAMP), familiarity (FAM), concreteness (CONC), imagery (IMAG), age of acquisition (AOA), and meaningfulness (MEANC and MEANP).

Correlation analysis was used to assess the degree of linear relationship between interest ratings and the above features. The results are shown in Table 2. As can be seen the highest predictor is the number of objects in a story, followed by the number of noun tokens and types. Imagery, concreteness and familiarity all seem to be significantly correlated with interest. Story length was not a significant predictor. Regressing the best predictors from Table 2 against the interest ratings yields a correlation coefficient of 0.608 ($p < 0.05$). The predictors account uniquely for 37.2% of the variance in interest ratings. Overall, these results indicate that a model of story interest can be trained using shallow syntactic and lexical features. We used the Aesop's fables with the human ratings as training data from which we extracted features that shown to be significant predictors in our correlation analysis. Word-based features were summed in order to obtain a representation for the entire story. We used Joachims's (2002) SVM^{light} package for training with cross-validation (all parameters set to their default values). The model achieved a correlation of 0.948 (Kendall's tau) with the human ratings on the test set.

Coherence Model As well as being interesting we have to ensure that our stories make sense to the reader. Here, we focus on *local coherence*, which captures text organization at the level of sentence to sentence transitions. We created a model of local coherence using using the Entity Grid approach described in Barzilay and Lapata (2007). This approach represents each document as a two-dimensional array in which the columns correspond to entities and the rows to sentences. Each cell indicates whether an entity appears in a given sentence or not and whether it is a subject, object or neither. This entity grid is then converted into a vector of entity transition sequences. Training the model required examples of both coherent and incoherent stories. An artificial training set was created by permuting the sentences of coherent stories, under the assumption that the original story is more coherent than its permutations. The model was trained and tested on the Andrew Lang fairy tales collection[5] on a random split of the data. It ranked the original stories higher than their corresponding permutations 67.40% of the time.

	Interest
NTokens	0.188**
NTypes	0.173**
VTokens	0.123*
VTypes	0.154**
AdvTokens	0.056
AdvTypes	0.051
AdjTokens	0.035
AdjTypes	0.029
NumSubj	0.150**
NumObj	0.240**
MEANC	0.169**

	Interest
NLET	0.120*
NPHON	0.140**
NSYL	0.125**
K-F-FREQ	0.054
K-F-NCATS	0.137**
K-F-NSAMP	0.103*
FAM	0.162**
CONC	0.166**
IMAG	0.173**
AOA	0.111*
MEANP	0.156**

Table 2: Correlation values for the human ratings of interest against syntactic and lexical features; $^{*}: p < 0.05$, $^{**}: p < 0.01$.

4 Experimental Setup

In this section we present our experimental set-up for assessing the performance of our story generator. We give details on our training corpus, system, parameters (such as the width of the beam), the baselines used for comparison, and explain how our system output was evaluated.

Corpus The generator was trained on 437 stories from the Andrew Lang fairy tale corpus.[6] The stories had an average length of 125.18 sentences. The corpus contained 15,789 word tokens. We

[3]See `http://www.webexp.info/`.

[4]`http://www.psy.uwa.edu.au/mrcdatabase/uwa_mrc.htm`

[5]Aesop's fables were too short to learn a coherence model.

[6]See `http://www.mythfolklore.net/andrewlang/`.

discarded word tokens that did not appear in the Children's Printed Word Database[7], a database of printed word frequencies as read by children aged between five and nine.

Story search When searching the story space, we set the beam width to 500. This means that we allow only 500 sentences to be considered at a particular depth before generating the next set of sentences in the story. For each entity we select the five most likely events and event sequences. Analogously, we consider the five most likely subcategorization templates for each verb. Considerable latitude is available when applying the ranking functions. We may use only one of them, or one after the other, or both of them. To evaluate which system configuration was best, we asked two human evaluators to rate (on a 1–5 scale) stories produced in the following conditions: (a) score the candidate stories using the interest function first and then coherence (and vice versa), (b) score the stories simultaneously using both rankers and select the story with the highest score. We also examined how best to prune the search space, i.e., by selecting the highest scoring stories, the lowest scoring one, or simply at random. We created ten stories of length five using the fairy tale corpus for each permutation of the parameters. The results showed that the evaluators preferred the version of the system that applied both rankers simultaneously and maintained the highest scoring stories in the beam.

Baselines We compared our system against two simpler alternatives. The first one does not use a beam. Instead, it decides deterministically how to generate a story on the basis of the most likely predicate-argument and predicate-predicate counts in the knowledge base. The second one creates a story randomly without taking any co-occurrence frequency into account. Neither of these systems therefore creates more than one story hypothesis whilst generating.

Evaluation The system generated stories for 10 input sentences. These were created using commonly occurring sentences in the fairy tales corpus (e.g., *The family has the baby*, *The monkey climbs the tree*, *The giant guards the child*). Each system generated one story for each sentence resulting in 30 (3×10) stories for evaluation. All stories had the same length, namely five sentences. Human judges (21 in total) were asked to rate the stories on a scale of 1 to 5 for fluency (was the sentence grammatical?), coherence (does the story make sense overall?) and interest (how interesting is the story?). The stories were presented in random order. Participants were told that all stories were generated by a computer program. They were instructed to rate more favorably interesting stories, stories that were comprehensible and overall grammatical.

[7]`http://www.essex.ac.uk/psychology/cpwd/`

System	Fluency	Coherence	Interest
Random	1.95*	2.40*	2.09*
Deterministic	2.06*	2.53*	2.09*
Rank-based	2.20	2.65	2.20

Table 3: Human evaluation results: mean story ratings for three versions of our system; *: significantly different from Rank-based.

5 Results

Our results are summarized in Table 3 which lists the average human ratings for the three systems. We performed an Analysis of Variance (ANOVA) to examine the effect of system type on the story generation task. Statistical tests were carried out on the mean of the ratings shown in Table 3 for fluency, coherence, and interest. We observed a reliable effect of system type by subjects and items on all three dimensions. Post-hoc Tukey tests revealed that the stories created with our rank-based system are perceived as significantly better in terms of fluency, interest, and coherence than those generated by both the deterministic and random systems ($\alpha < 0.05$). The deterministic system is not significantly better than the random one except in terms of coherence.

These results are not entirely surprising. The deterministic system maintains a local restricted view of what constitutes a good story. It creates a story by selecting isolated entity-event relationships with high *MI* scores. As a result, the stories are unlikely to have a good plot. Moreover, it tends to primarily favor verb-object or verb-subject relations, since these are most frequent in the corpus. The stories thus have little structural variation and feel repetitive. The random system uses even less information in generating a story (entity-action relationships are chosen at random without taking note of the *MI* scores). In contrast to these baselines, the rank-based system assesses candidate stories more globally. It thus favors coherent stories, with varied word choice and structure.

	The family has the baby	*The giant guards the child*
Random	The family has the baby. The family is how to empty up to a fault. The baby vanishes into the cave. The family meets with a stranger. The baby says for the boy to fancy the creature.	The giant guards the child. The child calls for the window to order the giant. The child suffers from a pleasure. The child longer hides the forest. The child reaches presently.
Determ	The family has the baby. The family rounds up the waist. The family comes in. The family wonders. The family meets with the terrace.	The giant guards the child. The child rescues the clutch. The child beats down on a drum. The child feels out of a shock. The child hears from the giant.
Rank-based	The family has the baby. The baby is to seat the lady at the back. The baby sees the lady in the family. The family marries a lady for the triumph. The family quickly wishes the lady vanishes.	The giant guards the child. The child rescues the son from the power. The child begs the son for a pardon. The giant cries that the son laughs the happiness out of death. The child hears if the happiness tells a story.

Table 4: Stories generated by the random, deterministic, and rank-based systems.

A note of caution here concerns referring expressions which our systems cannot at the moment generate. This may have disadvantaged the stories overall, rendering them stylistically awkward.

The stories generated by both the deterministic and random systems are perceived as less interesting in comparison to the rank-based system. This indicates that taking interest into account is a promising direction even though the overall interestingness of the stories we generate is somewhat low (see third column in Table 3). Our interest ranking function was trained on well-formed human authored stories. It is therefore possible that the ranker was not as effective as it could be simply because it was applied to out-of-domain data. An interesting extension which we plan for the future is to evaluate the performance of a ranker trained on machine generated stories.

Table 4 illustrates the stories generated by each system for two input sentences. The rank-based stories read better overall and are more coherent. Our subjects also gave them high interest scores. The deterministic system tends to select simplistic sentences which although read well by themselves do not lead to an overall narrative. Interestingly, the story generated by the random system for the input *The family has the baby*, scored high on interest too. The story indeed contains interesting imagery (e.g. *The baby vanishes into the cave*) although some of the sentences are syntactically odd (e.g. *The family is how to empty up to a fault*).

6 Conclusions and Future Work

In this paper we proposed a novel method to computational story telling. Our approach has three key features. Firstly, story plot is created dynamically by consulting an automatically created knowledge base. Secondly, our generator realizes the various components of the generation pipeline stochastically, without extensive manual coding. Thirdly, we generate and store multiple stories efficiently in a tree data structure. Story creation amounts to traversing the tree and selecting the nodes with the highest score. We develop two scoring functions that rate stories in terms of how coherent and interesting they are. Experimental results show that these bring improvements over versions of the system that rely solely on the knowledge base. Overall, our results indicate that the overgeneration-and-ranking approach advocated here is viable in producing short stories that exhibit narrative structure. As our system can be easily rertrained on different corpora, it can potentially generate stories that vary in vocabulary, style, genre, and domain.

An important future direction concerns a more detailed assessment of our search procedure. Currently we don't have a good estimate of the type of stories being overlooked due to the restrictions we impose on the search space. An appealing alternative is the use of Genetic Algorithms (Goldberg, 1989). The operations of mutation and crossover have the potential of creating more varied and original stories. Our generator would also benefit from an explicit model of causality which is currently approximated by the entity chains. Such a model could be created from existing resources such as ConceptNet (Liu and Davenport, 2004), a freely available commonsense knowledge base. Finally, improvements such as the generation of referring expressions and the modeling of selectional restrictions would create more fluent stories.

Acknowledgements The authors acknowledge the support of EPSRC (grant GR/T04540/01). We are grateful to Richard Kittredge for his help with RealPro. Special thanks to Johanna Moore for insightful comments and suggestions.

References

Asher, Nicholas and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Barzilay, Regina and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the HLT/EMNLP*. Vancouver, pages 331–338.

Barzilay, Regina and Mirella Lapata. 2007. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Briscoe, E. and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC*. Las Palmas, Gran Canaria, pages 1499–1504.

Callaway, Charles B. and James C. Lester. 2002. Narrative prose generation. *Artificial Intelligence* 2(139):213–252.

Chambers, Nathanael and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*. Columbus, OH, pages 789–797.

Duboue, Pablo A. and Kathleen R. McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the 2nd INLG*. Ramapo Mountains, NY.

Fass, S. 2002. *Virtual Storyteller: An Approach to Computational Storytelling*. Master's thesis, Dept. of Computer Science, University of Twente.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Grishman, Ralph, Catherine Macleod, and Adam Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th COLING*. Kyoto, Japan, pages 268–272.

Grosz, Barbara J., Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2):203–225.

Joachims, Thorsten. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD*. Edmonton, AL, pages 133–142.

Knight, Kevin and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd ACL*. Cambridge, MA, pages 252–260.

Korhonen, Y. Krymolowski, A. and E.J. Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of the 5th LREC*. Genova, Italy.

Kucera, Henry and Nelson Francis. 1967. *Computational Analysis of Present-day American English*. Brown University Press, Providence, RI.

Lavoie, Benoit and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th ANCL*. Washington, D.C., pages 265–268.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th COLING*. Montréal, QC, pages 768–774.

Liu, Hugo and Glorianna Davenport. 2004. ConceptNet: a practical commonsense reasoning toolkit. *BT Technology Journal* 22(4):211–226.

Mellish, Chris, Alisdair Knott, Jon Oberlander, and Mick O'Donnell. 1998. Experiments using stochastic search for text planning. In Eduard Hovy, editor, *Proceedings of the 9th INLG*. New Brunswick, NJ, pages 98–107.

Oinonen, K.M., M. Theune, A. Nijholt, and J.R.R. Uijlings. 2006. Designing a story database for use in automatic story generation. In R. Harper, M. Rauterberg, and M. Combetto, editors, *Entertainment Computing – ICEC 2006*. Springer Verlag, Berlin, volume 4161 of *Lecture Notes in Computer Science*, pages 298–301.

Propp, Vladimir. 1968. *The Morphology of Folk Tale*. University of Texas Press, Austin, TX.

Reiter, E and R Dale. 2000. *Building Natural-Language Generation Systems*. Cambridge University Press.

Robertson, Judy and Judith Good. 2003. Ghostwriter: A narrative virtual environment for children. In *Proceedings of IDC2003*. Preston, England, pages 85–91.

Shim, Yunju and Minkoo Kim. 2002. Automatic short story generator based on autonomous agents. In *Proceedings of PRIMA*. London, UK, pages 151–162.

Stent, Amanda, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd ACL*. Barcelona, Spain, pages 79–86.

Theune, M., S. Faas, D.K.J. Heylen, and A. Nijholt. 2003. The virtual storyteller: Story creation by intelligent agents. In S. Gbel, N. Braun, U. Spierling, J. Dechau, and H. Diener, editors, *TIDSE-2003*. Fraunhofer IRB Verlag, Darmstadt, pages 204–215.

Thorndyke, Perry W. 1977. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology* 9(1):77–110.

Turner, Scott T. 1994. *The creative process: A computer model of storytelling and creativity*. Erlbaum, Hillsdale, NJ.