Plot Induction and Evolutionary Search for Story Generation

Plot Induction and Evolutionary Search for 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

In this paper we develop a story generator that leverages knowledge inherent in corpora without requiring extensive manual involvement. A key feature in our approach is the reliance on a story planner which we acquire automatically by recording events, their participants, and their precedence relationships in a training corpus. Contrary to previous work our system does not follow a generate-and-rank architecture. Instead, we employ evolutionary search techniques to explore the space of possible stories which we argue are well suited to the story generation task. Experiments on generating simple children's stories show that our system outperforms previous data-driven approaches.

1 Introduction

Computer story generation has met with fascination since the early days of artificial intelligence. Indeed, over the years, several generators have been developed capable of creating stories that resemble human output. To name only a few, TALE-SPIN (Meehan, 1977) generates stories through problem solving, MINSTREL (Turner, 1992) relies on an episodic memory scheme, essentially a repository of previous hand-coded stories, to solve the problems in the current story, and MAKEBELIEVE (Liu and Singh, 2002) uses commonsense knowledge to generate short stories from an initial seed story (supplied by the user). A large body of more recent work views story generation as a form of agent-based planning (Swartjes and Theune, 2008; Pizzi et al., 2007). The agents act as characters with a list of goals. They form plans of action and try to fulfill them. Interesting stories emerge as plans interact and cause failures and possible replanning.

The broader appeal of computational story generation lies in its application potential. Examples include the entertainment industry and the development of tools that produce large numbers of plots automatically that might provide inspiration to professional screen writers (Agudo et al., 2004); rendering video games more interesting by allowing the plot to adapt dynamically to the players' actions (Barros and Musse, 2007); and assisting teachers to create or personalize stories for their students (Riedl and Young, 2004).

A major stumbling block for the widespread use of computational story generators is their reliance on expensive, manually created resources. A typical story generator will make use of a knowledge base for providing detailed domain-specific information about the characters and objects involved in the story and their relations. It will also have a story planner that specifies how these characters interact, what their goals are and how their actions result in different story plots. Finally, a sentence planner (coupled with a surface realizer) will render an abstract story specification into natural language text. Traditionally, most of this knowledge is created by hand, and the effort must be repeated for new domains, new characters and plot elements.

Fortunately, recent work in natural language processing has taken significant steps towards developing algorithms that learn some of this knowledge *automatically* from natural language corpora. Chambers and Jurafsky (2009, 2008) propose an unsupervised method for learning *narrative schemas*, chains of events whose arguments are filled with participant semantic roles defined over words. An example schema is {X arrest, X charge, X raid, X seize, X confiscate, X detain, X deport}, where X stands for the argument types {*police, agent, authority, government*}. Their approach relies on the intuition that in a coherent text events that are about the same participants are

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1562–1572,
Uppsala, Sweden, 11-16 July 2010.

likely to be part of the same story or narrative. Their model extracts narrative chains, essentially events that share argument slots and merges them into schemas. The latter could be used to construct or enrich the knowledge base of a story generator.

In McIntyre and Lapata (2009) we presented a story generator that leverages knowledge inherent in corpora without requiring extensive manual involvement. The generator operates over predicate-argument and predicate-predicate co-occurrence tuples gathered from training data. These are used to produce a large set of candidate stories which are subsequently ranked based on their interestingness and coherence. The approach is unusual in that it does not involve an explicit story planning component. Stories are created stochastically by selecting entities and the events they are most frequently attested with.

In this work we develop a story generator that is also data-driven but crucially relies on a story planner for creating meaningful stories. Inspired by Chambers and Jurafsky (2009) we acquire story plots automatically by recording events, their participants, and their precedence relationships as attested in a training corpus. Entities give rise to different potential plots which in turn generate multiple stories. Contrary to our previous work (McIntyre and Lapata, 2009), we do not follow a generate-and-rank architecture. Instead, we search the space of possible stories using Genetic Algorithms (GAs) which we argue are advantageous in the story generation setting, as they can search large fitness landscapes while greatly reducing the risk of getting stuck in local optima. By virtue of exploring the search space more broadly, we are able to generate creative stories without an explicit interest scoring module.

In the remainder of this paper we give a brief overview of the system described in McIntyre and Lapata (2009) and discuss previous applications of GAs in natural language generation (Section 2). Next, we detail our approach, specifically how plots are created and used in conjunction with genetic search (Sections 3 and 4). Finally, we present our experimental results (Sections 6 and 7) and conclude the paper with discussion of future work.

2 Related Work

Our work builds on and extends the story generator developed in McIntyre and Lapata (2009). The system creates simple children's stories in an interactive context: the user supplies the topic of the story and its desired length (number of sentences). The generator creates a story following a pipeline architecture typical of natural language generation systems (Reiter and Dale, 2000) consisting of content selection, sentence planning, and surface realization.

The content of a story is determined by consulting a data-driven knowledge base that records the entities (i.e., nouns) appearing in a corpus and the actions they perform. These are encoded as dependency relations (e.g., *subj-verb*, *verb-obj*). In order to promote between-sentence coherence the generator also make use of an *action graph* that contains action-role pairs and the likelihood of transitioning from one to another. The sentence planner aggregates together entities and their actions into a sentence using phrase structure rules. Finally, surface realization is performed by interfacing RealPro (Lavoie and Rambow, 1997) with a language model. The system searches for the best story overall as well as the best sentences that can be generated from the knowledge base. Unlikely stories are pruned using beam search. In addition, stories are reranked using two scoring functions based on coherence and interest. These are learnt from training data, i.e., stories labeled with numeric values for interest and coherence.

Evolutionary search techniques have been previously employed in natural language generation, especially in the context of document planning. Structuring a set of facts into a coherent text is effectively a search problem that may lead to combinatorial explosion for large domains. Mellish et al. (1998) (and subsequently Karamanis and Manurung 2002) advocate genetic algorithms as an alternative to exhaustively searching for the optimal ordering of descriptions of museum artefacts. Rather than requiring a global optimum to be found, the genetic algorithm selects an order (based on coherence) that is good enough for people to understand. Cheng and Mellish (2000) focus on the interaction of aggregation and text planning and use genetic algorithms to search for the best aggregated document that satisfies coherence constraints.

The application of genetic algorithms to story generation is novel to our knowledge. Our work also departs from McIntyre and Lapata (2009) in two important ways. Firstly, our generator does not rely on a knowledge base of seemingly unrelated entities and relations. Rather, we employ

a document planner to create and structure a plot for a story. The planner is built automatically from a training corpus and creates plots dynamically depending on the protagonists of the story. Secondly, our search procedure is simpler and more global; instead of searching for the best story twice (i.e., by first finding the *n*-best stories and then subsequently reranking them based on coherence and interest), our genetic algorithm explores the space of possible stories once.

3 Plot Generation

Following previous work (e.g., Shim and Kim 2002; McIntyre and Lapata 2009) we assume that the user supplies a sentence (e.g., *the princess loves the prince*) from which the system creates a story. Each entity in this sentence (e.g., *princess*, *prince*) is associated with its own *narrative schema*, a set of key events and actors co-occurring with it in the training corpus. Our narrative schemas differ slightly from Chambers and Jurafsky (2009). They acquire schematic representations of situations akin to FrameNet (Fillmore et al., 2003): schemas consists of semantically similar predicates and the entities evoked by them. In our setting, every entity has its own schema, and predicates associated with it are ordered. Plots are generated by merging the entity-specific narrative schemas which subsequently serve as the input to the genetic algorithm. In the following we describe how the narrative schemas are extracted and plots merged, and then discuss our evolutionary search procedure.

Entity-based Schema Extraction Before we can generate a plot for a story we must have an idea of the actions associated with the entities in the story, the order in which these actions are performed and also which other entities can participate. This information is stored in a directed graph which we explain below. Our algorithm processes each document at a time, it operates over dependency structures and assumes that entity mentions have been resolved. In our experiments we used Rasp (Briscoe and Carroll, 2002), a broad coverage dependency parser, and the OpenNLP[1] coreference resolution engine.[2] However, any dependency parser or coreference tool could serve our purpose. We also assume that the actions associated with a given entity are ordered and that linear order corresponds to temporal order. This is a gross simplification as it is well known that temporal relationships between events are not limited to precedence, they may overlap, occur simultaneously, or be temporally unrelated. We could have obtained a more accurate ordering using a temporal classifier (see Chambers and Jurafsky 2008), however we leave this to future work.

For each entity e in the corpus we build a directed graph $G = (V, E)$ whose nodes V denote predicate argument relationships, and edges E represent transitions from node V_i to node V_j. As an example of our schema construction process, consider a very small corpus consisting of the two documents shown in Figure 1. The schema for *princess* after processing the first document is given on the left hand side. Each node in this graph corresponds to an action attested with *princess* (we also record who performs it and where or how). Nodes are themselves dependency trees (see Figure 4a), but are linearized in the figure for the sake of brevity. Edges in the graph indicate ordering and are weighted using the mutual information metric proposed in Lin (1998) (the weights are omitted from the example).[3] The first sentence in the text gives rise to the first node in the graph, the second sentence to the second node, and so on. Note that the third sentence is not present in the graph as it is not about the *princess*.

When processing the second document, we simply expand this graph. Before inserting a new node, we check if it can be merged with an already existing one. Nodes are merged only if they have the same verb and similar arguments, with the focal entity (i.e., *princess*) appearing in the same argument slot. In our example, the nodes "prince marry princess in castle" and "prince marry princess in temple" can be merged as they contain the same verb and number of similar arguments. The nodes "princess have influence" and "princess have baby" cannot be merged as *influence* and *baby* are semantically unrelated. We compute argument similarity using WordNet (Fellbaum, 1998) and the measure proposed by Wu and Palmer (1994) which is based on path length. We merge nodes with related arguments only if their similarity exceeds a threshold (determined empirically).

[1] See `http://opennlp.sourceforge.net/`.

[2] The coreference resolution tool we employ is not error-free and on occasion will fail to resolve a pronoun. We map unresolved pronouns to the generic labels *person* or *object*.

[3] We use mutual information to identify event sequences strongly associated with the graph entity.

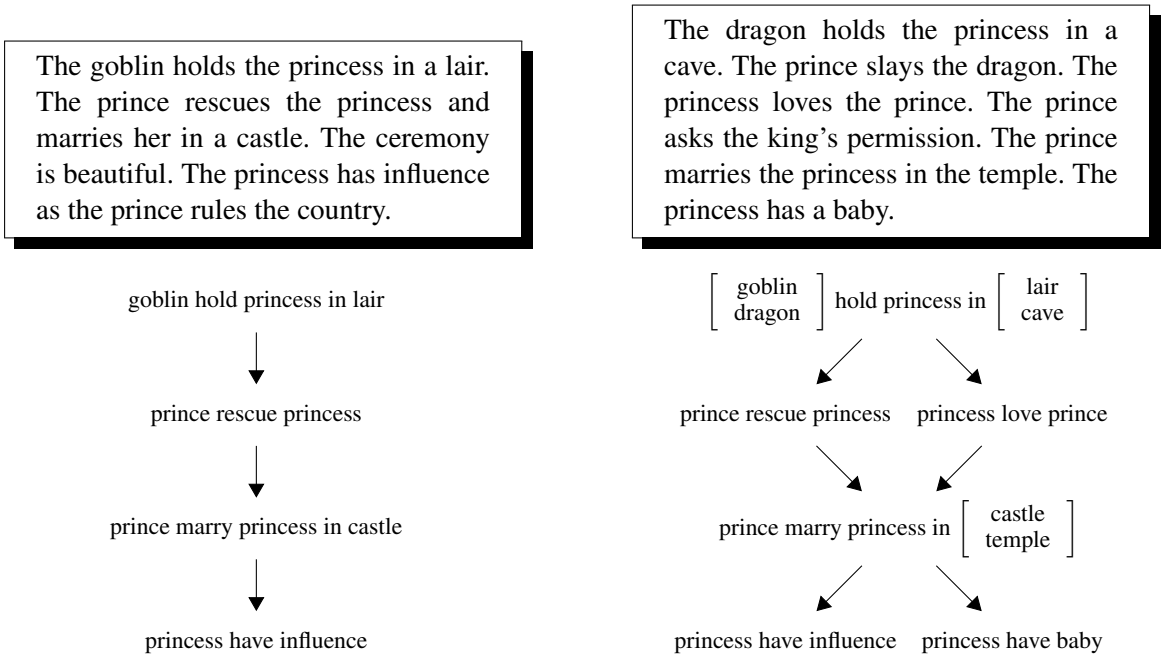

Figure 1: Example of schema construction for the entity *princess*

The schema construction algorithm terminates when graphs like the ones shown in Figure 1 (right hand side) have been created for all entities in the corpus.

Building a Story Plot Our generator takes an input sentence and uses it to instantiate several plots. We achieve this by merging the schemas associated with the entities in the sentence into a plot graph. As an example, consider again the sentence *the princess loves the prince* which requires combing the schemas representing *prince* and *princess* shown in Figures 2 and 1 (right hand side), respectively. Again, we look for nodes that can be merged based on the identity of the actions involved and the (WordNet) similarity of their arguments. However, we disallow the merging of nodes with focal entities appearing in the same argument slot (e.g., "[prince, princess] cries").

Once the plot graph is created, a depth first search starting from the node corresponding to the input sentence, finds all paths with length matching the desired story length (cycles are disallowed). Assuming we wish to generate a story consisting of three sentences, the graph in Figure 3 would create four plots. These are (princess love prince, prince marry princess in [castle, temple], princess have influence), (princess love prince, prince marry princess in [castle, temple], princess have baby), (princess love prince, prince marry princess in [castle, temple], prince rule country), and (princess love prince, prince ask king's permission prince marry princess in [castle, temple]). Each of these plots represents two different stories one with *castle* and one with *temple* in it.

Sentence Planning The sentence planner is interleaved with the story planner and influences the final structure of each sentence in the story. To avoid generating short sentences — note that nodes in the plot graph consist of a single action and would otherwise correspond to a sentence with a single clause — we combine pairs of nodes within the same graph by looking at intrasentential verb-verb co-occurrences in the training corpus. For example, the nodes (prince have problem, prince keep secret) could become the sentence *the prince has a problem keeping a secret*. We leave it up to the sentence planner to decide how the two actions should be combined.[4] The sentence planner will also insert adverbs and adjectives, using co-occurrence likelihoods acquired from the training corpus. It is essentially a phrase structure grammar compiled from the lexical resources made available by Korhonen and Briscoe (2006) and Grishman et al. (1994). The grammar rules act as templates for combining clauses and filling argument slots.

[4] We only turn an action into a subclause if its subject entity is same as that of the previous action.

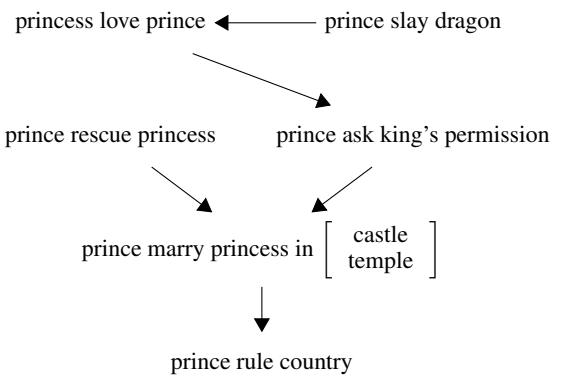

Figure 2: Narrative schema for the entity *prince*.

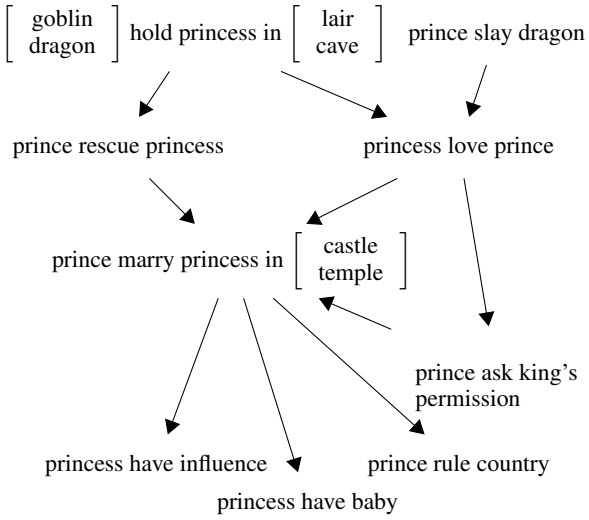

Figure 3: Plot graph for the input sentence *the princess loves the prince*.

4 Genetic Algorithms

The example shown in Figure 3 is a simplified version of a plot graph. The latter would normally contain hundreds of nodes and give rise to thousands of stories once lexical variables have been expanded. Searching the story space is a difficult optimization problem, that must satisfy several constraints: the story should be of a certain length, overall coherent, creative, display some form of event progression, and generally make sense. We argue that evolutionary search is appealing here, as it can find global optimal solutions in a more efficient way than traditional optimization methods.

In this study we employ genetic algorithms (GAs) a well-known search technique for finding approximate (or exact) solutions to optimization problems. The basic idea behind GAs is based on "natural selection" and the Darwinian principle of the survival of the fittest (Mitchell, 1998). An initial population is randomly created containing a predefined number of individuals (or solutions), each represented by a genetic string (e.g., a population of chromosomes). Each individual is evaluated according to an objective function (also called a fitness function). A number of individuals are then chosen as parents from the population according to their fitness, and undergo *crossover* (also called recombination) and *mutation* in order to develop the new population. Offspring with better fitness are then inserted into the population, replacing the inferior individuals in the previous generation.

The algorithm thus identifies the individuals with the optimizing fitness values, and those with lower fitness will naturally get discarded from the population. This cycle is repeated for a given number of generations, or stopped when the solution obtained is considered optimal. This process leads to the evolution of a population in which the individuals are more and more suited to their environment, just as natural adaptation. We describe below how we developed a genetic algorithm for our story generation problem.

Initial Population Rather than start with a random population, we seed the initial population with story plots generated from our plot graph. For an input sentence, we generate all possible plots. The latter are then randomly sampled until a population of the desired size is created. Contrary to McIntyre and Lapata (2009), we initialize the search with complete stories, rather than generate one sentence at a time. The genetic algorithm will thus avoid the pitfall of selecting early on a solution that will later prove detrimental.

Crossover Each plot is represented as an ordered graph of dependency trees (corresponding to sentences). We have decided to use crossover of a single point between two selected parents. The children will therefore contain sentences up to the crossover point of the first parent and sentences after that point of the second. Figure 4a shows two parents (prince rescue princess, prince marry princess in castle, princess have baby) and (prince rescue princess, prince love princess, princess kiss prince) and how two new plots are created by swapping their last nodes.

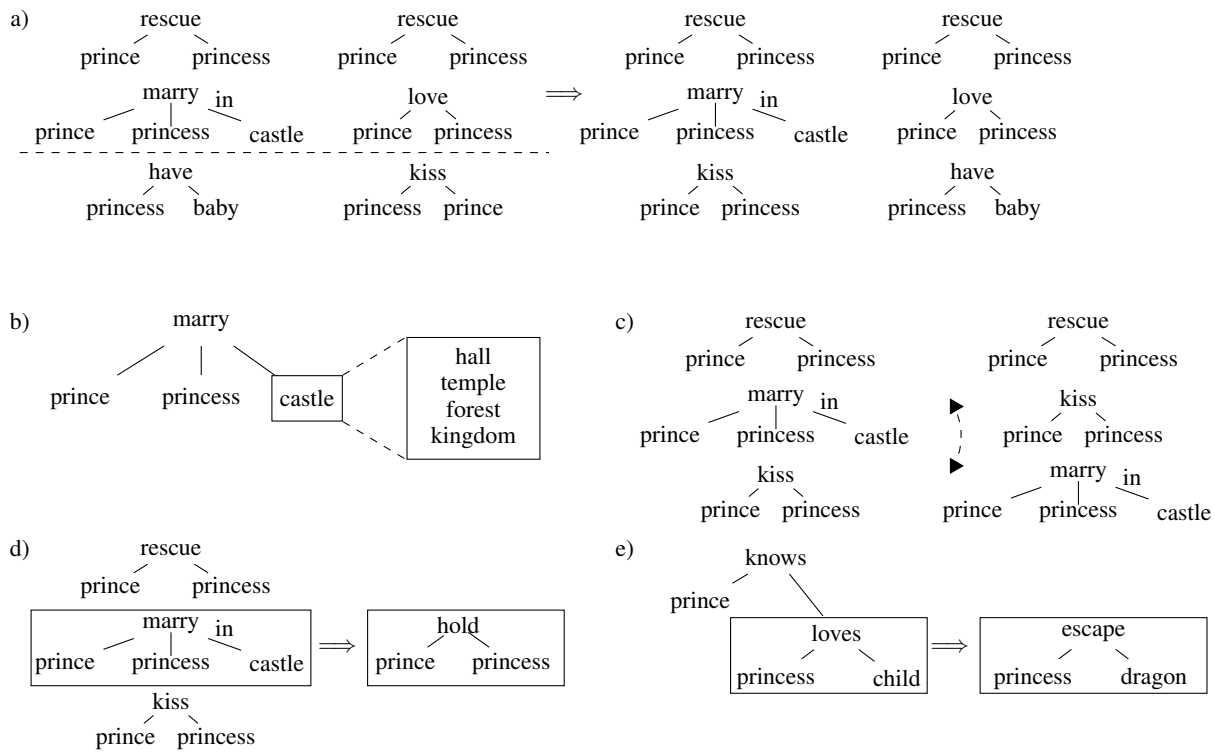

Figure 4: Example of genetic algorithm operators as they are applied to plot structures: a) crossover of two plots on a single point, indicated by the dashed line, resulting in two children which are a recombination of the parents; b) mutation of a lexical node, *church* can be replaced from a list of semantically related candidates; c) sentences can be switched under mutation to create a potentially more coherent structure; d) if the matrix verb undergoes mutation then, a random sentence is generated to replace it; e) if the verb chosen for mutation is the head of a subclause, then a random subclause replaces it.

Mutation Mutation can occur on any verb, noun, adverb, or adjective in the plot. If a noun, adverb or adjective is chosen to undergo mutation, then we simply substitute it with a new lexical item that is sufficiently similar (see Figure 4b for an example). Verbs, however, have structural importance in the stories and we cannot simply replace them without taking account of their arguments. If a matrix verb is chosen to undergo mutation, then a new random sentence is generated to replace the entire sentence (see Figure 4d). If it is a subclause, then it is replaced with a randomly generated clause, headed by a verb that has been seen in the corpus to co-occur with the matrix verb (Figure 4e). The sentence planner selects and fills template trees for generating random clauses. Mutation may also change the order of any two nodes in the graph in the hope that this will increase the story's coherence or create some element of surprise (see Figure 4c).

Selection To choose the plots for the next generation, we used fitness proportional selection (also know as roulette-wheel selection, Goldberg 1989) which chooses candidates randomly but with a bias towards those with a larger proportion of the population's combined fitness. We do not want to always select the fittest candidates as there may be valid partial solutions held within less fit members of the population. However, we did employ some elitism by allowing the top 1% of solutions to be copied straight from one generation to the next. Note that our candidates may also represent invalid solutions. For instance, through crossover it is possible to create a plot in which all or some nodes are identical. If any such candidates are identified, they are assigned a low fitness, without however being eliminated from the population as some could be used to create fitter solutions.

In a traditional GA, the fitness function deals with one optimization objective. It is possible to optimize several objectives either using a vot-

ing model or more sophisticated methods such as Pareto ranking (Goldberg, 1989). Following previous work (Mellish et al., 1998) we used a single fitness function that scored candidates based on their coherence. Our function was learned from training data using the Entity Grid document representation proposed in Barzilay and Lapata (2007). An entity grid is a two-dimensional array in which columns correspond to entities and 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. For training, this representation is converted into a feature vector of entity transition sequences and a model is learnt from examples of coherent and incoherent stories. The latter can be easily created by permuting the sentences of coherent stories (assuming that the original story is more coherent than its permutations).

In addition to coherence, in McIntyre and Lapata (2009) we used a scoring function based on interest which we approximated with lexical and syntactic features such as the number of noun/verb tokens/types, the number of subjects/objects, the number of letters, word familiarity, imagery, and so on. An interest-based scoring function made sense in our previous setup as a means of selecting unusual stories. However, in the context of genetic search such a function seems redundant as interesting stories emerge naturally through the operations of crossover and mutation.

5 Surface Realization

Once the final generation of the population has been reached, the fittest story is selected for surface realization. The realizer takes each sentence in the story and reformulates it into input compatible with the RealPro (Lavoie and Rambow, 1997) text generation engine. Realpro creates several variants of the same story differing in the choice of determiners, number (singular or plural), and prepositions. A language model is then used to select the most probable realization (Knight and Hatzivassiloglou, 1995). Ideally, the realizer should also select an appropriate tense for the sentence. However, we make the simplifying assumption that all sentences are in the present tense.

6 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 population size for the GA search), the baselines used for comparison, and explain how our system output was evaluated.

Corpus The generator was trained on the same corpus used in McIntyre and Lapata (2009), 437 stories from the Andrew Lang fairy tales collection.[5] The average story length is 125.18 sentences. The corpus contains 15,789 word tokens. Following McIntyre and Lapata, we discarded tokens that did not appear in the Children's Printed Word Database[6], a database of printed word frequencies as read by children aged between five and nine. From this corpus we extracted narrative schemas for 667 entities in total. We disregarded any graph that contained less than 10 nodes as too small. The graphs had on average 61.04 nodes, with an average clustering rate[7] of 0.027 which indicates that they are substantially connected.

Parameter Setting Considerable latitude is available when selecting parameters for the GA. These involve the population size, crossover, and mutation rates. To evaluate which setting was best, we asked two human evaluators to rate (on a 1–5 scale) stories produced with a population size ranging from 1,000 to 10,000, crossover rate of 0.1 to 0.6 and mutation rate of 0.001 to 0.1. For each run of the system a limit was set to 5,000 generations. The human ratings revealed that the best stories were produced for a population size of 10,000, a crossover rate of 0.1% and a mutation rate of 0.1%. Compared to previous work (e.g., Karamanis and Manurung 2002) our crossover rate may seem low and the mutation rate high. However, it makes intuitively sense, as high crossover may lead to incoherence by disrupting canonical action sequences found in the plots. On the other hand, a higher mutation will raise the likelihood of a lexical item being swapped for another and may improve overall coherence and interest. The fitness function was trained on 200 documents from the fairy tales collection using Joachims's (2002) SVM^{light} package and entity transition sequences of length 2. The realizer was interfaced with a trigram language model trained on the British National Corpus with the SRI toolkit.

[5] Available from `http://homepages.inf.ed.ac.uk/s0233364/McIntyreLapata09/`.

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

[7] Clustering rate (or transitivity) is the number of triangles in the graph — sets of three vertices each of which is connected to each of the others.

Evaluation We compared the stories generated by the GA against those produced by the rank-based system described in McIntyre and Lapata (2009) and a system that creates stories from the plot graph, without any stochastic search. Since plot graphs are weighted, we can simply select the graph with the highest weight. After expanding all lexical variables, the chosen plot graph will give rise to different stories (e.g., *castle* or *temple* in the example above). We select the story ranked highest according to our coherence function. In addition, we included a baseline which randomly selects sentences from the training corpus provided they contain either of the story protagonists (i.e., entities in the input sentence). Sentence length was limited to 12 words or less as this was on average the length of the sentences generated by our GA system.

Each system created stories for 12 input sentences, resulting in 48 (4×12) stories for evaluation. The sentences used commonly occurring entities in the fairy tales corpus (e.g., *The child watches the bird*, *The emperor rules the kingdom.*, *The wizard casts the spell.*). The stories were split into three sets containing four stories from each system but with only one story from each input sentence. All stories had the same length, namely five sentences. Human judges were presented with one of the three sets and 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 and participants were told that all of them were generated by a computer program. They were instructed to rate more favorably interesting stories, stories that were comprehensible and overall grammatical. The study was conducted over the Internet using WebExp (Keller et al., 2009) and was completed by 56 volunteers, all self reported native English speakers.

7 Results

Our results are summarized in Table 1 which lists the average human ratings for the four 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 1 for fluency, coherence, and interest.

In terms of interest, the GA-based system is significantly better than the Rank-based, Plot-based and Random ones (using a Post-hoc Tukey test, $\alpha < 0.05$). With regard to fluency, the Rank-based system is significantly worse than the rest ($\alpha < 0.01$). Interestingly, the sentences generated by the GA and Plot-based systems are as fluent as those created by humans. Recall that the Random system, simply selects sentences from the training corpus. Finally, the GA system is significantly more coherent than the Rank-based and Random systems ($\alpha < 0.05$), but not the Plot-based one. This is not surprising, the GA and Plot-based systems rely on similar plots to create a coherent story. The performance of the Random system is also inferior as it does not have any explicit coherence enforcing mechanism. The Rank-based system is perceived overall worse. As this system is also the least fluent, we conjecture that participants are influenced in their coherence judgments by the grammaticality of the stories.

System	Fluency	Coherence	Interest
GA-based	3.09	2.48	2.36
Plot-based	3.03	2.36	2.14^{*}
Rank-based	1.96^{**}	1.65^{*}	1.85^{*}
Random	3.10	2.23^{*}	2.20^{*}

Table 1: Human evaluation results: mean story ratings for four story generators; $^{*}: p < 0.05$, $^{**}: p < 0.01$, significantly different from GA-based system.

Overall our results indicate that an explicit story planner improves the quality of the generated stories, especially when coupled with a search mechanism that advantageously explores the search space. It is worth noting that the Plot-based system is relatively simple, however the explicit use of a story plot, seems to make up for the lack of sophisticated search and more elaborate linguistic information. Example stories generated by the four systems are shown in Table 2 for the input sentences *The emperor rules the kingdom* and *The child watches the bird*.

Possible extensions and improvements to the current work are many and varied. Firstly, we could improve the quality of our plot graphs by taking temporal knowledge into account and making use of knowledge bases such as WordNet and ConceptNet (Liu and Davenport, 2004), a freely available commonsense knowledge base. Secondly, our fitness function optimizes one ob-

PlotGA	The emperor rules the kingdom. The kingdom holds on to the emperor. The emperor rides out of the kingdom. The kingdom speaks out against the emperor. The emperor lies.	The child watches the bird. The bird weeps for the child. The child begs the bird to listen.The bird dresses up the child. The child grows up.
Plot	The emperor rules the kingdom. The emperor takes over. The emperor goes on to feel for the kingdom. Possibly the emperor sleeps. The emperor steals.	The child watches the bird. The bird comes to eat away at the child. The child does thoroughly. The bird sees the child. The child sits down.
Rank	The emperor rules the kingdom. The kingdom lives from the reign to the emperor. The emperor feels that the brothers tempt a beauty into the game. The kingdom saves the life from crumbling the earth into the bird. The kingdom forces the whip into wiping the tears on the towel.	The child watches the bird. The bird lives from the reign to the child. The child thanks the victory for blessing the thought. The child loves to hate the sun with the thought. The child hopes to delay the duty from the happiness.
Random	Exclaimed the emperor when Petru had put his question. In the meantime, mind you take good care of our kingdom. At first the emperor felt rather distressed. The dinner of an emperor! Thus they arrived at the court of the emperor.	They cried, “what a beautiful child!” “No, that I cannot do, my child” he said at last. “What is the matter, dear child?” “You wicked child,” cried the Witch. Well, I will watch till the bird comes.

Table 2: Stories generated by a system that uses plots and genetic search (PlotGA), a system that uses only plots (Plot), McIntyre and Lapata (2009)’s rank-based system (Rank) and a system that randomly pastes together sentences from the training corpus (Random).

jective, namely coherence. In the future we plan to explore multiple objectives, such as whether the story is verbose, readable (using existing readability metrics), has two many or two few protagonists, and so on.

Thirdly, our stories would benefit from some explicit modeling of discourse structure. Although the plot graph captures the progression of the actions in a story, we would also like to know where in the story these actions are likely to occur—some tend to appear in the beginning and others in the end. Such information would allow us to structure the stories better and render them more natural sounding. For example, an improvement would be the inclusion of proper endings, as the stories are currently cut off at an arbitrary point when the desired maximum length is reached.

Finally, the fluency of the stories would benefit from generating referring expressions, multiple tense forms, indirect speech, aggregation and generally more elaborate syntactic structure.

References

Agudo, Belén Diáz, Pablo Gervás, and Frederico Peinado. 2004. A case based reasoning approach to story plot generation. In *Proceedings of the 7th European Conference on Case-Based Reasoning*. Springer, Madrid, Spain, pages 142–156.

Barros, Leandro Motta and Soraia Raupp Musse. 2007. Planning algorithms for interactive storytelling. In *Computers in Entertainment (CIE)*, Association for Computing Machinery (ACM), volume 5.

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 International Conference on Language Resources and Evaluation*. Las Palmas, Gran Canaria, pages 1499–1504.

Chambers, Nathanael and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Columbus, Ohio, pages 789–797.

Chambers, Nathanael and Dan Jurafsky. 2009.

Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Singapore, pages 602–610.

Cheng, Hua and Chris Mellish. 2000. Capturing the interaction between aggregation and text planning in two generation systems. In *Proceedings of the 1st International Conference on Natural Language Generation*. Mitzpe Ramon, Israel, pages 186–193.

Fellbaum. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, Cambridge, Massachusetts.

Fillmore, Charles J., Christopher R. Johnson, and Miriam R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography* 16:235–250.

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

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.

Joachims, Thorsten. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. Edmonton, Alberta, pages 133–142.

Karamanis, Nikiforos and Hisar Maruli Manurung. 2002. Stochastic text structuring using the principle of continuity. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'02)*. pages 81–88.

Keller, Frank, Subahshini Gunasekharan, Neil Mayo, and Martin Corley. 2009. Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods* 41(1):1–12.

Knight, Kevin and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*. Cambridge, Massachusetts, 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.

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

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistic*. Montreal, Quebec, pages 768–774.

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

Liu, Hugo and Push Singh. 2002. Using commonsense reasoning to generate stories. In *Proceedings of the 18th National Conference on Artificial Intelligence*. Edmonton, Alberta, pages 957–958.

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

Meehan, James. 1977. An interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*. Cambridge, Massachusetts, pages 91–98.

Mellish, Chris, Alisdair Knott, Jon Oberlander, and Mick O'Donnell. 1998. Experiments using stochastic search for text planning. In *Proceedings of the 9th International Conference on Natural Language Generation*. New Brunswick, New Jersey, pages 98–107.

Mitchell, Melanie. 1998. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, Massachusetts.

Pizzi, David, Fred Charles, Jean-Luc Lugrin, and Marc Cavazza. 2007. Interactive storytelling with literary feelings. In *Proceedings of the 2nd International Conference on Affective Computing and Intelligent Interaction*. Lisbon, Portugal, pages 630–641.

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

Riedl, Mark O. and R. Michael Young. 2004. A planning approach to story generation and history education. In *Proceedings of the 3rd International Conference on Narrative and Interactive Learning Environments*. Edinburgh, UK, pages 41–48.

Shim, Yunju and Minkoo Kim. 2002. Automatic short story generator based on autonomous agents. In *Proceedings of the 5th Pacific Rim International Workshop on Multi Agents*. Tokyo, pages 151–162.

Swartjes, I.M.T. and M. Theune. 2008. The virtual storyteller: story generation by simulation. In *Proceedings of the 20th Belgian-Netherlands Conference on Artificial Intelligence, BNAIC 2008*. Enschede, the Netherlands, pages 257–264.

Turner, Scott R. 1992. *Ministrel: A Computer Model of Creativity and Sotrytelling*. University of California, Los Angeles, California.

Wu, Zhibiao and Martha Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Las Cruces, New Mexico, pages 133–138.