Learning to Tell Tales: Automatic Story Generation from Corpora

Neil McIntyre

Doctor of Philosophy

Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh

2011

Abstract

Automatic story generation has a long-standing tradition in the field of Artificial Intelligence. The ability to create stories on demand holds great potential for entertainment and education. For example, modern computer games are becoming more immersive, containing multiple story lines and hundreds of characters. This has substantially increased the amount of work required to produce each game. However, by allowing the game to write its own story line, it can remain engaging to the player whilst shifting the burden of writing away from the game's developers. In education, intelligent tutoring systems can potentially provide students with instant feedback and suggestions of how to write their own stories. Although several approaches have been introduced in the past (e.g., story grammars, story schema and autonomous agents), they all rely heavily on handwritten resources. Which places severe limitations on its scalability and usage.

In this thesis we will motivate a new approach to story generation which takes its inspiration from recent research in Natural Language Generation. Whose result is an interactive data-driven system for the generation of children's stories. One of the key features of this system is that it is end-to-end, realising the various components of the generation pipeline stochastically. Knowledge relating to the generation and planning of stories is leveraged automatically from corpora and reformulated into new stories to be presented to the user.

We will also show that story generation can be viewed as a search task, operating over a large number of stories that can be generated from knowledge inherent in a corpus. Using trainable scoring functions, our system can search the story space using different document level criteria. In this thesis we focus on two of these, namely, coherence and interest. We will also present two major paradigms for generation through search, (a) generate and rank, and (b) genetic algorithms. We show the effects on perceived story interest, fluency and coherence that result from these approaches. In addition, we show how the explicit use of plots induced from the corpus can be used to guide the generation process, providing a heuristically motivated starting point for story search.

We motivate extensions to the system and show that additional modules can be used to improve the quality of the generated stories and overall scalability. Finally we highlight the current strengths and limitations of our approach and discuss possible future approaches to this field of research.

Acknowledgements

I would like to start by thanking my supervisor Mirella Lapata who has guided me throughout my research. For her continued patience and support, foresight and encouragement. Mirella has always made time to listen to my questions and ideas (good or bad), allowing me to benefit from her incredible knowledge and experience. I also owe her a great debt for helping me to keep my focus and sense of perspective over the years.

I am grateful to Jon Oberlander, Christopher Mellish, Helen Pain, Johanna Moore and Diane Litman for their insightful suggestions and comments as my thesis reviewers. Also, to all my colleges in the department, for always being available to throw around ideas, trade pointers and for providing an environment of camaraderie. The time I have spent at the University of Edinburgh has been a deeply enjoyable experience and I will fondly remember our time together.

And of course, to my friends and family for their unconditional support. Your patience, sympathy and faith are more than I could have asked for. Thank you for listening, for your wonderful distractions, your care and your love.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Neil McIntyre*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The process of generating stories using computers has a longstanding tradition in the field of artificial intelligence. Computational storytelling has the potential to bring benefits to both eduction and entertainment, providing a wealth of new example stories and allowing the on-line creation of interactive narratives. To this end, many different approaches have been proposed in order to solve the problem of how best to model and implement a story generation system. Current approaches, however, require a large volume of hand-crafted rules and knowledge bases, placing the burden of creating them on the system developer. In this thesis we take the first step towards constructing a story generator that consists of trainable components, and is thus learnable from text.

The ability to generate stories automatically is most relevant in education where stories are used to help teach children. Currently, however, story creation relies on human authors and the writing process demands considerable effort and time to ensure stories are both informative and entertaining. Teaching resources are therefore limited to the stories that are in circulation. Computational story writing has the potential to greatly increase the number of stories available by removing the reliance on human authors. Also, current research of interactive tutoring systems has highlighted the ability to provide students with feedback, specifically tailored to meet their needs based on automatic assessment of their work. The ability to analyse a story in development, offering suggestions and highlighting areas of concern would be invaluable to young authors. As a prototype for such a system, STORYSTATION (Halpin et al., 2004) is able to discern the difference between good and poor attempts by students at a story retelling task. Automatic storytelling also holds great potential as a component in interactive environments which can be used as learning aids. Robertson and Good (2003a) propose GHOSTWRITER, a system that allows children to interact in a story environ-

ment with the aim of collaboratively generating a narrative. Within this environment the students are encouraged to form relationships and make decisions pertaining to complex problems, with the overall aim of increasing their perception of story characterisation. Designing scenarios that the students encounter is currently the task of the developer, however, through automatic generation means, the scenarios used in interactive tutoring systems could be automatically tailored to specific age ranges, subjects and learning outcomes.

As technological media becomes more pervasive in human culture, Murray (1997) foresees a greater role for digital storytelling in everyday life. The more we interact with the web, social networking and computer games, the more we expect to find narrative structures to make them more accessible and entertaining. Stories are already used extensively in the entertainment industry, for instance within computer games. Many computer games, such as Role Playing Games (RPGs), allow the gamer to assume the role of a character and participate in a story line throughout the game. As the performance and processing power of games consoles increase, so does the potential for larger and more sophisticated gaming environments. Recent games (e.g., *The Elder Scrolls IV: Oblivion* and *Fallout 3*) include multiple story lines, hundreds of NPCs (Non Player Characters), game locations and thousands of lines of dialogue, resulting in games that involve hundreds of hours of game-play. Clearly, games of this magnitude require substantial amounts of work in developing. Also, the majority of these games will have their plots set during development and as a result each consecutive play of the game will be less interesting. However, if a game were to generate its story during runtime then the gamer would feel much more involved (i.e, less like they are acting out a script). Having a dynamic plot that responds to the actions of the gamer would make game play more involving and entertaining. As more players are signing up to Massively Multiplayer Online Role Playing Games Games (MMORPGs, e.g., *World of Warcraft* and *Star Trek Online*) which contain many players interacting in online environments, the dynamics of game plots are becoming much richer. Computers can therefore be utilised in this area to create interesting stories and plots for games within these dynamic environments. In addition to creating richer computer games, there is also the potential for automatic story generation to be integrated into tools for authors. One example is in providing hypothetical plot lines for screen writers, suggesting possible character attributes and scenes by evaluating the developing plot.

The art of storytelling is one that humans often take for granted but it forms an

integral part of our cultural heritage. We encounter stories in a large number of everyday situations. We use them to communicate with one another, to create examples for teaching and even as a resource in shaping our self-image in the form of a self-narrative containing our memories (Mar, 2004). However, it can be difficult to pin down exactly the skills required in order to produce or understand a simple story. Take for example, the well known fable of "The Hare and the Tortoise" attributed to Æsop.

> A Hare jeered at a tortoise for the slowness of his pace. But he laughed and said, that he would run against her and beat her any day she would name. "Come on," said the Hare, "you shall soon see what my feet are made of." So it was agreed that they should start at once. The Tortoise went off jogging along, without a moment's stopping, at his usual steady pace. The Hare, treating the whole matter very lightly, said she would first take a little nap, and that she would soon overtake the Tortoise. Meanwhile the Tortoise plodded on, and the Hare oversleeping herself, arrived at the goal, only to see the Tortoise had got in before her.
>
> Slow and steady wins the race.

The simplistic nature of this text classifies it as a children's story, but we must consider that reading this story involves complex thought processes. Besides language processing, this story requires us to make inferences based on our word knowledge, which informs us that hares are fast and tortoises slow. We are required to correctly interpret the story *fabula* (the correct ordering of all events performed by story characters, including implied actions that have been omitted) from the story *sujet* (the ordering presented in the text). This often includes the ability to 'look between the lines', using inference, world knowledge and personal experience to fill in gaps in the narrative (Anderson et al., 1977). We also have to consider that the fable contains a moral message, and how the situation acted out by the story characters is applicable to our own lives by considering our own personal experiences. When reading or hearing a story, interpretation requires looking further than the level of the text itself, we must also consider who the intended reader is and what the author intended to communicate. As a result, this make the task of story understanding and generation very challenging for computers to do well.

Storytelling is a useful skill as it allows us to transform information into a form that is entertaining and easier to understand. To date there has been little consensus on how to generate stories automatically and previous work has emanated from many different disciplines. For example, the Russian formalist Propp (1968) analysed Russian folk tales and identified recurring units (e.g., the villain is defeated, the hero is pursued)

and rules (e.g., an act of villainy will often be followed by meditation) which indicated how they could interact. Consider the following formula:

$$S \rightarrow ABC \uparrow DEFG \frac{HJIK \downarrow Pr - Rs^{0}L}{LMJNK \downarrow Pr - Rs} QExTUW$$

This is in fact a schema produced by Propp that represents a range of different Russian folk tales that he studied. In this schema, the individual propositions represent those morphemes that he identified as well as their ordering (e.g., W stands for a wedding, so from the schema above we see that many Russian folk tales end with the main characters getting married). Using Propp's rules, new Russian folk tales can be generated from the reformulation of the available morphemes. Structural Grammars, such as those used in language processing, have been used to both describe and create stories. For instance, Klein et al. (1979) used a grammar in order to generate stories of the murder mystery genre. Recent approaches to computational storytelling have focused on the use of autonomous agents (Swartjes and Theune, 2008). By supplying a character agent with a set of goals, a story can be created by recording the actions it performs in order to resolve these goals.

There are several common challenges that any hypothetical story generator would have to face. Perhaps the biggest concern is the amount of world knowledge required to create good stories. For this reason, previous systems have been primarily handcrafted. The amount of knowledge required varies but most approaches generally require information on the entities appearing in the story, how they interact and how a story can be structured to include them. For example, story generation with agents requires a set of actors with goals and action descriptions, an environment in which the agents will interact, and a description of possible relationships between the agents. All of this information must be complete to ensure that the planning algorithms will create successful plans. Using handcrafted data sets a limit on the domains that the story generator can cover. In agent-based systems, a whole new set of action descriptions and environments would have to be written in order to generate stories of a different genre. It is easy to imagine that the process of creating the required knowledge base for these systems will be laborious and time consuming which removes any impetus to expand them to cover new genres.

1.1 Thesis Hypothesis

Many of the components of a story, such as characters and themes, will repeat in similar forms in other stories. We can see from the work of Propp (1968) that several stories contained a Hero or a Villain and although they may not be have been identical in every story, they shared a lot of similar properties. It is precisely this redundancy that we wish to exploit in order to create new stories, by discovering regularities in a large sample of manually authored ones. Our goal is to generate stories automatically by leveraging knowledge inherent in corpora. Rather than hand-coding all the information in the system for each genre, we want the system to acquire this information from stories it has already seen. This will allow the us to port our story generator to different genres without extensive manual hand-coding.

There are many different story genres for which we could generate stories. Examples include fairy tales, parables, fables or even murder mysteries. However, we wish to start simple in order to see if our hypothesis is at all feasible. For this reason we have chosen to focus on children's stories. Children's stories offer several benefits as the basis of a feasibility study. To start with, the vocabulary and the content of each story will be limited as they must meet the knowledge held by children. Secondly, they have great potential for educational applications, such as educational interactive systems. These systems usually have an introductory story that the children seek to develop. We believe that if such a system could generate stories itself then it could provide children with much richer examples as they attempt to write their own.

1.2 Contributions

The main contributions of this thesis are as follows:

- We describe an end-to-end story generation system that is comprised of both trainable and off-the-shelf components. We outline an approach to story generation that removes the reliance on hand-crafted data sources, which to date places a large burden on the system developer. As research into trainable components for natural language generation (NLG) continues to rise, systems that fall under the purview of NLG should seek to incorporate them. We describe, in this thesis, trainable components for content selection, sentence planning, document planning and surface realisation and show how they be combined to form a NLG system capable of producing short children's narratives. Unlike the majority of

research which focuses on individual components, we propose an end-to-end system realising the whole NLG pipeline. The evaluation methods developed in the thesis make it possible, for the first time, to establish the state of the art in the field. And the thesis (therefore) delivers state of the art performance, a suitable baseline against which to compare future developments.

- We present a viable bottom-up model of story generation, without rhetorical structures and other high level document structures. We view the task of generating a story as a search problem that simultaneously seeks the best content and structure of the document it is producing. Our approach is data lean and starts by first building the best sentences and then through their combination, the best stories.

- We demonstrate the utility of trainable models for interest and local coherence that can be used to evaluate the quality of stories in the story search space. These models are trained using shallow document features, removing the need for hand annotation, making them desirable for automatic systems which are required to evaluate large numbers of possible texts.

- We compare several methodologies for searching the space of possible stories. First, we outline a generate-and-rank approach in which stories are created one sentence at a time, with the subsequent stories being ranked and the best retained. We also present a genetic algorithm for optimising stories generated from document plans. Interesting stories are sought though the repeated application of mutation and crossover operators, which allows for exploration of the story space.

- This thesis reports the first work to induce plots from corpora and generate novel stories from these. Specifically, we extract from the corpus, knowledge about the expected action progressions and interactions for each entity. We represent this knowledge in the form of directed graphs, in which nodes represent actions and transitions indicate ordering. From these graphs we can then generate plots, which are essentially schema, each encoding a large number of possible stories. The stories that we generate from plots represent advantageous areas of the search space which we use as the starting point for our genetic algorithm search.

- The final contribution is to demonstrate the thesis work provides a platform for future work. The extensions are not carried out to show improved text qual-

ity, but to show ease of extensibility, scalability and portability. Specifically, we introduce components for generating referring expressions and integrating commonsense knowledge, which rely on data mined from corpora or publicly available knowledge bases. We also investigate the portability of the system, by evaluating its ability to retrain on new corpora and to fulfil requirements for a new task, namely, finishing incomplete stories.

1.3 Thesis Overview

The remainder of the thesis is structured as follows:

- Chapter 2 starts with a review of current approaches to computational story generation. To date a wide variety of systems have been developed that differ in their methodology and the data sources that they utilise. In particular, we focus on those systems that generate stories using problem solving, autonomous agents, commonsense knowledge, story grammars and story schemata. The second half of the chapter then explores current work in natural language generation (NLG), specifically those approaches that create trainable components in order to partially implement the NLG pipeline as described in Reiter and Dale (2000), namely, content selection, sentence planning, document planning and surface realisation. We then conclude by arguing that a story generation system should ideally draw from both of these fields.

- Chapter 3 introduces our story generation system, designed to create short stories targeted at young children. This is an end-to-end system comprised of trainable components. We describe each of our components for content selection, sentence planning and surface realisation in detail. We formulate the story generation task as a search problem and describe a generate-and-rank methodology for selecting the best sentences and stories. Shallow document features are utilised to train models for evaluating stories. We outline a model of interest, trained on ratings of Æsop's fables elicited from human subjects. We also motivate the use of a an entity based model of local coherence (Barzilay and Lapata, 2008). Finally, through a human evaluation study, the generate-and-rank system was evaluated by comparing it to two simple baselines.

- Chapter 4 looks to improve upon the stories generated in Chapter 3 by introducing an approach for document planning, namely the use of story plots. We

present a graph formulation for encoding the action progression of an individual story protagonist across a collection of texts. By combining the graphs of two protagonists, story plots are created from the regions in which they interact. We show that the plot graphs have an impact on the stories the system is capable of creating, specifically with respect to content selection.

- Chapter 5 motivates the use of evolutionary search algorithms, akin to Darwin's 'survival of the fittest', to improve on the limitations of the generate-and-rank approach described in Chapter 3. Specifically, we present and motivate a genetic algorithm implementation that optimises stories generated from plots. GAs are well suited to this task as they allow the system to creatively search a larger section of the story search space. We formulate operators for crossover (recombination) and mutation specific to our story generation task and outline a series of possible fitness functions based on local coherence. We conclude by comparing the GA-based system with the generate-and-rank system as well as two baselines, through a human evaluation study.

- Chapter 6 provides a platform for exploring the extensibility and portability of the basic story generation system by motivating additional components and tasks. Firstly, we propose a trainable component for generating referring expressions to address observed deficiencies in the generated narratives. This component would supply definite descriptions and pronoun references for inclusion inclusion in the surface text. We then introduce a component that expands upon the generated stories using a database of commonsense knowledge facts (Singh, 2002). These facts are included to explain the motivation for and effects of story actions, along with providing details about story entities. The second half of the chapter is devoted to the exploration of the system's portability. We explore the ability of the system to generate stories from a new corpus, representing a new domain. We also explore the capability of the system to perform a new task, namely, its ability to complete partial or unfinished stories.

- Chapter 7 concludes the thesis with a summary of the main contributions and findings, and highlights areas of potential future research.

1.4 Published Work

The thesis expands on material that has been previously published. Chapter 3 gives a more detailed account of the system and human elicitation studies presented in McIntyre and Lapata (2009). Further information is given on preliminary studies of the system and the evaluation models that were trained.

Chapter 4 and Chapter 5 expand on research detailed in McIntyre and Lapata (2010). In particular, Chapter 4 provides a detailed explanation of how our story plots are elicited and explores their capacity for generating stories. Chapter 5 presents further information on our genetic algorithm formalism, including a preliminary parameter study.

Chapter 2

Related Work

There have been many different approaches to the problem of automatic story generation. These differ in the data sources they employ and methods they use when generating their stories. In the first half of this chapter we give an overview of these approaches and the systems that have been constructed to date. These approaches to story generation can be grouped into the following: problem solving, agent-based systems, story grammars, story schema and those using commonsense knowledge. Our primary focus is to examine the scalability of these approaches, in particular how easy it is to transfer from one domain to another.

The second half of this chapter focusses on recent work in the field of natural language generation (NLG). Our aim in this thesis is to produce textual stories, therefore we need to examine the methods in which text documents can be generated from knowledge bases. Our focus will be on those NLG systems that have been trained on data, relieving the reliance on hand-crafted knowledge bases.

We will end the chapter by proposing that the story generator we seek to create must lie in the overlap of these two fields.

2.1 Story Generation

Story generation has a long-standing tradition in the field of Artificial Intelligence and many different approaches have been proposed in order to solve the task. Viewing story generation as a computational task requires several decisions to be made: on how to structure story knowledge, whether or not to incorporate human cognitive processes, whether generation is solely the task of the system or if humans can be used as collaborative authors, and how best to represent the space of possible stories the system

```
(define (action shoot)
   :parameters (?attacker ?victim ?weapon ?place)
   :precondition ((character ?attacker) (character ?victim)
                           (weapon ?weapon) (location ?place)
                           (at ?attacker ?place)
                           (at ?victim ?place)
                           (has ?attacker ?weapon)
                           (violent ?attacker))
   :effect ((not (alive ?victim))))
```

Figure 2.1: Example of a planning rule for the action *shoot* from Riedl and Young (2006a).

is capable of generating. Below we will outline some of the approaches to story generation that have been used to date and see how they respond to these questions, in structure and procedure.

2.1.1 Problem Solving

TALE-SPIN (Meehan, 1977) is one of the earliest computer storytellers. It's approach to story generation is to supply story characters with goals and use the resulting search traces as the basis of the story output. The stories generated were about animal characters in a style similar to Æsop's fables. Each story starts with a character being supplied a set of goals, such as *John Bear is hungry*. Planning rules are then searched to find actions that can be applied to the current state of the story world in order for a goal state to be achieved, in this example, *John bear is not hungry*. An example of a generic planning rule is shown in Figure 2.1. TALE-SPIN requires knowledge about each of the characters in the story, including their personalities and relationships to one another, as well as the story environment, i.e, how locations in the story are related so characters can move from one to another. Most importantly, it requires a knowledge base of the actions that are applicable to the story domain, each of which require the prerequisite state for an action to be applicable and the postconditions that show how the resulting environment will have changed. The completed stories contain both statements of fact about the environment and characters, i.e., "there was water in the river" and "Wilma bird lived in a nest", and also the actions that appear in the problem solver's trace, "Wilma flew from her nest across a meadow through a valley to the river".

The Vengeful Princess

Once upon a time there was a Lady of the Court named Jennifer. Jennifer loved a knight named Grunfeld. Grunfeld loved Jennifer.

Jennifer wanted revenge on a lady of the court named Darlene because she had the berries which she picked in the woods and Jennifer wanted to have the berries. Jennifer wanted to scare Darlene. Jennifer wanted a dragon to move towards Darlene so that Darlene believed it would eat her. Jennifer wanted to appear to be a dragon so that dragon would move towards Darlene. Jennifer drank a magic potion. Jennifer transformed into a dragon. A dragon moved towards Darlene. A dragon was near Darlene.

Grunfeld wanted to impress the king. Grunfeld wanted to move towards the woods so that he could fight a dragon. Grunfeld moved towards the woods. Grunfeld was near the woods. Grunfeld fought a dragon. The dragon died. The dragon was Jennifer. Jennifer wanted to live. Jennifer tried to drink a magic potion but failed. Grunfeld was filled with grief.

Jennifer was buried in the woods. Grunfeld became a hermit.

MORAL: Deception is a weapon difficult to aim.

Figure 2.2: Example of a story produced by MINSTREL, taken from Turner (1992).

When generating stories, TALE-SPIN focuses mainly on satisfying the goals of the story characters, but as Turner (1992) explains, this often leads to stories that lack purpose as stories are developed without an overall message. Turner's system MINSTREL, on the other hand, generates stories using a combination of character and authorial goals. These author goals drive the overall search procedure of the system so that although the goals of the story characters are achieved, the events that appear in the story conform to an overall theme which is the point or message behind the story. For example, one theme used by MINSTREL represents the moral "Done in haste is done forever", which results in a story where an impulsive action performed by a character ultimately leads to a regrettable outcome.

MINSTREL is capable of generating stories about King Arthur and the Knight of the Round Table. An example of a story generated by MINSTREL is presented in Figure 2.2. Once again, generation is a problem solving task. Interestingly, Turner paid

particular attention to what it would mean for these stories to be considered creative. The approach taken is to model MINSTREL's problem solving technique on human episodic memory. The system utilises case-base reasoning, where previous solutions to problems are stored for reuse. But in order to encourage creativity, the system remembers when solutions have been used previously so as not to simply generate the same story over and over again. Novel stories are generated through the application of Transform-Recall-Adapt Methods (TRAMs). When the system encounters a problem one of the characters is striving to overcome, it can either directly call up a solution to that problem if it has been encountered before or attempt to modify a similar solution or sequence of solutions that would bring about the same result. A TRAM works by modifying the current problem situation until it matches one in the database; for example, *'the knight accidentally meets a princess'* requires the system to create a scene for which it has no previous knowledge. However, the relevant knowledge can be found in a different domain, specifically by transforming the problem to *'a businessman accidentally meets a person'*, for which the system already has a solution. In this manner, creativity appears in the generated stories through the system's ability to adapt previous solutions to new problems, even across domains. Recently, Tearse et al. (2010) have recreated MINSTREL in their system MINSTRELREMIXED, and propose making the system interactive allowing human users to collaborate in the generation process.

The aim of UNIVERSE (Lebowitz, 1985) is to create ongoing plot lines for soap operas. This system generates story plots through the use of planning, but uses a database of 'plot fragments' from which to construct a story. These fragments cover a range of levels, from thematic plans that cover larger time scales, to specific plot actions. Each fragment then outlines a series of goals and sub-goals that are expected to appear in the story. Rather than making them character specific, the plot fragments are generalised. These fragments can then be selected in order to create authorial level goals for the generated stories, similar to those in Turner (1992), however, it should be noted that the task is quite different as the plot lines generated by UNIVERSE are not intended to have an ending. Some of the author level rules used by the system reflect this constraint, such as the author goal KEEP-STORY-MOVING, which forces characters to perform actions even when they are not optimal for meeting their character level goals. Although the development of story actions are the result of author level direction, UNIVERSE requires extensive information about the characters involved. This includes encoding information about a character's level of wealth, intelligence and promiscuity; and their relationships to other characters in the system

such as current and previous spouses. The system does however allow characters to be grouped together into stereotypes, such as *socialite* and *doctor*, and according to their past event histories, e.g., *divorce* and *illness*, which allow characters to inherit default traits. Lebowitz (1983) outlines an approach for generating a cast of characters by iteratively selecting under-defined characters and hypothesising their personal histories from the system's plot fragments.

Another approach is that of MEXICA (Pérez y Pérez and Sharples, 2001), a system that generates stories about the Mexicas (the people that once lived in what is now Mexico city). This system relies on the user to build the knowledge base from which the stories can be created, first by describing the actions that can take place and then providing a set of example stories (the stories are defined as sequences of actions). Using the previous stories as a basis, MEXICA then learns the rhetorical knowledge it will use to generate new stories. The system attempts to explore the cognitive process of writing by modelling generation using an engagement-reflection cycle. During the engagement cycle, a selected story action is applied (the first story action is supplied by the user) and each story character's attributes and relationships are updated (information relevant to the story characters are stored in structures called Story World Contexts (SWC)). For example, the *knight* having been healed of his injuries by the *princess* may now feel indebted towards her. Rather than looking specifically at the goals of the story characters, the system uses the character's SWCs to find those actions from the previous stories (considered the system's long term memory (LTM)) that could follow from the previous action. One of these possible follow-on actions is then selected at random and the process continues. The second phase is known as the reflection phase, during which the system evaluates each story for coherence, interest and novelty. The coherence of each story is assessed by ensuring the preconditions of each action are satisfied in the story world. Interest is assessed by looking at the expected audience tension that is associated with the actions appearing in the story. Finally, the novelty of a story is decided based on the frequency that actions appear in the LTM, with infrequent actions being deemed the most novel. Also, the system can overcome impasses where SWCs do not relate to actions in LTM simply by selecting actions known to advance the previous action in the example stories. A story is then generated through subsequent applications of engagement and reflection. Each has the ability to modify the existing story, for example, the reflection cycle may highlight problems with the current story and identify where the changes need to be made in the engagement cycle. Stories are therefore not created simply as a sequence of actions,

but can be edited throughout the generation process.

An approach for story planning similar to Turner (1992) and Lebowitz (1985), is the *vignette* described in Riedl and Leon (2008) and Riedl (2008). A vignette is a fragment of a story that is considered a 'good' representation of a situation and/or context that has been found to commonly occur in stories. These may include bank robberies, betrayals, combat situations, ect. These are essentially an ordered set of events, perceived to be important to that specific narrative situation. Each vignette represents a fragment of a narrative plan that is general enough to be employed in different contexts depending on the characters and objects assigned. Riedl (2008) outlines the Vignette-Based Order Causal Link (VP-POCL) planner that allows flaws in incomplete story plans to be corrected by incorporating partial plans stored in the vignette library. Upon finding a vignette that contains an action that satisfies the incomplete story plan, the planner can then integrate the stored partial plan into the current story plan. Unlike the episodic memory approach used in Turner (1992), these plans are not considered new solutions to a particular problem and are therefore not retained for future use. The vignettes can, however, have an impact on the creativity of the story as they represent fragments of good narrative plans rather than efficient problem solving solutions, allowing the planner to incorporate events which enhance the narrative even though the planner would have considered them superfluous.

Peinado et al. (2004) and Gervás et al. (2004) both use a formal structure for generating stories in a compositional manner based on the findings of Propp (1968). Propp, a Russian formalist, studied a selection of Russian folk tales and broke them down into recurring units or morphemes. He noted that although the names and some of the attributes of the characters change, the actions primarily remain the same and he defined these actions as *functions*. For instance some of the functions are "The Hero leaves home" and "The Villain is defeated". He then created rules explaining how these functions interact with one another. Despite being rather specific, Propp's morphemes can be used to generate many Russian style folk tales. Peinado et al.'s (2004) system, PROTOPROPP, uses a database describing the Propp functions. The user inputs a query and the system uses it to reason about characters and the order of the Propp functions that best create a story for that query. For example, the user may request a story about a *prince* that includes a "Test of the Hero" and a "Wedding" (both "Test of the hero" and "Wedding" being Propp functions). Using the dependencies that have been identified in the database ontology, the system generates a story allowing for possible temporal restraints and dependencies between actors within functions. For example, if the hero

was to be kidnapped during the story, then there would also be a dependency on him being released before he could attend his wedding. As a problem solver, the system employs case-based reasoning in order to identify previous story structures from Propp that can be moulded to fit the user's story requirements.

A major drawback of these systems is that the structural formality of the rules they use for planning and search limits their scalability and robustness. For instance, PROTOPROPP (Peinado et al., 2004) is based solely on Propp's rules. This means it can only generate stories in the genre of the Russian folk tale and only specific types of events, or functions, can appear in the story, namely those identified by Propp. Similarly, MINSTREL (Turner, 1994) can only generate stories about King Arthur and the Knights of the Round Table. For a planner to work effectively it is vital that each of the planning rules clearly state all of the relevant preconditions and postconditions of the story environment. Figure 2.1 shows an example of a single rule. As we can see there are many conditions that must first be met before the action can be applied. A human developer is required to write these rules based on their knowledge of the real world and the domain at hand. As the complexity of the environment and the actions therein increases so does the number and complexity of rules required, placing a heavy burden on the developer.

Although these systems are all capable of planning stories they vary in terms of the linguistic sophistication of the texts they generate. UNIVERSE and vignettes (Riedl, 2008) are focussed on solely creating story plans. TALE-SPIN, MINSTREL, MEXICA and PROTOPROPP, all use either canned text that is associated with story actions or map those actions to sentence level schema from which they create stories. MINSTREL incorporates some linguistic knowledge as is evident by the use of pronominalisation. Gervás et al. (2004) outline an extension to PROTOPROPP that would allow it to generate more natural texts and break the its reliance on sentence templates.

2.1.2 Autonomous Agents

Agent-based Story Generation is currently the most popular research area in the story generation community. This approach uses autonomous story agents that have been embedded within a story world and creates stories based on the actions they perform. Agents are described in Russell and Norvig (2003) as being able to interact with the environment around them through sensors and effectors, as shown in Figure 2.3. The figure illustrates a generic agent that can use sensors to perceive the environment and

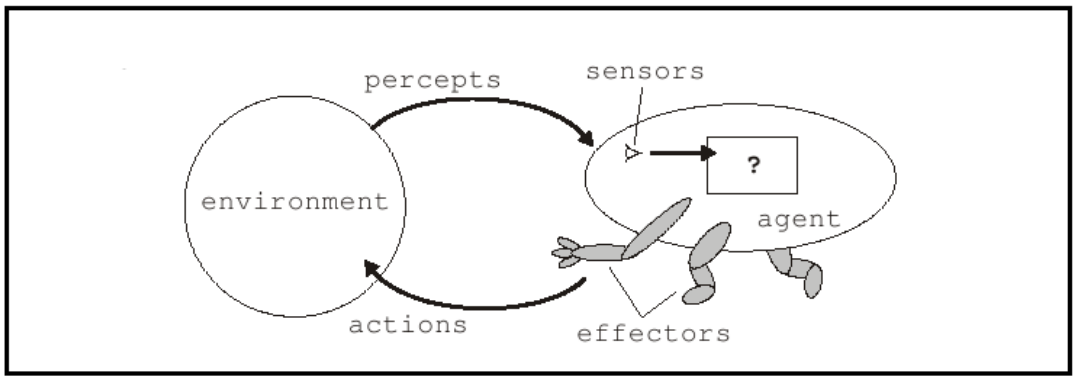


Figure 2.3: A generic agent interacting with an environment. The agent's sensors allow it to view the current state of the environment. An internal mechanism within the agent will take this information and use it to decide which action to make next. The agent can manipulate the environment through the use of its effectors. Agents can appear in many different types of environment and in different forms, for example, robots in the real world or web crawlers on the Internet. The example is taken from Russell and Norvig (2003).

then decides how to use its effectors in order to manipulate that environment based on the instruction of an internal reasoning mechanism. In a story environment, the agents act as characters with the ability to sense everything within the environment and perform actions in order to manipulate it. The environment itself may contain a collection of character agents as well as story objects which are used as props within the story. An example story environment is shown in Figure 2.4. Each agent starts with a list of goals and by using planning techniques, similar to that of Meehan (1977), forms plans of action and tries to fulfil them. In contrast though, the planning is distributed amongst the agents. In Figure 2.4, the agent representing a prince wishes to wear the crown and to do this it has to make a plan of action based on its knowledge of the environment. In this example the agent chooses to first enter the castle and then put on the crown. Performing each action will effect the state of the environment leading to a state in which the crown is on the agent's head. The actions performed by the agents as they interact with their story environment, as well as with one another, will become the outline for the story. Interesting stories occur as the plans of several different agents interact and cause failures and possible re-planning.

The major drawback of using agent-based systems is that they require the entire story world environment and all actions that can be performed within it to be explicitly

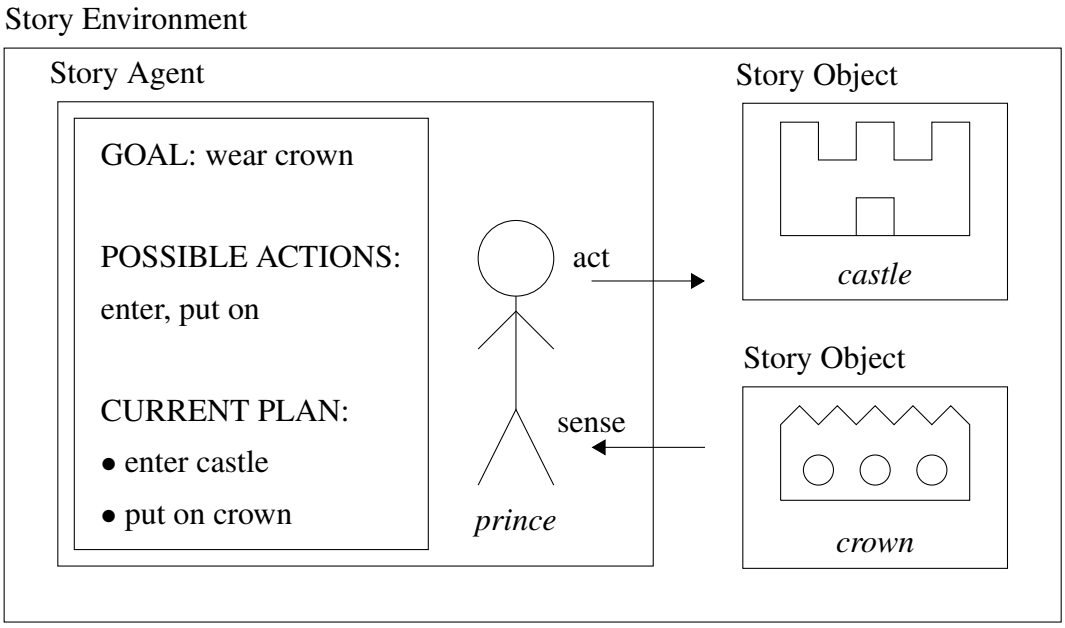


Figure 2.4: Example of a story environment with one actor agent and two story objects. In this example the story agent has constructed a simple two action plan in order to reach its goal of wearing the crown. More sophisticated systems would have multiple story agents, capable of many more actions and the possibility for several goals.

programmed into the system. This leaves the story world with finite possible actions and plans for each agent and a finite set of story props with which they can interact. The Virtual Storyteller project of Fass (2002); Theune et al. (2003); Swartjes and Theune (2008) is an example of such a system that uses many actor agents, performing character level goals, under the guidance of a narrator agent, performing author level goals, to construct fairy tales. These stories are then presented to the user of the system using a voice synthesis speaking presentation agent.

In Oinonen et al. (2006) a framework is proposed to allow the Virtual Storyteller system to learn from human authored or machine generated texts is given. Here, data related to different levels of story representation are extracted from a collection of texts. The first level is story knowledge, in which they extract the current state of characters and objects in the story environment with respect to time. The second level is the character level where information about each character in the stories, including their motivations, emotional state, relationships with other characters, personalities, and beliefs are collected. The next level is to acquire information from the texts about the plot by building up a database of story fragments as causal networks or graph

models of actions and events. This allows them to view the temporal relationships between actions within a story. The top levels of story representation require gauging the effect of the text upon the reader. For this the they require feedback annotations from readers to discover how different narratology concepts are constructed and how different story construction techniques (e.g., adding suspense) affect the reader. There is to date, however, no evidence that their framework has been indeed implemented.

Shim and Kim (2002) also use autonomous character agents to generate short stories. They break the task into a set of multi-level goals that have to be achieved in order to create a story; viewer goals (emotional state of the reader), plot goals and character goals. The agents within these stories have a limited sense of emotion and as different actions are performed on or near them, their internal states are changed depending on the rules associated with these actions. This supplies the story with an extra level of information for the viewer but also assumes that the effects of each action are explicitly entered during construction of the story world. The user of the system starts by supplying it with a desired emotional state that the story should express. The story is then generated as a series of episodes in which the characters select and perform actions that will either maintain the current plot emotion or seek to change it in order to match the viewer specified emotion. The resulting story is generated as a comic strip with an image depicting each episode.

Research into the use of autonomous agents has yielded more sophisticated agents that can be used as the basis for story generation systems. One of the main aims in this field of research is to make the agents more lifelike. To this end, models of emotional response have been developed. Theune et al. (2004) describe how the agents in the Virtual Storyteller can record their current emotional state (following the model of Ortony, Collins and Clore (OCC, Ortony et al., 1988))[1]. Different events, such as those near the completion of the agent's goals, will affect their emotional state in a way that is outlined by the character's personality. The agents can now form emotional bonds between one another, e.g., fear and love, and can be emotionally affected by the events performed in the story, e.g., they can become sad if a character they liked is killed. Agent's emotional states can be also used to trigger certain actions, such as singing when they are happy, or give weighting to possible actions and goals, e.g., Diana's reluctance to carry out her goal of killing Brutus because she fears him. Loyall and Bates (1997) also outline how agents can incorporate language in order to transfer beliefs and facts from one agent to another. This would allow an agent-based system

[1] See also Bartneck (2002) and Chen et al. (2001).

to simulate speech acts, furthering plot developments in which a character must first gain information from another before they can accomplish their goals.

Another reason that this area of research is growing in popularity is that it allows the user to take on the role of one of the agents and therefore interact in the story being created. For instance, Young and Riedl (2003) use Unreal Tournament to create a 3D interactive narrative world populated with story agents. The user then assumes the role of one of the characters in this world and, through their actions, affect the story being constructed. This allows the user to be a key part of an unfolding story which helps increase their enjoyment and leads to novel stories.

Recent years have seen growing interest in online multiplayer communities where many human players can interact when playing the same game. Fairclough and Cunningham (2003) describe a story management system which can develop stories in a multi-player environment. Their system generates stories using case-based reasoning over Propp Functions. The story actions, characters, objects and locations are those described in Propp (1968). Each human player selects a character to play, and a story director agent is created in order to generate stories for them specifically. There is also a main story director agent, whose job is to update the character agents, referred to as non-player characters (NPCs), depending on their interactions with the human players. Character agents are also capable of communicating with one another, creating speech acts which are presented to the players on screen using canned text. Being able to construct and maintain multiple storylines is clearly the biggest task of such a system. In order to properly engage the human player, there must be a suitable number of possible actions for them to perform so as to maintain interest in the developing story. Combined with this is the need for realistic NPCs that act similar to other human players, performing their own goals whilst also supporting the developing storyline of the player. The lucrative nature of the computer games industry ensures that this will remain a hot topic for research.

Interestingly, we see a difference of focus in the presentation goals for the agent-based systems when we compare them with the problem solving systems in the previous section. Virtual Storyteller, although capable of producing a textual representation of the generated stories, uses speech synthesis technology incorporated with a graphical agent to present their fairy tales. Shim and Kim (2002) present their stories as a comic strip in which the story environment, character agents and objects are associated with images. Obviously, story generation within an interactive game environment requires the stories to be acted out rather than transcribed, although text generation

may be required in order to inform human players of scene specific information and generate dialogue.

2.1.3 Commonsense Knowledge

Liu and Singh (2002) propose an approach to story generation that relies on commonsense knowledge. Using causal chains, the intention of a particular character's actions can be followed throughout the story. This work is based on commonsense information, supplied by the Open Mind Commonsense Knowledge Base (OMCS[2], Singh, 2002). The OMCS contains 400,000 semi-structured English sentence of commonsense relationships that have been obtained from contributors on the web. This is information such as, an *eye* is a *body part* and that the effect of *riding a bike* can be an *accident*. Their system, MAKEBELIEVE, starts by taking an input sentence from the user and then uses this as the initial sentence to create a story of length 5 to 20 sentences. It works by finding likely consequences to the actions that appear in the sentence, by comparing the similarity of the words appearing in the current sentence to all the possible following consequences in the commonsense database. For example, given the sentence "Mary broke her leg" we could expect "Mary went to hospital" to be the following sentence. This approach to story generation allows for realistic stories to be produced, although it is also very dependent on the information contained within the commonsense database. Each entry in the commonsense database is associated with a sentence, into which the current character of the story can be substituted to produce the final text.

2.1.4 Story Grammars

One of the oldest approaches to story generation is based on story grammars. These grammars consist of a set of expandable rules that allow the creation of different stories depending on the expansions used. Story grammars involve several levels of granularity, each being expanded by transformation rules until the terminals of the stories are reached. The terminals contain the text that will form the story. For some of these terminals there may be variables that are related across the entire story, such as the names of the story entities. These are satisfied once the entire story tree has been created. Arguably, the oldest proposed story grammar was developed by Lakoff (1972),

[2]See `http://agents.media.mit.edu/projects/commonsense/`

who used rewrite rules to create a grammar of Russian folk tales based on Propp's findings. The grammar consists of Propp functions which act as non-terminals. Thorndyke (1977) proposed his own story grammar with the intention of identifying the structural elements common to a set of narrative discourses. This grammar can be seen in Figure 2.5 and an example of a story derivation is shown in Figure 2.6. As we can see, a story is described by the expansion of the grammar rules with the terminals of the tree representing text, such as *EVENT* or *STATE* in Figure 2.5.

An overview of work based on story grammars can be found in Black and Wilensky (1979). They also list many of the problems that face the creation of fully expressive story grammars. For example, the grammars focus on the structure of the stories they create rather than ensuring that their content is cognitively coherent. It is important for a reader to understand the intentions of a character as they perform an action. Black and Wilensky analysed several grammars and found that they missed many of the stories they should have been able to generate and also generated stories that did not make sense. They argue that grammars are also inadequate to capture stylistic literary devices common to many narratives, such as switching between sub-plots and foreshadowing. Further information on the story grammar polemic can be found in Andersen and Slator (1990).

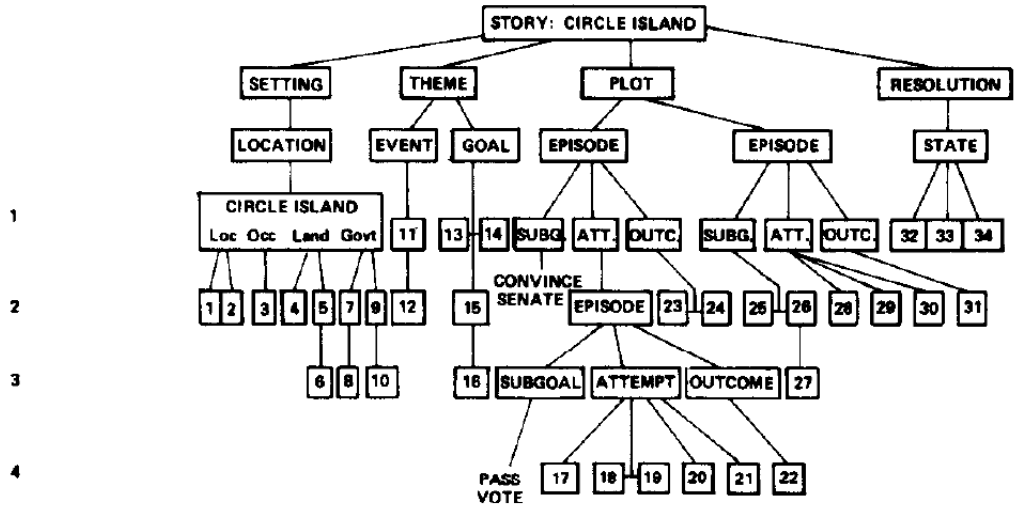
Despite the controversy, several systems have been developed that rely on a story grammar in order to generate stories. Correira (1980) present two systems, TELLTALE and BUILDTALE, the former describing a story generator and the latter a parser capable of analysing the generated stories. Both systems use the same inference engine, based on extended horn clauses (EHC, similar to planning rules described in Section 2.1.1 as they contain preconditions and postconditions defining their use). TELLTALE generates a story by expanding on rules held within a database. The database itself was manually constructed to produce short fairy tales. An example of a rule is, that each fairy tale must contain a setting and at least one episode, terminating with the main characters living happily ever after. Rules in the database are parametrised so that search and inference can take place to find applicable rules that will result in a story. The final story consists of a set of propositions, which are not translated into prose, each of which relates to a story action.

GESTER (Pemberton, 1989) uses a narrative grammar that generate stories in the genre of the medieval French epic. The grammar is hand-coded from the analysis of nine French epic poems. The terminals of a tree generated from this grammar depict what they call, the *narrative motifs*, which indicate the chronological flow of the text.

Rule Number			Rule
(1)	STORY	→	SETTING + THEME + PLOT + RESOLUTION
(2)	SETTING	→	CHARACTERS + LOCATION + TIME
(3)	THEME	→	(**EVENT**)* + GOAL
(4)	PLOT	→	EPISODE*
(5)	EPISODE	→	SUBGOAL + ATTEMPT* + OUTCOME
(6)	ATTEMPT	→	**EVENT*** / EPISODE
(7)	OUTCOME	→	**EVENT*** / **STATE**
(8)	RESOLUTION	→	**EVENT** / **STATE**
(9)	SUBGOAL / GOAL	→	**DESIRED STATE**
(10)	CHARACTERS / LOCATION / TIME	→	**STATE**

Figure 2.5: Thorndyke's Grammar (1977). Story rewrite rules are shown in (1)-(10). The right hand side consists of terminals or nonterminals. Terminals are indicated in bold face. The "+" symbol indicates a combination of elements in a sequential order and the "*" indicates that an element may be repeated.

However, when presenting the story to the user to read, the actions in the terminals may be reordered so that stories containing more than one storyline can be interwoven. Each terminal of the story plan is associated with a canned-text representation for inclusion in the final text. The rule expansions used will result in variables in the story tree, which can, however, require a large amount of inference to be satisfied. For example, Pemberton (1989) also maintains a set of rules describing the French epic genre, including rules about loyalty, inheritance, religious belief, marriage and military practice. This requires additional human annotation of the data set, decreasing the system's portability.



(1) Circle Island is located in the middle of the Atlantic Ocean, (2) north of Ronald Island. (3) The main occupations on the island are farming and ranching. (4) Circle Island has good soil, (5) but few rivers and (6) hence a shortage of water. (7) The island is run democratically. (8) All issues are decided by a majority vote of the islanders. (9) The governing body is a senate, (10) whose job is to carry out the will of the majority. (11) Recently, an island scientist discovered a cheap method (12) of converting salt water into fresh water. (13) As a result, the island farmers wanted (14) to build a canal across the island, (15) so that they could use water from the canal (16) to cultivate the island's central region. (17) Therefore, the farmers formed a procanal association (18) and persuaded a few senators (19) to join. (20) The procanal association brought the construction idea to a vote. (21) All the islanders voted. (22) The majority voted in favor of construction. (23) The senate, however, decided that (24) the farmers' proposed canal was ecologically unsound. (25) The senators agreed (26) to build a smaller canal (27) that was 2 feet wide and 1 foot deep. (28) After starting construction on the smaller canal, (29) the islanders discovered that (30) no water would flow into it. (31) Thus the project was abandoned. (32) The farmers were angry (33) because of the failure of the canal project. (34) Civil war appeared inevitable.

Figure 2.6: An example of a story and its parse tree using the grammar defined in Thorndyke (1977). Each numbered region of text maps to a terminal node of the parse tree.

2.1.5 Story Schema

Fayzullin et al. (2007) propose a system for generating stories from heterogeneous

data sources. These include text documents, web pages and XML documents to create story texts about archaeological artefacts in Pompeii. The motivation for this system is to allow viewers of archaeological entities to obtain extra information about particular entities in the form of a story. As people may have different backgrounds and interests, it is not sufficient to inundate them with all known facts for any given entity, so the system needs to be able to decide which information to include.

The stories are created using a schema-base approach. Each story consists of a set of entities and their associated attributes. The entities are the objects of interest (e.g., sculptures, paintings), the objects and events they depict and other entities that are related to them. Relationships between entities, such as "Bacchus was the enemy of Pentheus" and "Elektra and Iphigenia were daughters of Agamemnon", are discovered from additional texts and databases. The process of creating a story is viewed as a constraint satisfaction problem where a story schema is filled using collected data. Specifically, entities and their attributes are extracted from multiple data sources and are then transformed into English either by using the sentence they originated from or through the use of templates. Fayzullin et al. (2007) show the process of creating an optimal story under these circumstances is NP-complete which leads them to create heuristic algorithms for finding quick sub-optimal stories; by restricting the search space, using genetic programming techniques and employing dynamic programming. A human-based evaluation reveals that a dynamic programming algorithm produces the best stories and that the readers prefer shorter stories, although the use of template sentences may have contributed to this result.

2.1.6 Evaluation of Story Generators

There has been little consensus to date on how best to evaluate the performance of a given story generator, or how best to compare the performance of one story generator to another. This stems partially from the the fact that many story generators are developed in order to explore a specific theory, computational approaches or phenomenon (e.g., MEXICA, MINSTREL explore human memory processes and problem solving). These systems are not necessarily developed in order to produce 'good' stories, but rather to develop and test specific theories. This makes it difficult to gauge the performance of particular story generators and, as a result, the quality of the stories that they generate. In order for these systems to be directly comparable it would be necessary to define a set of criteria that is common to the stories that they generate. For example, reader

enjoyment and expectation, story coherence and creativity. These stories would then have to be evaluated in an un-biased manner by experts (this is usually taken to mean human readers). To date there have been few human elicitation studies performed to evaluate the stories generated by a given system, and even then, the evaluation criteria often differ. For example, planning systems tend to focus on story and character plausibility whilst those systems that generate textual stories focus on the quality of the generated text.

Any systematic evaluation of these systems must also contend with the fact that different systems will tend to use different data sources (for which those data sources have been developed exclusively) and focus on different aspects of the generation process (e.g, generating complex story planning whilst performing no rendering into natural language). Any standardised evaluation of story generation systems would have to take these factors into account as all systems are not necessarily directly comparable. For example, the impact of transcribing system output by hand for one system against template-based story output for another may greatly influence a human evaluators judgement of a story without necessarily providing the intended comparison of story quality. This is also an issue when the media of presentation for different systems is very different (e.g., interactive systems, comic strips).

2.2 Probabilistic Natural Language Generation

Natural Language Generation (NLG) has a long standing tradition in the field of Natural Language Processing. NLG allows us to transform non-linguistic representations of information into text. This information may be facts in a database, time series data, propositions, or discourse trees. Within NLG, we wish to focus on the probabilistic methods that are available since they are more central to the goals of this thesis. Reiter and Dale (2000) describe how an NLG system can be decomposed into distinct modules that form a pipeline process. An overview of the pipeline is shown in Figure 2.7. The input to a hypothetical NLG system is a communication goal that describes the desired output of the system (this is dependent on the purpose of the NLG system, e.g., the communication goal submitted to a weather forecast summary system will be in terms of the data it stores, such as ‘what was the weather like last month?’). The output of the system is a text generated in the target language. The pipeline itself can be decomposed into three distinct modules, document structuring, microplanning and surface realisation. These modules can then be further decomposed into a series of

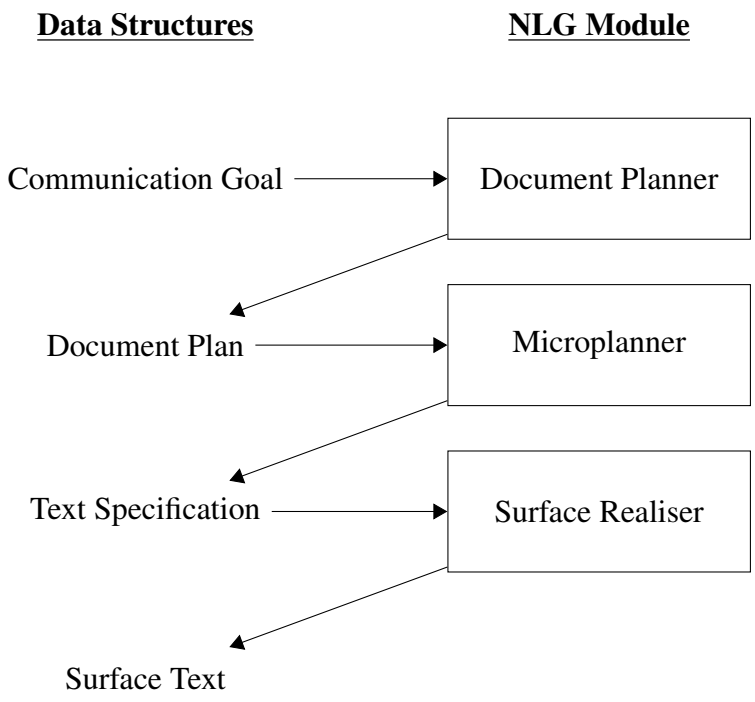


Figure 2.7: Overview of the natural language generation pipeline as described in Reiter and Dale (2000).

tasks described below:

Document Planner Using the data-sources of the NLG system, the document planning module will decide what information should be included in the produced text (content determination) and order it to match the readers expectation of a document of the desired domain (document structuring). The output of the module is a document plan, typically a tree where internal nodes represent structure and leaf nodes represent content.

Microplanner Although the document plan outlines structure and content, there are still decisions over how to express that information in natural language. The job of the microplanner is to decide upon the words and structures that will be used to communicate the document plan. This is broken into three tasks. The first is to select the words and syntactic constructs required (lexicalisation). Secondly, deciding how the document will represent the entities present (referring expression generation). Lastly, the mapping of the document plan to linguistic structures, resolving any unspecified ordering of the information (aggregation). The result of the microplanner is a text specification, once again a tree structure

in which the structural elements of the document are represented in the internal nodes, however the leaf nodes here represent sentences.

Surface Realiser The final module is concerned with taking the text specification and transforming it into actual text. This requires both transforming the representations of individual sentences into text (linguistic realisation) and also those of the larger structures such as paragraphs and sections (structural realisation). The output will then be the final text of the document.

Recent work has focused on improving these modules by constructing them in a probabilistic manner that facilitates learning. Previous work has included training components for the tasks of content selection (content determination), sentence planning (lexicalisation and aggregation), document structuring and surface realisation which are reviewed below:

2.2.1 Content Selection

When generating a text, we must first select what specific information we wish to include in it. For instance, there is an abundance of data that can be collected from the stock market during the course of a day, but any particular investor will not be interested in viewing all of it. The goal of content selection is to identify from the data sources available the information that is most useful for the reader. The traditional method for content selection is to use rules that define what information to include in the text. Recent work attempts to learn these rules automatically. In Duboue and McKeown (2002) a stochastic search method is used to learn tree-like structures for a content planner from a parallel corpus. Their training corpus contains information gathered from cardiac surgeries and summary briefings to post operation carers. A genetic learning process is employed that allows candidate planners to be scored with respect to their similarity with the corpus. They find that the learning algorithm outperforms randomly created planners.

Barzilay and Lapata (2005) acquire content selection rules from a parallel corpus containing statistics of American Football games and their text summaries. The goal here is to train the system to select the particular information that should appear in the summaries. Each document in the training set consists of a database of statistics about the game (e.g., a record of passes performed be each player) and human written summaries of the games. From the written summaries each database entry can then

be tagged to say whether or not it was included. From the training set, rules can be learnt based both on selection preferences of individual entities (e.g., touchdowns, passes) and on links between database entities (e.g., if you give the statistics for one player that scores, then you should also give those for any others that do the same). Their process of content selection therefore considers all of the entities in the database simultaneously, making it a *collective classification* task. The motivation is that a text in which the entities are related will be more coherent that one in which they do not. Both of these studies view content selection as a classification task although the latter focuses on finding a globally optimal solution. They also presuppose the existence of structured data sets and training examples for the domains in which they are to be implemented.

2.2.2 Sentence Planning

Once the content of the text has been selected, the words and syntactic structures that will form the text's sentences must be chosen. Stent et al. (2004) trained a sentence planner for the domain of restaurant recommendations where the original information was described as a set of assertions and the rhetorical relations between them. Their system, SPARKY (Sentence Planning with Rhetorical Knowledge) which is an extension to their previous system SPOT (Sentence Planner Trainable, Walker et al., 2002), which was domain dependant and only operated on simple content plans that did not contain rhetorical relations. For SPARKY, a ranking system was trained, using human annotated data, to select the best sentence plan trees generated by the planner. The learning algorithm used 7,024 real valued features that were automatically generated using feature templates. Their results showed that their trainable sentence planner could be extended to a new domain whilst still being comparable to template sentence planners that had been hand crafted for that particular domain.

Recent work by Mairesse and Walker (2008) has also shown that decisions on how to represent content can be made by modelling the intended author. Their work incorporated the Big Five personality model to generate text tailored towards a specific personality. The Big Five models consists of the features *extraversion*, *emotional stability*, *agreeableness*, *conscientiousness* and *openness to experience*. Structural decisions can then be weighted with their likelihood for inclusion in texts for readers with given personality traits, for example, extraversion may be strongly related to the use of exclamations. Through the use of these weights, the system can narrow down the space

of possible representations, removing those not suited to the readers personality traits. Both of these approaches require the content of each document to have been decided in advance. SPARKY in particular requires the rhetorical relationships between assertions in the document to be given as input. These relationships, such as, one statement contrasts another, are difficult to ascertain automatically and are therefore annotated by hand.

2.2.3 Document Structuring

It is crucial that any text being produced by a NLG system is coherent. To this end there are several different approaches to structuring a text in order to maximise coherence. There has recently been work to remove the reliance of text structuring on limited domain handcrafted rules by employing learning techniques. Mellish et al. (1998) use stochastic search methods in order to train a text planner for automatically generated descriptions of museum artefacts. Their approach focuses on identifying an optimal Rhetorical Structure (RS, Mann and Thompson, 1987) tree for a description. A genetic search process is used that evaluates candidate RS trees based on different metrics of document level information. This process allows them to select an optimal RS tree from the search space by creating successive generations of possible candidates that improved with respect to the scoring function. We shall discuss this approach in more detail in Chapter 5.

We can view a text as being composed of discourse units that must be placed in an order that achieves maximum coherence. Althaus et al. (2004) refer to this as the *discourse ordering problem* and show that it is NP-complete. Centering Theory (CT, Grosz et al., 1995; Walker et al., 1997) is used to measure the cost of transitions from one discourse unit to another. The problem to solve becomes that of finding the path through each of the discourse units that has the lowest cost. Using different optimisation strategies they were able to implement time efficient search procedures for short text segments.

The ordering of sentences in a generated text can affect the coherence of the overall text. Lapata (2003) trains an unsupervised probabilistic model that learns ordering constraints from a large corpus. Training involves learning the sequences of features that are likely to co-occur in coherent texts. These features are shallow, such as the verbs with their subjects and nouns with their modifiers. An acceptable ordering of the sentences can then be constructed by searching greedily though the space of possible

orderings. Their results show that low level lexical and syntactic features can be used to successfully train models for single and multiple document ordering tasks.

Local coherence relies strongly on the distribution of entities within a text. Barzilay and Lapata (2008) use an entity grid approach to capture the distribution of entities in text. The entity grid records the role (e.g., subject, object, missing) of each document entity in each sentence. The likelihood of a given transition, e.g., being the subject of one sentence and then being the object of the next, can then be learnt. This allows us to evaluate the coherence of human authored and machine generated texts from the probabilities of the entity transitions. We shall discuss this in more detail in Chapter 3.

The above approaches to document structuring differ in the granularity of document analysis required for their implementation, yet all assume that the basic units, be they sentences or utterances, have already been decided. Encouragingly, Lapata (2003) and Barzilay and Lapata (2008) only require shallow document features for training and implementation which improves their portability.

2.2.4 Surface Realisation

The process of converting the conceptual expressions, that represent a text, into natural language is by no means a straightforward task. It requires the integration of syntactic and lexical information, knowledge about morphology and tense, and so on. Knight and Hatzivassiloglou (1995) conduct their investigation of this area in the context of Machine Translation, where a symbolic representation of a sentence in the source language must be realised in the target language. The process they employed maps symbolic structures to word lattices (acyclic state transition networks with one start state, one end state and transitions labelled by words). The lattice effectively shows the possible realisations of the semantic input and by using a bigram model they are able to select the n-best paths through the lattice, depicting the best sentences. The sentences generated using bigram probabilities are found to outperform the selection of random paths through the lattice.

Word lattices were also used for surface realisation in Langkilde and Knight (1998) as part of their natural language generator, NITROGEN. Different levels of knowledge including lexical, morphological and grammar formulations are used to build up sections of the word lattice and combine them together. This bottom up process contrasts the inefficiency of top-down methods which often repeat the same calculations when processing. The main advantage of NITROGEN is that it utilises corpus based statistics

when making linguistic decisions giving it broad coverage.

2.3 Summary of Chapter

The majority of the story generation systems we have encountered create their stories to meet a specific user goal. For instance, asking for a story in which a hero overcomes a villain or even asking the system to give the description of a museum artefact. It is important to remember that stories are generated for the reader and therefore must cater for their needs or they will become ineffectual. Most of the above systems focus primarily on the structure and content of the stories that they produce and hand over the task of generating the story itself to either future work or other systems. This is in marked contrast with NLG where the aim is to generate well formed documents from symbolic or other representations.

The main differences, across story generation systems, stem from the way they form their stories and the data sources they use. For instance, agent-based systems focus on planning techniques with databases of actions, while commonsense systems focus on a sentence by sentence structure utilising a database of commonsense relations. These approaches draw inspiration from different areas of research which allows for different styles of stories to be created. For example, high levels of interaction between characters are plausible in agent-based stories, however, this is less evident in commonsense stories where the generation focuses on the logical progression of sentences rather than on any particular character's goals.

The reliance on handcrafted rules appears to the be the biggest problem facing current story generation systems. This means that the coverage available to each system is narrow and usually confines them to small domains. The process of creating handcrafted rules and databases is time consuming and must be repeated anew when extending these systems to new domains. To overcome the dependence on handcrafted data we propose to borrow and further develop ideas from recent work in probabilistic NLG. Trainable modules allow automatic discovery of rules and are used in different areas of text generation. So it seems feasible that they could be extended to discovering rules for story generation. In contrast to previous work, which focused on individual modules of the NLG pipeline, we propose to create an end-to-end probabilistic system. Previous work by Chen et al. (2002) has shown that the potential exists to combine trainable components together to create a portable NLG system. In particular, they combine a sentence planner and surface realiser and show that they can be

successfully used as part of a dialogue system.

Another tendency of story generation systems is to overlook the important role natural language generation has to play in producing a high quality narrative. For example, MEXICA (Pérez y Pérez and Sharples, 2001) and GESTER (Pemberton, 1989) bypass the process of generating text by utilising 'canned-text' sentences. The production of high quality narratives clearly involves an integration of the story generation process with the natural language generation pipeline. Callaway and Lester (2002b) propose an architecture AUTHOR and an implementation STORYBOOK for bridging the gap between the two fields. Taking as input, a narrative plan, they focus on how linguistic structures can be constructed to generate high quality prose. In particular, attention is paid to the use of a discourse history to improve lexical choice and generating referring expressions (we shall discuss this in more detail in Chapter 6). Lönneker (2005) argues that STORYBOOK does not fully close the gap between story generation and NLG as they focus primarily on microplanning and generating surface features. Instead, Lönneker (2005) suggests that an architecture for generating narratives must take the linguistic aspects of content determination and document structuring into consideration, rather than leaving them as the task of a separate story generator. They describe how the phenomena of narrative levels, "stories within stories", rely on the ability to embed narratives within one another during document planning. Also, making decisions on when to use flashbacks or ellipses (removing events that the reader can infer) requires such a system to consider the rhetorical structure of the document being developed.

In the next chapter we shall outline our first attempt at creating a story generation system that follows the NLG pipeline in which each of the components are trained. We will motivate a generate-and-rank procedure that makes use of the wealth of possible stories available from corpus extracted knowledge. Also, we will discuss evaluation techniques designed to automatically asses our generated stories.

Chapter 3

Generate-and-Rank Story Generation

In the previous chapter we presented current story generation systems and showed that they suffer both with respect to scalability and robustness. The majority of the systems we have seen utilise hand coded data representing a particular genre. In the agent-based systems, the planning rules used by the agents are entirely hand coded. This means that adding new actions to the story generator requires substantial effort. Several systems use Propp's functions as their basis. As these functions are rooted in Russian folk tales, this is the only domain in which they are able to produce stories. Due to their reliance on hand crafted data, the robustness of these systems is compromised. It is easy to foresee situations in which they will suffer from missing information, especially in cases where the user tries to input data the system is unfamiliar with. The reason for these problems is clearly linked to the difficulty in implementing knowledge in a fashion a computer system can understand. Without giving a system access to a large knowledge base, the effectiveness of its story generation abilities will be limited.

Within Natural Language Generation (NLG) the desire to escape the dependence on hand-crafted rules has given rise to probabilistic systems. These systems, however, do not tackle NLG as a whole but rather individual modules within the NLG pipeline. Our goal is to create an end-to-end probabilistic system for generating natural language stories.

In this chapter we introduce an approach to story generation that takes its inspiration from the recent work in trainable NLG. We follow the traditional NLG pipeline which decomposes generation into content selection, sentence planning, surface realisation and document planning. We realise each of these processes with trainable or off-the-shelf components. The system we outline in this chapter does not make use of explicit document planning, however, we will discuss in Chapter 4 how it can be

incorporated. We will also show that story generation can be viewed as a search task and outline measures for assessing the quality of a generated story automatically. We will then conclude, by showing through experimentation that stories generated by our generate-and-rank system are rated more preferably, by human evaluators, than those of two baseline systems.

3.1 The Story Generation Task

In this thesis we consider how simplified stories can be generated automatically from corpora. We define these simplified stories as being only a few sentences long and containing only a couple of entities. In addition, we view these stories as the composition of a series of actions that are typical of the entities participating in them. These actions represent the highest meaning bearing content within our stories. This means that we do not consider document level story structures, such as whether they portray a specific message or result in a specific goal. Although other story generation approaches focus on building models for character, author and reader level goals, our simplified stories are designed only to communicate a series of actions for a given set of story entities that meet the reader's expectations for those entities within a given genre. Our approach to story generation is thus to identify the entities and actions that are to be included in a story and compose them into a document for the reader. Although we are not concerned with high level document structures such as metaphor, these stories still need to be fit for purpose, they must be intelligible and entertaining enough that the reader will want to read them. The main goal of this thesis is to show that there is a feasible approach to automatically generating these simplified stories. We will also consider the limitation of our approach and the systems we develop with regards to more complex stories definitions.

Developing a story generation system to produce the simplified stories described above still faces several daunting challenges. The system must have access to a knowledge base of entities and actions, as well as the natural language resources for generating the final text. These resources need to be accurate to meet reader expectation, however, they must also be flexible enough that portability is not compromised. Managing the synergy between story content and document coherence requires a story generation approach that can perform multiple evaluations and search procedures. Another consideration is how to search through the multitude of entities, actions, sentences and stories in order to decide which one story should be presented to the reader. Below we

outline our first attempt at generating a system that can generate these stories: starting with a generate-and-rank approach we will show the abilities and limitations of that system before continuing in the following chapters to show how these limitations can be overcome through the addition of modelling story plots and the use of evolutionary search techniques. We will show how our approaches to generating these simplified stories perform and compare with one another, for the task of generating simplified stories.

3.2 The Story Generator

Below we outline a first attempt at relieving the knowledge acquisition bottleneck associated with computational storytelling by generating stories from an automatically created knowledge base. We propose that stories can be generated by reformulating information about entities, their attributes and interactions as attested within a corpus. For instance, we expect *dog* to appear as part of certain actions, such as *barking*, and we know that *food* will often be *eaten*. These relationships of the form *verb-subject* and *verb-object* can be automatically extracted from text using a state-of-the-art parser. Interestingly, these actions can form new completely different and unique story arrangements different to those found in the corpus. As Turner (1994) explains, we are capturing the milieu, or environment, in which the story entities exist. His system, MINSTREL, was able to tell stories about King Arthur and the Knights of the Round Table by providing it with information, such as, "knights love princesses and kill dragons", "hermits live in caves and heal people". Although MINSTREL can generated stories for other milieus, the overhead in time and effort is prohibitive. The information we extract from the corpus must therefore be representative of the environment in which our stories are to be generated, yet it must also be straightforward to extract to ensure portability.

As a proof of concept we initially focus on children's stories (see Figure 3.1 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, 2003b).

As common in previous work (e.g., Shim and Kim, 2002; Liu and Singh, 2002), we assume that the process of generating a story occurs in an interactive context. Specifi-

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

cally, the user supplies the topic of the story and its desired length. By topic we mean the story protagonists, i.e., the entities (or characters) around which the story will revolve. This topic can either be a list of nouns such as *prince* and *princess* or a sentence, such as *the prince marries the princess.* To simplify the problem, we start by limiting the number of entities in the topic to two. This means each story we generate will have two protagonists. We also assume that each sentence in the story must contain one of the story protagonists as its subject. This is an assumption inspired by Centering Theory (Grosz et al., 1995), which indicates that coherent texts tend to maintain their focus on particular entities from one utterance to the next. Each story must be presented in a form the reader can understand. We have elected that the medium for presentation will be English text, although some systems use alternative formats. For example, the Virtual Storyteller (Swartjes and Theune, 2008) uses an animated figure of a wizard, using speech synthesis to read the generated story out loud. Shim and Kim (2002) present their generate stories to the user as an animated comic strip.

Stories rendered as text are clearly literary artefacts and as such we can consider them as the intended output of NLG systems. We assume that NLG systems should conform to the NLG pipeline outlined in Reiter and Dale (2000). Lonneker et al. (2005) outline a theoretical system for producing narratives based on the NLG pipeline. They explain how the task of each module can be encoded for the task of building narratives, however their system relies on the specifications for each module being hand-coded, something we endeavour to avoid. An overview of our system is shown in Figure 3.2,

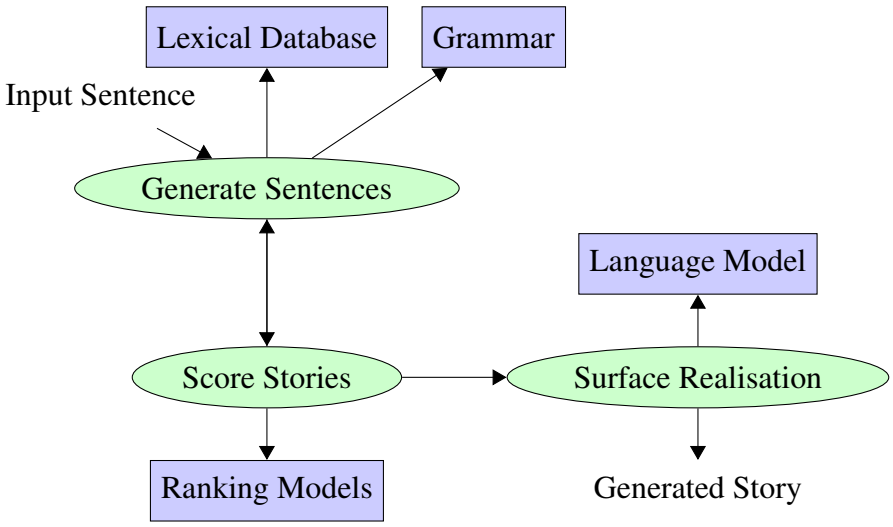


Figure 3.2: Overview of the story generation system. Ellipses indicate processes and rectangles indicate data sources.

ellipses represent processes and rectangles represent data sources. The input to our system is the user supplied topic sentence and the output, a story realised in English. Our approach follows a generate-and-rank framework, meaning that the system has the ability to create a large number of candidate stories which it can evaluate and rank. These stories are generated one sentence at a time. The process of generating sentences (see Generate Sentences ellipse in Figure 3.2) encompasses our system's content selection and sentence planning (the former making use of a lexical database and the latter a grammar), we apply these simultaneously rather than in a pipeline fashion. The generate-and-rank search procedure makes use of automatic evaluation (see Score Stories ellipse in Figure 3.2) using ranking models to decide which of the stories are better than others; bear in mind we want to return only one story to the user and that should be the best one. Finally, the selected story will undergo surface realisation (see Surface Realisation ellipse in Figure 3.2) which transform the abstract system representation of the story into English, using a language model to decide upon the best representation from those possible. As we mentioned earlier there is no explicit document planning in this system, rather the order in which sentences are generated will form the final story.

We now explain in more detail each of the procedures and data sources of the system, finishing with a worked example.

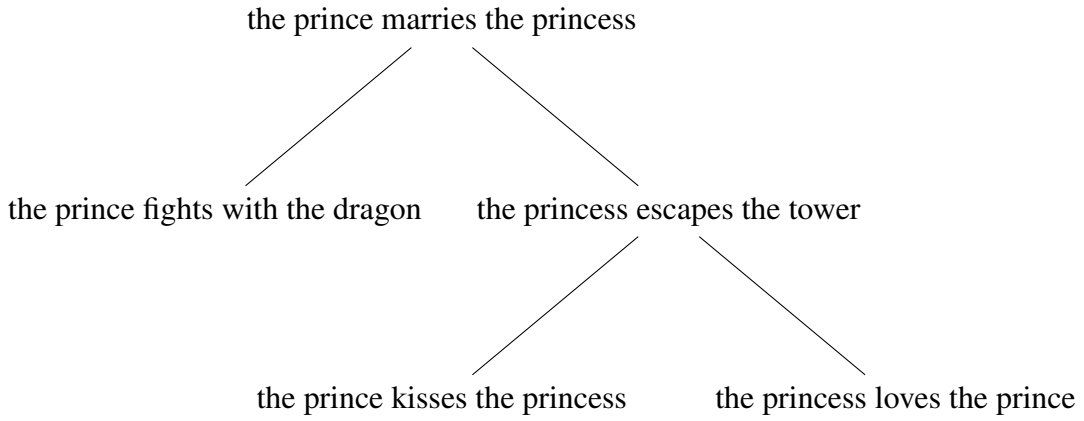


Figure 3.3: Example of a simplified story tree.

3.2.1 The Search Procedure

Our approach is to generate multiple stories involving the supplied topic entities by consulting a knowledge base containing information about them. So if we wanted a story about a *prince* and *princess* we can consult their associated actions (e.g., *princes fight*, *princesses escape*) and their interactions (e.g., *princes marry princesses*, *princesses love princes*). We conceptualise the story generation process as a tree (see Figure 3.3) 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. For example, the tree in Figure 3.3 can generate three stories; (*the prince marries the princess, the prince fights with the dragon*), (*the prince marries the princess, the princess escapes the tower, the prince kisses the princess*) and (*the prince marries the princess, the princess escapes the tower, the princess loves the prince*). We assume that each sentence in the tree has a score representing the quality of a story terminating at that sentence. Story generation can then be viewed as the process of searching this tree at the required depth for the highest scoring story.

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 3.6 and Algorithm B.1 in Appendix B). 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. Each story and sentence can be scored by specific criteria. We discuss scoring methods in Section 3.3.1.

3.2.2 Content Selection

The content selection procedure makes use of two different resources, both extracted from the corpus. A database of lexical relationships is used for sentence generation and an action graph is used to encourage logical actions progressions across sentences.

Lexical Database The process of content selection utilises a database containing information on entities, actions and how they relate. Specifically we use *subj-verb*, *verb-obj*, *verb-adverb*, *verb-verb* and *noun-adjective* relationships. These relationships can be extracted from a dependency parse tree of each sentence and we assume that entities and actions can are represented as simple word tokens, such as nouns and verbs. We can also give likelihoods for each of these relationships in order to build a model of the distribution of events and entities within a particular corpus. Above we suggested that it is common for *dogs* to *bark* and *food* to be *eaten*. If this is the case then we would expect to find a strong relationship between these entity-action pairs in the corpus. The characteristics of entities and actions, identified as adjectives and adverbs, are also important to capture. Think about how a description can change your opinion of a particular character, i.e., *the cruel prince* or *the heroic prince.*

This database is then used to indicate all the lexical items that are available for a given lexical dependency, e.g., what are the objects of the action *protect*? It is then up to the sentence generator to decide how these will be used. Scores can be assigned to each relationship based on their distribution in the corpus to aid in selection, we discuss the scoring functions used by our system in Section 3.3.1. In the event of a lexical item having no co-occurring relationships, the content selection process will generalise the identified entity (e.g., if we query the lexical database for adjectives that can describe a *prince* but we find that none were observed in the corpus). Using the WordNet (Fellbaum, 1998) hierarchy, we can locate those entities that are hypernyms[1] of *prince*. Although we may not have any adjectives for *prince* we may have encountered some for *aristocrat*, *leader* or *person*. We then select adjectives for *prince* from the closest hypernym for which adjectives are attested. The relationships created

[1]Hypernym relationships are of the form, *x is-a y*. For example, a *dog* is a *canine*.

through this process will clearly be weaker than those directly attested in the corpus, however, it does make the most of the lexical database and also ensures that a story can be created.

The stories we generate should be representative of the corpus from which the database was extracted. This means we want the entities that appear in our stories to act in a manner consistent with their behaviour in the corpus. We expect to find that the distribution of actions to entities will differ between corpora which should then lead to the entities performing differently within the generated stories. For example, consider the entity *prince*. In a fictional setting we could expect *dragon slaying* and *rescuing* to be recurring actions associated with *princes*. However, in the real world a *prince* is much more likely to be found opening shopping centres and meeting with heads of state and it is exactly this discrepancy that the database is intended to capture.

Action Graph The lexical database described above can only inform the generation system about relationships at 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 the discourse relations between propositions. Although we do not 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), who obtain *narrative event chains* from news text using unsupervised methods. These *narrative event chains* represent partially ordered actions performed by an entity, known as the protagonist, within a text. Similarly, we identify the events every entity relates to and record their (partial) order. Here we aim to build a global view of valid sequences of actions rather than those performed by a single protagonist within a text. 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*.

We define the *action graph* as a directed graph $G = (V, E)$ whose nodes V denote action-role pairs (the actions are verbs in the story and the role is either subject or object), and edges E represent transitions from node V_i to node V_j. We will discuss

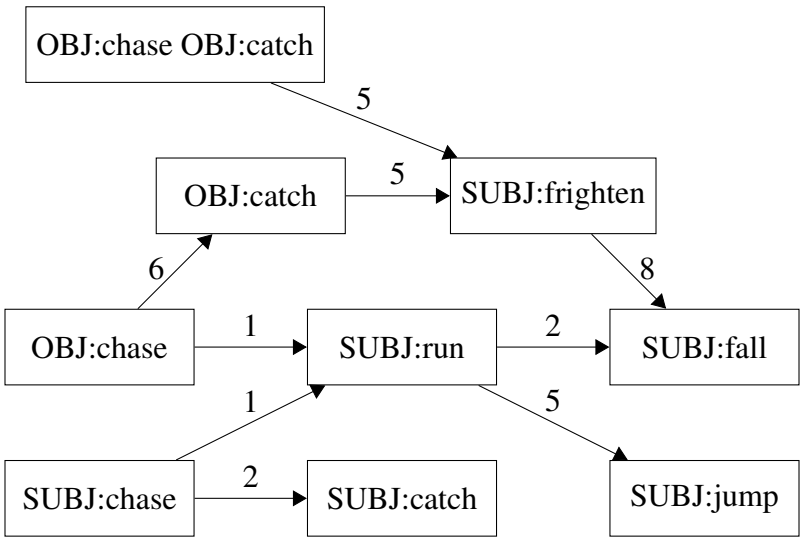


Figure 3.4: Example action graph encoding (partially ordered) chains of events.

how weights are calculated and used in Section 3.4.1. For each document in the corpus we record the actions that each entity participates in and its role in that action. 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. Once all chains of actions are identified, the entities are stripped and the chains are combined to a single graph. An example action graph is shown in Figure 3.4. These links appear as bigrams of entities but we have also added trigrams where a node in the graph represents the two previous actions. Each node links to the actions that will follow. From the example graph in Figure 3.4 we can see that an entity who was previously *chased and caught* (*OBJ:chase OBJ:catch*) will then be *frightened* (*SUBJ:frighten*) and *fall* (*SUBJ:fall*). As before we can gain insight as to how likely action transitions are by viewing their frequency of occurrence. The action graph then allows the system to give preference to actions which follow on in a logical sequence from those it has already encountered.

Discourse History In order to maintain consistency throughout a story being generated, we must allow the system to maintain a discourse history of entities and events that have occurred thus far in a story. There are two entities that are the focus of the

story but they are unlikely to be the only entities that will appear. The positioning of entities within a text is related to both coherence and reader expectations. Centering Theory (CT, Grosz et al., 1995) argues that the focus of a discourse revolves around a centred entity. It is also assumed that texts whose utterances share a common center are more coherent than texts where the center shifts from one utterance to the next and that salient entities are more likely to be centers. CT thus allows us to make predictions about which entities are likely to reoccur within a story. We can imagine the complete break down of coherence that would occur if a story simply introduced each entity with a single mention and then never referred to it again. We have already taken a step towards enforcing coherence by only allowing one of our two topic entities to occupy the subject role of each sentence. However, we can further enhance story coherence by giving preference to those entities that appeared previously in the story regardless of whether they are the most likely entity to occupy that role within an action. The content selection process, when generating a particular sentence, depends not only on the database but also the sentences that preceded it (see Algorithm B.2 in Appendix B).

Using the action graph requires the discourse history to record the actions performed by each entity in a given story. The system then selects a list of expected actions for an entity based on the previous one or two actions that it performed and uses these to reorder the actions available for the entity to perform from the lexical database.[2] Also, in order to maintain consistency we only allow verbs to be selected from the action graph, for a given entity, if that verb is also related to that entity in the database.

3.2.3 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 subcategorisation information which we collected from the lexicon created by Korhonen et al. (2006) and the COMLEX dictionary (Grishman et al., 1994). A selection of these rules are shown in Table 3.1. The grammar rules act as templates which have the structural form of a dependency tree. Each node represents a lexical item, and each rule has a verb node as its root. Each grammar node takes one of two possible forms, primary or auxiliary. Primary nodes are content bearing, whereas auxiliary nodes contain information pertaining to sentence structure.

[2] In the event that there are no attested actions in the action graph based on the entity's history, the system uses the subject verbs for the entity that appear in the content selection database.

Rule Name	Rule		
INTRANS	[VP rel:@r lex:@v1]	→	$[NP role:S rel:I lex:@n1]
NP	[VP rel:@r lex:@v1]	→	$[NP role:S rel:I lex:@n1]
			$[NP role:O rel:II lex:@n2]
NP-NP	[VP rel:@r lex:@v1]	→	$[NP role:S rel:I lex:@n1]
			$[NP role:O rel:III lex:@n2]
			$[NP role:X rel:II lex:@n3]
ADJP	[VP rel:@r lex:@v1]	→	$[NP role:S rel:I lex:@n1]
			$[ADJP rel:ATTR lex:@a symb:@a-adj-@n1]
PP	[VP rel:@r lex:@v1]	→	$[NP role:S rel:I lex:@n1]
			([PP lex:@p rel:ATTR] →
			$[NP role:O rel:II lex:@n2])
INF-AC	[VP rel:@r lex:@v1]	→	$[NP role:S rel:I lex:@n1]
			$[VP subj:null rel:II lex:@v2 mood:inf]
HOW-S	[VP rel:@r lex:@v1]	→	$[NP role:S rel:I lex:@n1]
			([X lex:how rel:II] →
			$[VP rel:II lex:@v2])

Table 3.1: Example of grammar rules used by the sentence planner. Nodes prefixed with a *$* are those that require satisfaction by the system. Variables are prefixed with an *@*. Rules contain a set of attributes which include; *lex* the lexeme of the node, *rel* the role of the node in the dependency structure, *role* the role of the entity in the clause, *mood* the mood of the verb for the realisation and *symb* which adds a relationship between two lexical variables.

The primary nodes represent *nouns*, *verbs*, *adverbs* and *adjectives*. These nodes require their lexemes to be selected by the content selection module during generation. In addition, some rules call for auxiliaries such as *prepositions* and *particles*. Many of these auxiliaries are specific to the rule itself, for example the *HOW-S* rule in Table 3.1 contains a node with the set lexeme *how* that governs the subclause. In addition to supplying the information on which rules attach to which verbs, the lexicons also supply the lists of *prepositions* and *particles* for these rules. A default list of prepositions was used in the event that no prepositions were assigned to a given rule. To give an example, for the verb *play*, the rule *PP* is assigned with the list of prepositions: *for*, *on*, *upon*, *against* and *with*. This allows the system to construct sentences such as, *the prince plays upon the horse*.

Due to the subcatagorisation structure of the rules, the choice of verb during generation will affect the structure of the sentence. The subcategorisation templates are weighted by their probability of occurrence in the reference dictionaries. This allows the system to prefer less elaborate grammatical structures. We also grouped together rules with similar content structures. For instance, the primary nodes of the rules *NP* and *PP* are identical, only the latter has a preposition governing the object. By combining the rules that have identical primary node structures we are delaying the decision as to the structure of the final sentence. This is an important step in the way our generation system works as the placement of primary nodes is required before the content words for the sentence can be chosen. The structure of the sentence, however, has an impact on the meaning of the sentences even after the content words have been finalised. Taking the example of the rules *NP* and *PP* again, *the prince hides the treasure* and *the prince hides from the treasure* have very different meanings and making the wrong choice can adversely effect the quality of the produced story. The decision on which of the available rules to use in our case is thus deferred to the surface realiser (see Section 3.2.4).

Both lexicons treat the verb *be* different from other verbs so they did not directly associate it with any rules. For *be* we manually selected the four most frequently occurring rule groups (excluding the *INTRANS* rule which would only generate sentences of the type *the prince is*) along with the *ADJP* rule as this will allow the system to generate simple descriptive sentences. The *ADJP* rule was chosen by inspection to be useful for the types of stories we hoped to generate.

Some rules also contain information about entity placement, e.g., by requiring that the entity in the subject role also appear as the subject of any rule appended as a

subclause. To simplify our evaluation of story coherence, described in Section 3.3.3, we also added each nouns functional information, i.e., whether they were the subject (*S*), direct object (*O*) or indirect object (*X*) of the governing verb. The grammar rules also contain information that will be used by the surface realiser, described in the next section. Certain grammar rules dictate the manner in which their resulting sentences should be realised. For example, the *NP-P-POSSING* rule states that the verb of the subclause be realised as a present participle, creating a sentence such as, *the prince wakens the dragon by* ***stealing*** *the gold.*

3.2.4 Surface Realisation

The final step for the system is to take the generated story and transform it into English text for presentation to the user. Up to this point the story has been represented as an ordered list of dependency trees. The problem is that each of these trees may have multiple representations and the job of the surface realiser is to select the best overall rendering for the story. Our surface realisation module currently interacts with the REALPRO generation system (Lavoie and Rambow, 1997). This system takes a Deep-Styntactic Structure (DsyntS) tree and using a rule-based approach transforms it into English. The Dsynt structure was inspired by Mel'čuk (1988) and is ideal for our representation as the syntactic relationships are labelled with grammatical information (e.g., subject, object) and each node corresponds to a meaning bearing lexical item (i.e., the lemma). An example of a DsyntS tree for the sentence *'The prince runs quickly up the stairs'* is shown in Figure 3.5.

There are, however, several grammatical issues that will affect the final realisation of the sentence that cannot be included in the subcategorisation rules. 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 and particles can either be pre-verbal or post-verbal. Verbs that appear as sub-clauses may take on different mood attributes, whether they are imperative, infinitives, etc. Prepositions and particles will arrive at the surface realiser as lists as they are not directly content bearing. The realiser is therefore required to select the most appropriate lexeme from those available for each of these word types. As we mentioned in the previous section, some sentences may be realised using several possible structures. 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.

```
DSYNTS:
run [class:verb tense:pres](
      I prince [class:common_noun article:def]
      ATTR up [class:preposition] (
            II stair [class:common_noun number:pl article:def]
      ATTR quickly [class:adverb position:post-verbal]
      )
)
END:
```

Figure 3.5: Example of a Deep-Syntactic Structure (DsyntS, Mel'čuk, 1988) tree for the sentence *'The prince runs quickly up the stairs'*.

Since we do not know a priori which of these parameters result in a grammatical sentence, we generate all possible combinations and select the most likely one according to a language model. When scoring with the language model we are looking for the sentence that has the lowest perplexity[3]. This is an approach used when generating text, for example in Machine Translation, where the final translation may have many possible realisations. We used the SRI tool-kit[4] to train a trigram language model on the British National Corpus[5], with interpolated Kneser-Ney (Kneser and Ney, 1995) smoothing and perplexity as the scoring metric for the generated sentences. Once again we encounter a search problem with a potentially large number of possible candidates. We reduce the number of candidates for scoring by searching for the correct representation of the lowest clause in a sentence first before moving to its parents. It is possible that the realisation that result from this approach may not be the optimum but for sentences that contain multiple prepositional phrases it is not feasible to try all possible realisations of the sentence. As another simplification, we assume that the input entities are singular and are proceeded by a definite article. A history is also maintained so that once the parameters of a noun have been established for one sentence they will be repeated in all subsequent sequences in which that noun appears.

[3]Perplexity can be thought of as the level of surprise resulting from reading the next word in a sequence. The perplexity should be low when a sequence of words conforms to our expectations of a language. Formally, perplexity is defined as 2^H where H is the the entropy, a measure of the uncertainty associated with a random variable (see Jurafsky and Martin, 2009, chap. 6).

[4]Available from `http://www-speech.sri.com/projects/srilm/`

[5]Available from `http://www.natcorp.ox.ac.uk/`

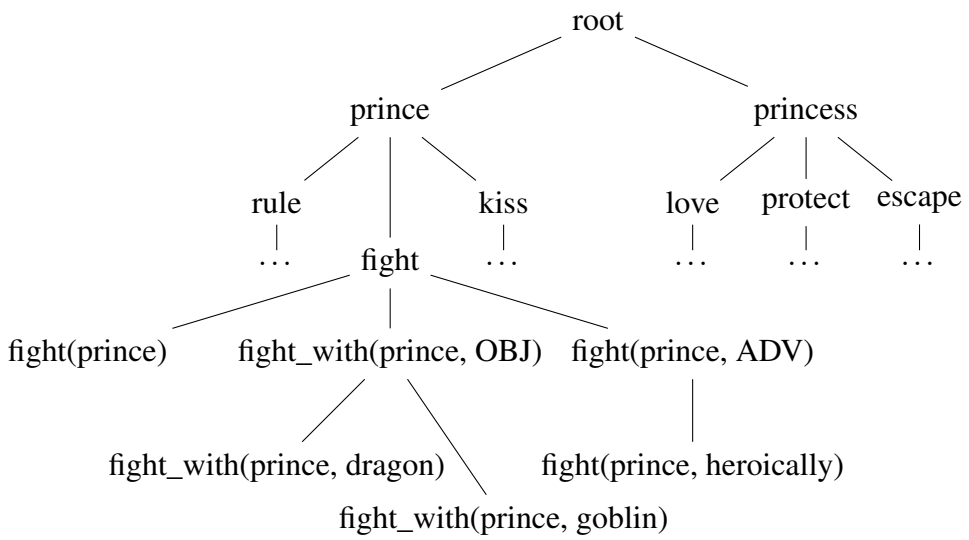


Figure 3.6: Simplified sentence generation example for the input sentence *the prince marries the princess*.

3.2.5 Example

It is best to illustrate the generation procedure with a simple example (see Figure 3.6). Given the sentence *the prince marries the princess* as input, our generator assumes that either *prince* or *princess* will be the subject of the following sentence. As we explained earlier, 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 *prince* and *princess* as their subject (e.g., *fight, kiss, love*). Our beam search procedure will reduce the list of verbs to a small subset by giving preference to those that are likely to follow *marry* and have *prince* and *princess* as their subjects or objects.

The sentence planner gives a set of possible frames for these verbs which may introduce additional entities (see Figure 3.6). For example, *fight* can be intransitive or take an object or adverbial complement. We select an object for *fight*, by retrieving from the knowledge base the set of objects it co-occurs with. Our surface realiser will take structures like *fight(prince,heroically)*, *fight_with(prince,dragon)*, *fight_with(prince,goblin)* and generate the sentences *the prince fights heroically*, *the prince fights with the dragon* and *the prince fights with the goblin*. This procedure is repeated to create a list of possible candidates for the third sentence, and so on.

As Figure 3.6 illustrates, there are many candidate sentences for each entity. In

default of generating all of these exhaustively, our system utilises the likelihoods 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. An overview of the approach is given in Algorithm B.1 in Appendix B.

3.3 Searching the Story Space

We can think of all the possible stories existing within a search space defined by the content selection database. These stories contain differing content words, sentence structures and length. One strategy for searching this space is to always follow the path with the highest score. However, this approach is not guaranteed to lead us to the best scoring story overall.

The first question that comes to mind when considering the story space is, what is its size? This is a non-trivial question and is ultimately dependent on the size and structure of the content selection database. An entity may be the subject of many actions, leading to several template structures which are possibly a single clause of an ultimately larger sentence. For example, a verb may take both a direct and indirect object, each of which could take one of 40 possible nouns. This would result in 1600 versions of this sentence. This may seem like a small number but consider that this verb may only be a subclause of the matrix verb, which itself may have a similar number of variations. It is now quite easy to see how the search space can explode as the size of the content selection database increases.

Secondly, how do we evaluate a complete or partial story and select which path in the search space to follow? As we established, the search space has a potentially intractable size making a complete search impossible. We therefore, prune the search by removing solutions that look unprofitable. Ideally, we would like to include all actions associated with each entity as we do not know which of them will lead to the better story overall but this approach is not practical. Pruning is required at both the sentence and document level. So, when generating sentences we focus on events that are more strongly associated with the noun. We do this by assigning likelihood to the stored syntactic relationships (e.g., *subj-verb*, *verb-obj*) in the content selection database. This allows us to prune away the unlikely events associated with each entity, leaving those that we would actual expect the entity to perform.

Pruning is also required at the story level. As we mentioned earlier, a beam search

can be employed for this purpose. A cache of solutions is held and when we reach a point where the cache becomes full we simply reduce the number of stories. We can also prune at each level of the story search to reduce the number of candidates under consideration when we start generating the next level of our tree search. A scoring measure is utilised to rank each of the stories in the cache and the *N* best are kept. This approach has been used successfully in previous NLG systems (see Walker et al., 2001; Knight and Hatzivassiloglou, 1995). A useful feature of a rank based system is that it allows for multiple scoring measures which may rate differing aspects of the candidate solutions. We will now discuss how we select the stories we keep in the cache. This brings us to one of the most fundamental questions for a search based story generator, namely “What makes a good story?"

3.3.1 Scoring a Story

In essence, we must now define what makes a ‘good’ story. It is unlikely that everyone will appreciate a story in the same way as people are often influenced by their personal background, age, knowledge and interests, ect. Therefore, we need to look for overarching features present in all stories which we can be used to judge their quality. Turner (1994) identified what he considered to be important features of a story. These include interest, coherence, originality and creativity. Turner was keen to investigate to what extent a story generation system could be considered creative. He argued that creativity emerges in problem solving through the retrieval and adaptation of previous solutions Our approach is different, as it is not a problem solving system in which satisfaction of author and character level goals drive the production of the final text. As our system generates stories at the syntactic and lexical level, our evaluation must reflect this. Therefore assessing our stories in terms of interest, coherence and originality, but not creativity. As our stories are generated from relationships extracted by decomposing the corpus, we can assume that each story will be original with respect to the original corpus.

Coherence is a factor in generated texts that has garnered much interest in the past (see Reiter and Dale, 2000 and the references therein). Any NLG system that produces text must ensure that the reader will be able to understand its content and this holds true for stories generated by our system. However, the interest of a story deserves more consideration. Interest is subjective but there are some key features we tend to look for in a story, e.g., does it have a good plot, is there a twist or suspense, even humour?

This is somewhat problematic as many of these features exist at the document level and we have no means of determining them automatically. A crucial factor in any evaluation performed by our system is that it is performed quickly and automatically. Human readers make the best judges, but when considering a system that may produce hundreds of thousands of stories in a single run, this is simply not practical. Any measures utilised must therefore be performed by the system itself and be fast enough to allow for a reasonable runtime.

Our focus will be on models that utilise the surface features of texts, as these are easily observable. In particular, we will use these surface features to train models for interest and local coherence. Surface features have already featured highly in work on recognising local coherence in texts (Barzilay and Lapata, 2008). Combining these methods will allow the system to evaluate the generated stories and effectively score them. Traversing the search space, we can then move towards an optimal solution, the most interesting and coherent story.

3.3.2 Modelling Interest

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 Æsop'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. Short stories were preferable as interest may vary during longer stories making it harder to associate interest to specific lexical features.

We asked native English speaking participants to judge 40 fables on a set of criteria: plot (how strong was the plot of the story?), events (how novel were the events in the story?), characters (how believable were the characters?), coherence (how easy was the story to follow?) and interest (how interesting was the story?). Ratings were given on a 7-point scale. These criteria were chosen to help us predict what human readers were looking for when deciding the interest of a story. In addition, the evaluators were given the chance to write down their reasoning for scores they assigned to each story. The fables were split into 5 sets of 8; each participant was randomly assigned

Features	Interest	Features	Interest
NTokens	0.188**	NLET	0.120*
NTypes	0.173**	NPHON	0.140**
VTokens	0.123*	NSYL	0.125**
VTypes	0.154**	K-F-FREQ	0.054
AdvTokens	0.056	K-F-NCATS	0.137**
AdvTypes	0.051	K-F-NSAMP	0.103*
AdjTokens	0.035	FAM	0.162**
AdjTypes	0.029	CONC	0.166**
NumSubj	0.150**	IMAG	0.173**
NumObj	0.240**	AOA	0.111*
MEANC	0.169**	MEANP	0.156**

Table 3.2: Correlation values for the human ratings of interest against syntactic and lexical features; $^{*}: p < 0.05$, $^{**}: p < 0.01$.

one of the 5 sets to judge. We obtained ratings (440 in total) from 55 participants, using the WEBEXP[6] (Keller et al., 2009) experimental software over the internet. The experimental materials are given in Appendix A.

We next investigated if easily observable syntactic and lexical features were correlated with interest. Participants gave the fables an average interest rating of 4.27 (SD 1.70). 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. Features were also extracted from the MRC Psycholinguistic database[7]. In particular, as we are generating children's stories, we are interested in the age at which words are acquired and the imagery of such words (see Bird et al., 2001). Using the MRC database, tokens were 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 using Pearson's r was performed to assess the degree of linear

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

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

relationship between interest ratings and the above features. The results are shown in Table 3.2. As can be seen the highest predictor is the number of objects in a story followed by the number of noun tokens and types, indicating interesting stories contain actions with more than one participant and the stories themselves contain several different entities. Imagery, concreteness and familiarity all seem to be significantly correlated with interest. Although the number of sentences in a story was not a significant predictor, as each story was approximately the same length, the length of those sentences was. This is reflected in the significant correlation between interest and the number of entities and actions in each story. Regressing the best predictors from Table 3.2 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.

However, it is important to point out that there are some caveats with regards to training an interest model on Æsop's Fables. First, Æsop's fables contained inherent properties that posed several difficulties to our study. Several contain archaic words, some of which could have been unknown to our native English speaking participants. Secondly, and more importantly, some of these stories are well known. Such as the "The Hare and the Tortoise" story. These stories tend to be rated either very positively or very negatively in terms of interest. Participants giving low ratings so due to their familiarity with story content, not the quality of the story itself. Also, several participants indicated that they disagreed with the underlying message and characters in the fables, strongly influencing their ratings of interest. Fables are a very particular form of story and are similar in style to parables, containing an underlying morale. Many of the moral messages contained in the fables are now out of fashion, with their value diminished in modern society. In order to portray these morals, the characters within the fables (mostly animals) quite often act in a manner contrary to our expectations. For example, "The Stag and the Pool", contains a particularly vain deer, a characteristic not often associated with that particular animal. Appendix A contains examples of fables with their associated comments.

A model of interest, based on the findings of human experiment, was trained using the fables combined with the human ratings as training data. From each document we extracted those features that were shown to be significant predictors in our correlation analysis (the features are those shown in Table 3.2 that are marked with an asterisk). Word-based features were summed across each text in order to obtain a representa-

	Plot	Coherence	Action	Interest
Character	0.535**	0.409**	0.306**	0.408**
Plot		0.328**	0.495**	**0.703****
Coherence			0.3077**	0.318**
Action				**0.644****

Table 3.3: Correlation values for the human ratings of Æsop's Fables; ** : $p < 0.01$

tion for the entire story. We used Joachims (2002) SVM^{light} package[8] to train a linear model preference ranker (all parameters set to their default values). Each training example represents a single interest rating, from a single human evaluator, for a given document and consists of both the numerical rating and a vector of the documents features. An evaluation was conducted of the model's performance to predict the ranking of documents for which the interest ratings were already known. Due to the small size of the data set, evaluation was carried out using cross-validation on the training set. The training set was split into 8 sets of size 5 and cross-validation involved training the model on 7 of the sets whilst testing on the last. The test set for each group contained 5 documents for which the interest score was derived by taking the average of the interest scores given to it by the human evaluators. Using these scores we can then assign a ranking to the documents. To evaluate we used Kendall's Tau statistic. The latter computes the distance between the order of the models produced ranking in comparison to the original ranking by recording the number of inversions it would take to produce it.

$$\tau = 1 - \frac{2(\text{number of inversions})}{N(N-1)/2} \tag{3.1}$$

Where N is the number of items in the ranking and the inversions are the interchanges necessary to create the new ranking. A score of 1 is given to a those that are identical, -1 is given to those that are in the completely reverse order. Our evaluation shows that the model achieved an average Kendall's tau correlation of 0.948 (SD 0.025) with the human ratings.

Table 3.3 shows the correlation matrix for character, plot, coherence and interest. As we can see, the perceived strength of the plot and the novelty of actions are the strongest indicators of interest in a story. The coherence of the stories is less correlated, but this makes sense as we can expect human authored stories to be coherent.

[8] http://www.cs.cornell.edu/People/tj/svm_light/

The fact that the believability of the characters is not so strongly correlated with interest may indicate that the readers are not so interested in whether the character is likely to perform an action, but rather, what that action is and how it fits into the plot. It is clear that the readers were looking for a strong plot, but as we mentioned before this is a largely document level feature of a story and can be difficult to discern automatically. However, we found that the plot ratings also strongly correlated with the lexical features of the fables, supporting our use of the interest ratings as a basis for a trainable model.

3.3.3 Modelling Coherence

In addition to evaluating the interest of a story, we also want to ensure that the story produced is coherent. This is not a problem with Æsop's fables as we can assume that coherence is inherent in human authored stories. However, as our system generates each story one sentence at a time, we need to make sure that the story, is overall coherent. Considerable research has been conducted into how to generate coherent texts, but this has largely focussed on the use of rhetorical relationships. As our system uses the shallow features that can be extracted from text, we do not make use of rhetorical relationships that model global coherence. As a result, we shall resort to assessing the coherence of each document using a model of local coherence. Local coherence is intended to capture the relatedness between sentence transitions. As our story generation approach searches for the next sentence to add to each current story hypothesis, we are interested in how these sentences will affect the coherence of each story.

We evaluate the local coherence of a text using an *entity grid* as described in Barzilay and Lapata (2008). This approach represents each document as a 2-dimensional array in which the columns corresponds 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, direct or indirect object. This entity grid is then converted into a vector of entity transition sequences. From a corpus we can learn the probabilities of such transition sequences and then use them to calculate the probability of the sentence transitions in a given text. An example entity grid is shown in Figure 3.7.

This approach was successfully used in Barzilay and Lapata (2008) to learn a ranking function for sentence ordering. Following their work, we train our model using artificial test data created by permuting the sentences of a text and then ranking the resulting texts by their proximity to the original. We can assume that the original doc-

The **prince** loves the **princess**. The **princess** sees a **dragon** near the **city**. The **prince** draws his **sword** and goes to meet the **dragon**. The **prince** slays the **dragon** with the **weapon**. The **princess** kisses the **prince**.

	prince	princess	dragon	city	sword	weapon
1	*S*	*O*	–	–	–	–
2	–	*S*	*O*	*X*	–	–
3	*S*	–	*X*	–	*O*	–
4	*S*	–	*O*	–	–	*X*
5	*O*	*S*	–	–	–	–

SS	*SO*	*SX*	*S–*	*OS*	*OO*	*OX*	*O–*	*XS*	*XO*	*XX*	*X–*	*–S*	*–O*	*–X*	*– –*
.04	.04	0	.08	.04	0	.04	.08	0	.04	0	.08	.08	.08	.08	.29

Figure 3.7: An example entity grid and vector for a given story. Rows correspond to sentences in the story and columns to story entities. Grid cells correspond to grammatical roles: subjects (S), direct objects (O)and indirect objects (X). If an entity is not present in a sentence then it is recorded as (–).

ument will be coherent. We trained our model on a selection of documents from the Andrew Lang fairy tales collection (see Section 3.4.1). We selected fairy tales rather than Æsop's fables as fables tend to contain only a couple of sentences, making it difficult to extract sufficient large transition sequences to train effectively. A more detailed account of our model of local coherence is given in Section 3.4.3.

3.4 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, the system parameters, the baseline systems used for comparison, and explain how our system output was evaluated.

Once upon a time there lived a poor fisher who built a hut on the banks of a stream which, shunning the glare of the sun and the noise of the towns, flowed quietly past trees and under bushes, listening to the songs of the birds overhead. One day, when the fisherman had gone out as usual to cast his nets, he saw borne towards him on the current a cradle of crystal. Slipping his net quickly beneath it he drew it out and lifted the silk coverlet. Inside, lying on a soft bed of cotton, were two babies, a boy and a girl, who opened their eyes and smiled at him. The man was filled with pity at the sight, and throwing down his lines he took the cradle and the babies home to his wife. The good woman flung up her hands in despair when she beheld the contents of the cradle. "Are not eight children enough," she cried, "without bringing us two more? How do you think we can feed them?" "You would not have had me leave them to die of hunger," answered he, "or be swallowed up by the waves of the sea? What is enough for eight is also enough for ten." The wife said no more; and in truth her heart yearned over the little creatures. ...

Once upon a time there lived a King and Queen, who were the best creatures in the world, and so kind-hearted that they could not bear to see their subjects want for anything. The consequence was that they gradually gave away all their treasures, till they positively had nothing left to live upon; and this coming to the ears of their neighbour, King Bruin, he promptly raised a large army and marched into their country. The poor King, having no means of defending his kingdom, was forced to disguise himself with a false beard, and carrying his only son, the little Prince Featherhead, in his arms, and accompanied only by the Queen, to make the best of his way into the wild country. They were lucky enough to escape the soldiers of King Bruin, and at last, after unheard-of fatigues and adventures, they found themselves in a charming green valley, through which flowed a stream clear as crystal and overshadowed by beautiful trees. ...

Figure 3.8: Extracts of stories from the Andrew Lang fairy tale corpus.

3.4.1 Corpus Data

We trained the content selection database and action graph on a corpus consisting of 437 stories from the Andrew Lang fairytale collection[9]. Sample excerpts from the corpus are shown in Figure 3.8. We selected fairy tales as they contain characters and themes children will be familiar with. The average length of a fairy tale is 125.65 sentences (SD 76.82) with an average sentence length of 15.29 words (SD 4.57) and there are 1,254,306 word tokens present in the corpus consisting of 15,789 word types. Each document was parsed using RASP (Briscoe and Carroll, 2002). The parser computes part of speech (POS) tags for each word and then through syntactic analysis outputs the grammatical relations that appear within the text. We discarded word tokens that did not appear in the Children's Printed Word Database[10], a database of printed word frequencies as read by children aged between five and nine, as well as any word not found in WordNet (Fellbaum, 1998). We extracted binary relationships, for example *(dog:subject:bark)*, and stored the number of occurrences of each relationship. We only retained relationships that appeared at least 5 times in the corpus. We only stored common-nouns, proper-nouns and pronouns were discarded; we also split compound nouns. Verbs were stored in their lemmatised word form.

Using the counts of occurrences that we collected from the corpus, we scored each grammatical relation using the mutual information (MI) score proposed in Lin (1998):

$$MI = ln\left(\frac{\parallel w,r,w' \parallel \times \parallel *,r,* \parallel}{\parallel w,r,* \parallel \times \parallel *,r,w' \parallel}\right) \tag{3.2}$$

where w and w' are two words related by relation type r'. $*$ denotes all words in that particular position and $\parallel w,r,w' \parallel$ represents the counts of w,r,w' occurring in the corpus. MI scores, unlike likelihoods, do not give an indication of which relationships are most likely to occur in the corpus. Rather, they indicate the uniqueness of these relationships. We would therefore expect the relationship between *dog* and *bark* to be scored much higher than that of *dog* and *be*, not because we find that *'dog is'* is less prevalent in the corpus but rather that the majority of entities can also be attested with the verb *be* but not so with *bark*. We can therefore, think of *dog* and *bark* as having a unique relationship that other entities do not share. Mutual Information was used for selecting subject-verb, verb-adverb, noun-adjective, verb-verb (within a sentence) and verb-verb (between sentence) relations. The MI scores were calculated for all co-occurring word pairs of the aforementioned relations. Any relations that had an

[9] Available on-line at `http://www.mythfolklore.net/andrewlang/`

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

MI score less than 0 were disregarded as being unlikely (see Algorithm B.3 in Appendix B). MI scores were also calculated for both bigram and trigram relationships in the action graph, the first two actions of the trigram being treated as a single relationship.

Mutual information allows us to score binary relations, however, this may lead to problems when likely selections do not make sense in a ternary relation, such as subjects and objects of verbs. For instance, although both *dog:subject:run* and *president:object:run* are probable, we may not want the system to create the sentence *The dog runs for president*. Some verbs can accept up to two object entities if they are transitive so we also need to stop unlikely objects appearing together, such as *John went for a walk in the mirror* although *a look in the mirror* would have been fine. To help reduce these problems we need to satisfy subjects, verbs and objects together at the same time. To do this we gathered data from the corpus that allowed us to calculate the conditional probability;

$$p(o1, o2 \mid s, v) = \frac{\| s, v, o1, o2 \|}{\| s, v, *, * \|} \tag{3.3}$$

where s is the subject of verb v, $o1$ is the first object of v and $o2$ is the second object of v and $v, s, o1 \neq \varepsilon$. When faced with a unknown set of object relations the system will first try and select those object entities with this conditional probability before backing off and selecting them with the MI scores (see Algorithm B.4 in Appendix B).

3.4.2 Corpus Analysis

Before presenting our system's output we analyse the corpus on which it was trained, to give an indication of the possibilities it holds, but also to highlight the challenging nature of our task. In total the fairy tale corpus contains 1,541 noun tokens, 724 verbs, 218 adverbs and 326 adjectives. In Table 3.4 we see the 10 entities that appear most frequently in the corpus, and in Table 3.5 the top 10 verbs. It is encouraging to see that within the most frequent entities we find those that we would expect to appear in fairy tales, namely, *king*, *prince* and *princess*. These are entities we presume will assume protagonist roles in fairy tales. However, there are also some unexpected entities in this list, such as *way* and *head*. These words are common throughout the corpus as supplementary entities (e.g., *'...but the daughter had forgotten the way'*), protagonists tend to appear frequently across a document whereas these supplementary entities are prevalent in the document collection as a whole. These also include the top two nouns,

Noun	Subject	Object	Adjective	Total
man	2223 (204)	1466 (104)	1879 (43)	5568
woman	887 (148)	598 (92)	1000 (33)	2485
king	1318 (169)	749 (83)	100 (24)	2157
way	241 (77)	1289 (104)	294 (37)	1824
prince	1043 (164)	448 (81)	108 (24)	1599
girl	764 (135)	452 (69)	347 (36)	1563
thing	468 (86)	491 (79)	428 (70)	1387
wife	545 (111)	732 (80)	106 (30)	1383
head	261 (80)	1055 (115)	66 (34)	1382
princess	746 (154)	487 (97)	147 (38)	1380

Table 3.4: Properties of relationships for the most frequent entities in the fairy tale corpus. The number of occurrences for each relationship is reported with the number of tokens it relates to shown in parentheses.

man and *woman*, which appear as referring terms for many entities.[11]

Tables 3.4 and 3.5 also show the number and types of relationships for each of these most frequent word. For example, the entity *man* appears 2,223 times as the subject of 204 verbs, and the verb *be* appears 5,889 times with 664 different subjects. If we were to allow unrestricted search when building a sentence with *man* as the subject then we would need to consider all of these verbs. Assuming that the first verb was *be* and we were using the *NPPP* template (consisting of a verb, a subject, a direct object and an indirect, for example, *the man is the king of the country*), then we would also have to consider all combinations of objects attested with *be*. We see from Table 3.5 that there are 650 objects, resulting in 422,500 (650*650) different sentences. Considering that this is just a single sentence within the story search space, a complete search would clearly make the system non-viable.

Looking at specific entities within the database we can see how well the MI scores capture the stereotypical relationships within the corpus. Table 3.6 shows the top and bottom subject, object and adjective relationships for the entities *king*, *prince* and *princess*. From the top scoring subject and object relationships we see some interesting verbs appearing. Many of these are expected e.g., *kings sign*, *princes woo* and

[11] We did not undertake coreference resolution before processing the corpus so these references now appear as entities within their own right, throughout the corpus.

Verb	Subject	Object	Adverb	Total
be	5889 (664)	9355 (650)	2524 (75)	17768
go	1158 (253)	2781 (320)	1437 (55)	5376
take	811 (232)	3179 (422)	575 (64)	4565
have	833 (236)	3099 (430)	591 (63)	4523
come	1527 (295)	1919 (381)	1042 (62)	4488
say	1115 (198)	2198 (260)	658 (59)	3971
see	823 (208)	1971 (443)	1078 (50)	3872
give	647 (233)	2213 (375)	343 (47)	3203
make	712 (283)	1780 (345)	389 (65)	2881
find	362 (170)	1427 (359)	544 (55)	2333

Table 3.5: Properties of relationships for the most frequent verbs in the fairy tale corpus. The number of occurrences for each relationship is reported with the number of tokens it relates to shown in parentheses.

princesses are married. There are also some high scoring relationships that are surprising, for example, *kings spit* and *princes wet*, actions we do not strongly associate with fairytale *princes* and *kings*. Furthermore, note that low scoring actions are applicable to many entities in the corpus, e.g., *think*, *say* and *do*. These actions, say little about any particular character that takes part in a story, however they are potentially important for generating fluent and human-like stories. Our search procedure gives preference to sentences with high scoring relationships, as they are more expected of the character. However, other verbs can also be used when selected by the action graph based on the discourse history. The action graph trained on the fairy tale corpus containing 23,109 nodes with an average of 1.832 (*SD* 4.343) outgoing edges per node. There were 1,012 bigram (e.g., *subj:run subj:catch*) relationships and 22,097 trigram relationships (e.g., *subj:run subj:catch subj:hold*). Several examples of action graph entries are shown in Table 3.7, each example shows the actions that follow a particular node, with the MI score for each transition. In Table 3.7 we see what if the previous actions performed by an entity were to *hide* and then *wait* then it is expected that the next action would be either to *watch*, *go* or *run*.

Noun	Subject	Object	Adjective
king	empty (3.4)	report (3.6)	powerful (4.0)
	exchange (3.4)	improve (3.3)	astonished (3.6)
	spit (2.8)	colour (3.1)	false (3.3)
	woo (2.8)	reign (3.1)	silly (3.1)
	sign (2.5)	sob (2.9)	angry (2.7)
	...	...	...
	force (0.12)	step (0.2)	rich (0.56)
	hold (0.089)	raise (0.14)	great (0.53)
	do (0.06)	bear (0.09)	wicked (0.43)
	lose (0.02)	help (0.02)	full (0.37)
	lead (0.01)	happen (0.01)	strange (0.11)
prince	court (3.7)	attack (3.5)	unfortunate (3.6)
	wet (3.7)	waken (3.2)	instant (3.6)
	station (3.0)	sign (3.0)	gracious (3.5)
	woo (3.0)	praise (2.9)	beloved (3.1)
	equal (3.0)	suck (2.8)	pale (2.8)
	...	...	...
	step (0.05)	leave (0.09)	beautiful (1.0)
	say (0.03)	stay (0.07)	other (0.66)
	jump (0.03)	see (0.05)	new (0.60)
	save (0.02)	live (0.02)	dead (0.43)
	pay (0.02)	order (0.02)	splendid (0.20)
princess	study (4.0)	report (4.1)	proud (3.8)
	tease (3.3)	joke (4.1)	beloved (3.5)
	report (2.9)	sway (4.1)	glad (3.3)
	express (2.9)	rejoin (3.5)	vain (3.5)
	race (2.9)	marry (3.1)	younger (3.1)
	...	...	...
	want (0.05)	behold (0.07)	rich (0.26)
	think (0.04)	wait (0.06)	young (0.17)
	return (0.03)	fetch (0.04)	dead (0.12)
	begin (0.02)	leave (0.01)	full (0.06)
	laugh (0.01)	pay (0.01)	wonderful (0.6)

Table 3.6: Example of highest and lowest scoring relationships (with MI scores shown in parentheses) for commonly occurring entities in the fairy tale corpus.

Preceding action(s)	Following actions	
	subject	object
subject:fly	buzz (5.1) peck (5.1) tap (5.1) scratch (4.4) flutter (4.4) swing (4.4) leap (4.2) drop (4.0) snatch (4.0) cure (3.5) pass (2.6) eat (2.4) leave (2.3) throw (2.0) begin (1.8) take (1.1)	lock (4.4) offer (4.4) disappear (4.0) wait (3.3) make (3.0) give (2.0) ask (2.0) see (1.9) look (1.6) like (1.6)
object:marry	obey (4.6) write (4.6) love (3.9) happen (3.4) eat (2.9) return (2.6) pass (2.5) sit (2.4) tell (2.1) go (2.0) live (1.8) have (1.7) see (1.6) begin (1.5) take (1.5) grow (1.2)	fetch (4.4) win (4.3) behold (4.2) order (3.8) save (3.6) seek (3.3) look (2.4) put (2.2) give (1.5)
subject:hide subject:wait	watch (4.0) go (3.1) run (2.6)	serve (7.2)

Table 3.7: Example of entries in the action graph constructed from the fairy tale corpus. For a set of preceding actions, the expected next actions are reported with their MI scores in parentheses. Each action is labelled with whether the entity is the subject or the object.

3.4.3 Parameter Tuning

There are several parameters that we have thus far eluded to but not set. The coherence model in Section 3.3.3 can be trained on either bigrams or trigrams of entity transitions. Also, as we are using two separate ranking models (interest and local coherence) we must decide in which order they should be applied to the generated stories. To select the best parameter setting for a full human participant study, we must first investigate the effects that these parameters have on the stories being generated.

In training the entity grid model for evaluating local coherence we can choose whether to look at bigrams or trigrams of entity transition sequences. Moving from bigrams to trigrams increases the number of different sequence types, and therefore the length of the vectors increase, as we have a vocabulary of size 4 (S, O, X and –) the length of the bigram vector is $4^2 = 16$ and the trigram vector $4^3 = 64$. It does however, allow the model to consider a larger window of transitions from the discourse history. Each model is trained using SVM^{light} (Joachims, 2002). Each vector is paired with a real number value representing the document's perceived coherence. The training vectors were created by taking the original document and permuting the sentences, under the assumption that the original document is coherent and that any permutations of sentence order will lead to a less coherent document. Each vector is scored using Kendall's Tau (Lapata, 2003) which indicates how similar the sentence order of the permutation is to the original sentence order. The input to the SVM is thus the documents vector and its Kendall's Tau score.

For each document we randomly selected pairs of permutation, in which one would would have a higher tau than the other, to be used as training examples. We trained both models on 200 documents of the fairy tale corpus and evaluated on 40. Evaluation was performed by recording the number of vector pairs the models could correctly rank. The bigram model correctly ranked the pairs 0.91% of the time and the trigram model 0.96%. Although this would seem to indicate that the trigram model is better, it should also be noted that the fairy tales are much longer and contain many more entities than the stories our system produces. We therefore propose that the performance of both the bigram and trigram models requires to be evaluated on stories generated by the system.

Another consideration is how to apply our two ranking models of interest and local coherence. One approach is to apply them both simultaneously using a simple voting method to achieve a combined ranking (see Algorithm B.5 in Appendix B). Another alternative is to rank with the coherence model first and then the interest model (or

vice-versa). The stories produced by the system would then vary as we change the priority of what we are looking for in the story. So, we could first remove stories that are not coherent before we decide which of the coherent stories are interesting. Each of these approaches will produce different types of story depending on their focus.

We ran the generate-and-rank system with the different coherence models described above and then varied the order in which interest and coherence models were applied. We used 12 input sentences with 2 coherence models and 3 different methods for applying the coherence and interest models (coherence and interest simultaneously, coherence first then interest, interest first then coherence) which gave in total 72 stories. Two human judges rated these stories on a scale of 1 to 5 in terms of both coherence and interest. It was found that the best setting was to apply the coherence and interest models simultaneously using entity transition sequences of length 2. A shorter chain of entity transition sequences does makes sense as our stories are short with a few entities. Applying the models in succession yielded worse stories. The entity-grid model tended to maximise coherence by selecting only short sentences with very few entities, leading to few lexical items for the interest model to evaluate. The interest model, on the other hand, tended to select stories with large sentences in which numerous one-off entities appeared resulting in less coherent texts.

3.4.4 Evaluation

We compared the generate-and-rank system (Rank-based) against two baseline systems. Both systems do not use a beam. Instead, they decide deterministically how to generate a story. The first baseline, Best-selection, does so on the basis of the predicate-argument and predicate-predicate relationships in the knowledge base, it will always select the relationship with the strongest MI scores. Only a single story is created, although the search procedure has the ability to backtrack if a dead-end is reached before the story is of the desired length. The second baseline, Random-selection, uses a uniform distribution when making decisions ignoring the co-occurrence frequencies. Again, only one story hypothesis is considered whilst generating.

The three systems 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

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

Table 3.8: Human evaluation results: mean story ratings for three versions of our system; *: significantly different from Rank-based (* : $p < 0.05$).

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?). These criteria were selected as they covered fundamental aspects of our generated stories, i.e., they must make sense and be interesting, whilst being represented adequately in the text. There are other evaluation measures we could have considered. For example, Callaway and Lester (2002a) asked participants to judge the level of detail and the believability of the produced stories when evaluating the output of their story generator STORYBOOK. However, as their focus was on the production of high quality prose rather on how the story itself was generated,[12] the level of detail in the stories produced were more complex, requiring more granularity during evaluation. In particular, they were interested in the effects of microplanning and surface level features on the perceived quality of the generated texts. Our evaluation is therefore much closer to that of Peinado and Gervás (2006) who asked participants to evaluate stories produced by PROTOPROPP against randomly generated stories and stories from the original corpus. Participants rated each story for linguistic quality, coherence, interest and originality.

The experiment was conducted using theWEBEXP[13] (Keller et al., 2009) experimental software over the internet and the stories were presented in random order. Participants were told that all stories were generated by a computer program and were instructed to rate more favourably interesting stories, stories that were comprehensible and overall grammatical.

[12] STORYBOOK assumes the existence of a narrative generator that produces a story specifications. For their evaluation, Callaway and Lester, 2002a used a primitive narrative planner, capable of producing two different Little Red Riding Hood stories of two to three pages in length.

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

3.4.5 Results

Our results are summarised in Table 3.8 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 3.8 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 best-selection and random-selection systems ($\alpha < 0.05$). The best-selection system is not significantly better than random-selection system except in terms of coherence.

These results are not entirely surprising. In contrast to the baselines, the rank-based system assesses candidate stories more globally, thus favouring more coherent stories, with varied word choice and structure. The best-selection 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 favour 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 short sentences generated by this system seems also to confirm that sentence length can be taken to be a significant predictor of story interest. The random-selection system uses even less information in generating a story (entity-action relationships are chosen at random without taking note of the MI scores). A note of caution here concerns referring expressions which our systems did not generate. This may have disadvantaged the stories overall, rendering them stylistically awkward.

Table 3.9 shows example output from the three systems. As can be seen, the stories created by the generate-and-rank system include some unexpected lexical choices. The sentence, *The giant cries that the son laughs the happiness out of death*, presents several oddities. The first is the addition of entities such as *happiness* and *death*, which are rather dramatic for a simple children's story. These words are highly favoured by the interest model, and it is true that these words can be thought of as interesting. Although interesting on their own, these words do not relate to the story's context, and the interest model has no way of knowing this. As a result, the actions and entities favoured by the interest model seem surreal and require considerable effort on the reader's part in

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.
Best-selection	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.
Random-selection	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.

Table 3.9: Stories generated by our generate-and-rank system (Rank-based), a system that follows the highest scoring path through the search space (Best-selection), the a system that makes choices at random (Random-selection).

trying to construct a mental picture of what is said. The sentence above requires us to imagine a situation in which the *giant* is commenting on the fact that an additional entity, the *son*, is gaining joy by laughing at a life threatening situation. The composition of the sentence is needlessly complicated and it is a concern that the target audience, young children, will not be able to understand it.

One issue that needs to be addressed is the quality of the text of the realised story. Looking at the examples above, we see that there are some rather strange arrangements in word order, for example, *The child longer hides the forest*. These problems appear as the surface realiser tries to pick an appropriate place for the adverb in the sentence. We can also see that the sentence is missing a preposition, this means that the realiser has selected the wrong sentence structure. This example also suffers from the fact that the system does not have any mechanism for coping with negation (there is currently no mechanism for deciding whether a verbs polarity is positive or negative), so the sentence, *The child no longer hides in the forest*, cannot currently be generated. In addition, as the surface realiser scores each possible realisation with a trigram language model it fails to capture long range dependencies in the text and tends to favour sentences that contain commonly occurring groups of words. Other approaches to language modelling are available that are not constrained by such short word sequences (see Mitchell and Lapata, 2009 and the references therein).

Another consideration is the efficiency and coverage of the generate-and-rank approach. As we have shown, the space of possible stories can be incredibly large and as such we placed strict limits on the number of candidate stories we consider. Specifically, we set a limit on the number of lexemes considered for a sentence node and the number of sentence templates considered for a particular verb. Unfortunately, by reducing the search space in this manner, we are missing out on some optimal solutions. When selecting which lexical items to use in the story, the system utilises the MI scores to rank and select the top N words. The lexemes selected have a strong relationship with the words they are connected to in the sentence's dependency graph, however, viewing the document as a whole, it is not clear that they would make the best choices overall. Interestingly, we find that the stories generated by the random-selection system include some very good sentences, e.g., *The baby vanishes into the cave, The family meets with a stranger*. This would seem to suggest that there are interesting sentence structures the generate-and-rank approach simply cannot reach. We will discuss alternative search methods in Chapter 5.

Each sentence we generate has a strong dependency on the sentences that preceded

it in the story. As we explained earlier, a discourse history is maintained in order to steer the selection of entities towards those that have already appeared, and to encourage a logical action progression sequence. The downside of this approach is that we end up with a large amount of repetition in the sentences being generated. The same sentences are generated over and over again for inclusion at different positions in the story search tree. This leaves us with a lot of wastage in terms of run-time. We clearly need a more efficient way of searching the story space.

The system presented in this chapter covers the content selection, sentence planning and surface realisation tasks of Reiter and Dale's (2000) NLG pipeline. Without, however, incorporating any document planning. Each story was built one sentence at a time with each new sentence being dependent on those that came before it. Choices made early on in the creation of a story are, therefore, immutable and affect the generation of the rest of the story, perhaps leading to a story which is not the best story overall. Due to the high overhead associated with searching the story space we can assume that without proper planning before search begins, a large percentage of the solutions generated will be fruitless. By incorporating document planning, we can direct the system towards areas of the search space that are more likely to result in profitable solutions. In the next chapter we will propose a method for document planning through the use of story plots and show how these plots can be automatically extracted from the corpus and used to generate a wide range of stories.

3.5 Summary of Chapter

In this chapter 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 realises 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 generate-and-rank approach advocated here is viable in producing short stories that exhibit narrative structure. As our system can be easily retrained on different corpora, it can potentially generate stories that vary in vocabulary, style, genre, and

domain. In the next chapter we show how the issues with story structure, described above, can be improved using an explicit model of document planning, namely using story plots. Further, in Chapter 5 we will present an evolutionary search procedure for generating stories which uses the stories generated from plots as a basis. We show that by using evolutionary search methods we can overcome the limitations that were encountered with the generate-and-rank approach, specifically improving the exploration of the story search space.

Chapter 4

Plot-Based Story Generation

In the previous chapter we introduced a system that generated stories through the reformulation of knowledge about entities and actions acquired from corpora. Central to this approach was the idea that the building blocks of binary relationships between actions and entities could be combined in such a way that new sentences and subsequently new stories could be created. This approach, although providing a good starting point, suffered from the lack of document level control throughout the generation process. The latter manifested itself in several ways.

Firstly, the generate-and-rank search approach required a limit on the number of possible story candidates the system could consider due to the size of the search space. Each sentence within a story was generated from a template, each template contained lexical arguments to be filled using the knowledge from the corpus. It was necessary to limit the number of lexical items considered for these arguments (e.g., objects for the verb *protect* or adjectives to describe the noun *prince*), taking the *N* highest scoring candidates from the content selection database. However, as decisions on which entities to include are made based solely on the verb-entity relationships in the database, the system could not make selections that favour entities which are more likely to appear together. If we consider the entity *dog* then we see from the corpus that it is related with dog-like actions: *bark*, *bite* and *lick*. Selecting an action for which *dog* is the subject is a straightforward enough task. In contrast, though, suppose we are considering an object for the action *give*. The problem is that a large number of entities are associated with the action of *giving*. The system must select those entities which have the strongest relationship with the object role of the action. This means that no matter which entity appears as the subject, be it a *dog*, *king* or *castle*, it will find itself interacting with those same entities, which in the case of *give* are *helper*, *tug*, *shove*, *push*

and *kick*. It is clear that these lexical choice decisions cannot be made in isolation and require knowledge of entity co-occurrences. The generate-and-rank system attempted to reduce this problem by making use of a discourse history. However, this could only enforced the appearance of those entities that had already appeared in the current story. Conditional probabilities over verb arguments were also used but had little effect due to their sparsity in the corpus.

The system also struggled to enforce coherence as actions depicted in the stories were largely independent of one another. To improve coherence, verb choices were made based on likelihoods of action progressions (e.g., if someone *steals* then their next action will probably be *run* or *hide*). However, these action progressions were generic and did not take into account the expected action progressions of given entities. Can we really assume that all characters will react in the same way to a given situation? We therefore need to move away from a generic treatment of entity action sequences and attempt to bring in a level of character modelling. Lacking from the generate-and-rank system was the notion of how an entity behaves within the corpus, by itself and with others.

We hypothesise that these problems could be alleviated by generating stories from plots, which will allow document level control over the the generation process. As mentioned earlier, the generate-and-rank approach lacked direction when searching, this is unsurprising as it is trying to generate and complete the content of each sentence (i.e., clauses and their subjects, objects, adverbs and adjectives) at the same time, without the ability to consider how these choices affect the quality of the story in development with consideration future sentences.

In this chapter we will outline a story generator that employs a document planner to create and structure plots for stories. The planner is built automatically from a training corpus and creates plots dynamically depending on the protagonists of the story. Each plot is representative of a range of stories, the final stories are produced once the entities that are to appear in each story have been selected. Unlike the generate-and-rank system, the resulting stories will consist of action sequences that have been prescribed by the story protagonists. These actions will also contain information about which entities are likely to appear together to improve the interactions of entities in the story. From the set of stories prescribed by each plot, the system selects a single story to be presented to the user. We will start by describing how plots have been used in previous story generation work and natural language systems. And then we move on to describe an approach for extracting plots from corpora and show how these plots can be used to

generate stories.

4.1 Related Work

Plots are traditionally used in symbolic generation systems. Pemberton's (1989) GESTER system uses a grammar in order to generate story summaries of French epics. Grammars translate well into document structures, non-terminals indicate document sections whereas terminals are reserved for text. GESTER uses a grammar that was hand-coded from the analysis of nine poems composed in medieval France. The grammar defines a complex story which comprises one or more simple stories. A simple story consists of an initial situation, a series of active events and then a final situation. The events in each story consist of a motivation, plan, qualification, action and resolution. The grammar also outlines how simple stories and events relate to one another. For example, simple stories may have a *'cause'* relationship between them, the first story causes the second, or a *'motive'* relationship where an action in the first story is the motive for the second.

Once a plot has been generated from the grammar, it still represents only an abstract version of a story and require satisfaction of variables, selecting characters and events from the predefined epic story world. The placement of characters relies on the relationships between them, i.e., whether two characters are married or whether one character owns a particular entity or not. An attempt at maintaining coherence is made by assigning the same roles to characters throughout each story. The result of variable satisfaction is the final story, which Pemberton (1989) refers to as the *story line*. In particular, the distinction is made between the *story line* and the *story discourse* (comparable to the story *fabula* and *sujet* described by Propp, 1968). The story line contains all the actions performed by the characters in the story, yet the order in which actions appear in the final text may not necessarily be in the story line's chronological order. Some events in the story line may also be omitted if they are implied by other events. For example, for the story line *Simon is hungry, Simon has chicken, Simon eats chicken*, we would not necessarily have to add in the final text that he has the chicken, as the fact he is eating it already implies this. In particular, Pemberton (1989) typically removes the initial situation, the resolution and the final situation as these can be inferred from the events that take place in the story.

Changing the order of events between the story line and the story discourse occurs when a given story entity takes part in more than one event at the same time. Pemberton

(1989) gives an example from their corpus where the character will catch sight of their future wife during the conquest of a city. These distinct events although occurring chronologically at the same time will not appear together in the final text, as the focus should be maintained on one objective at a time. The output of the system is a story summary in a canned English-like representation.

Although not implemented in GESTER, Pemberton (1989) describes how the generation of the document structure should take into consideration, who the author and audience are. These factors will in part define the communication goal of the document being generated, for example, is this a story by an adult teacher conveying a moral to a child, or perhaps a light-hearted tale simply intended to entertain a 21st century adult reader. Solis et al. (2009) model the author and audience explicitly in their system, PICTUREBOOKS. The latter generates stories for children aged 4 to 6 that portray a moral situation, similar to that of a fable; it is interactive, the user starts by selecting a background setting which appears on the screen as a picture. From a prescribed list of pictures of objects and characters, the user then selects the main entities to appear in the story and places them in the background picture. Using this background and entities therein, the theme of the story and the events that will occur in it can be decided. For example, the themes associated with the *bedroom* background include *being neat* and *going to sleep early*.

The PICTUREBOOKS system database contains information on 11 different themes. Each theme comprises four distinct sections that are described by Machado (2003) as fundamental stages for a story plot. These are: an introduction to the problem, a rising action and plot development, an insightful answer of solution and a climactic scene. A theme is selected that contains applicable entities closely matching those the user placed in the background image. A grammar then defines how this theme can be expanded to create a text. The terminals of this grammar are the actions that will represent sentence clauses in the final story text. During the generation process, the rules regarding how the theme is developed will include arguments that require satisfaction, such as the objects used by the characters in the story. In particular, Solis et al. (2009) make use of a semantic ontology which they develop from ConceptNet (Liu and Davenport, 2004). This allows them to increase the number of applicable events and entities in a story by searching for those that are semantically related. The abstract story tree represented by the story plan and its argument satisfaction is finally realised using the *simpleng* realiser (Gatt and Reiter, 2009).

The realisation of the story differs depending on the age of the user, allowing for

simple audience modelling. A story generated for a four year old will contain simpler language, to represent the events in the story plan, than that of a story for six year old, e.g., *Sara was sad* rather than *Sara was upset*. The complexity of the final story can also be regulated by removing supplementary information, i.e., terminals of the story plot that do not represent key events, in order to shorten the story of younger readers.

Gervás et al. (2004) create stories by manipulating a database of plots manually constructed from the story morphemes identified by Propp (1968). Each plot represents a single story in the corpus of Russian folk tales on which Propp based his work. Gervás et al. (2004) construct an ontology of Propp's characters and recurring plot units, including the dependencies between them. A user can then stipulate which characters and events (only those identified by Propp) they wish to appear in a generated story. Retrieving a plot that contains all the requirements of the user is unlikely however, as they are free to select characters and plot elements from the entire corpus of stories. Plots are therefore adapted using the dependencies in the ontology. For example, their system substitutes the act of *Murder* for *Kidnapping* (which is in the original story) as they are both *Villainy* events. New plots can be created by adapting those in the database, using knowledge of how plot events relate.

Many other approaches to story generation rely heavily on planning to elicit a story (Meehan, 1977; Riedl and Young, 2004; Swartjes and Theune, 2008). They typically utilise a story world, defining the genre, along with an initial world state and a goal state representing the desired outcome of the story. Planning rules are then applied in order to find a sequence of events that will change the initial state of the story world into that of the goal state. A story results from the translation of the planning rules that have been applied into natural language. As with the work of Pemberton (1989), the order in which these story events are presented need not be the same as the order they were applied (see Riedl and Young, 2006b). Systems such as TALE-SPIN (Meehan, 1977), require only that facts and relationships about the world be stored and manipulated through planning rules. However, more recent work uses autonomous agents to represent characters in the story, which carry out their own planning (Swartjes and Theune, 2008). The overall story plot is a set of world goals, with agents that are concerned with their own particular goals throughout the story generation process. This leads to more realistic story events as the story characters are absorbed in meeting their own goals. However, this carries the risk of the system never reaching the final goal state which is required for the story plot. It is the job of the *director agent*, a special type of story agent that manages the story world as a whole, to supply the

character agents with their goals and manipulate the story environment to ensure that the final goal state is reached.

Riedl and Young (2006b) also employ a director agent as their planning system has a new and difficult element to contend with, the interactive human player. The generation of plots for agent-based systems becomes more complicated when the user interacts with the story agents. This is an advantageous area for integrating story generation into computer games. The human interacting with a story must have the option to explore the system they are participating in, they need to feel they are free to act on their own accord rather than simply play a part in a pre-written script. The FAÇADE system (Riedl and Young, 2006b) is a prime example. Here, the human player can participate in a story scenario. One example is that of a bank robbery. The user is free to perform their own actions, affecting how the scenario is ultimately played out. In this example, the bank manager is required to open the safe with their key before the human can steal the contents. However, by stealing the key from the bank manager before he unlocks the safe, the system must re-plan the key events of the story in order to reach the desired conclusion. The director agent therefore has to plan a sequence of actions to acquire back the key for the bank manager so that the scenario can proceed along its plot. In this case we can see the system uses a high level plot involving a set of key events, but the fine detail of its execution is left unspecified so as not to constrain the actions of the human player.

In most agent-based systems, agents must attain multiple goals. An exception is TALE-SPIN (Meehan, 1977) where story generation focuses on a single goal: the characters are thirsty and must find a solution to this problem. Multiple goal states can be used to create more intricate plots. Theune et al. (2004) view a story as the result of generating a series of *episodes* which outline the story's plot. An episode database is used in which each episode script contains information on setting, goals and constraints, once again derived from Propp (1968). A story plot is created by allowing users to select a number of episode scripts from the database. Although the desired outcomes for each episode are prescribed in the plot, the manner in which they are achieved is left to the system. Each episode indicates which agents should be included and what their goals are, and the final story results from their attempts to satisfy these goals.

Each of these approaches to story generation, rely on hand-coded knowledge bases, whether in terms of the story world and the applicable actions or a grammar describing how to structure a story text. This reliance reduces the number of plots available to

these story generation systems. Plan-based story generation systems not only require a carefully constructed database representing the story domain, but also knowledge of the initial and goal states for each story plot. None of the story generation systems mentioned above scale particularly well. The development of each is laborious and time consuming as the process must be repeated for any new characters, plots and domains. We propose that story plots need not be hand-written, but rather can be extracted automatically from corpora. In particular, we look at producing story plots aimed at young children, as in Solis et al. (2009). However, we shall refrain from explicitly modelling the author or the audience, instead relying on the information inherent in corpora to guide the generation process.

Our own plots are reminiscent of the narrative schemas introduced by Chambers and Jurafsky (2009). Their goal is to automatically extract from corpora representations of situations akin to those found in FrameNet (Fillmore et al., 2003). To achieve this they construct schemas consisting of semantically similar predicates and the entities evoked by them. Events within a corpus are clustered into *narrative chains* by comparing actions with a shared protagonist. These chains are then merged with those of other entities appearing in the corpus to create *narrative schemas*. 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*}. These schemas represent events and entities that strongly associate within the corpus. Our narrative schemas differ slightly from Chambers and Jurafsky (2009). In our setting, every entity has its own schema, and predicates associated with it are ordered. Plots are generated by merging together the entity-specific narrative schemas.

Our plots bear some similarity to those utilised by Halpin and Moore (2006) who outline a computer agent, as part of an interactive tutoring system, that gives personalised feedback to children partaking in a story rewriting task. Specifically, children are asked to write from memory in their own words a story they have recently been told. The aim is to asses how well the children understood the original story. To do this, the agent compares the plot of the story written by the child against that of the original story. Halpin and Moore (2006) define plots as a series of linear events appearing in written text. The events are represented as predicate argument structures with verbs as predicates and nouns as the arguments (e.g., *become(boy, elf)*). A major hurdle to the extraction process was that the text produced by children is of a low quality. For each retelling, the plot was extracted and compared against a gold standard plot from the story told to the children. By comparing the proportion of overlapping events a grade

(excellent, good, fair or poor) can be selected, indicating how well the student understood the story. In addition, a method is outlined for providing personalised feedback, in particular, whether significant plot events are missing from their retelling or are in the different order from the original story. Significant plot items are identified as those which occurred in the majority of stories that were rated as excellent. Rather than building a plot of any one particular story, our aim is to create new plots by examining the action sequences of entities throughout the entire corpus. Our plots, unlike those of Halpin and Moore (2006), focus only on the protagonists, the main story characters, rather than enumerating all entities and actions within a document.

We will now outline our approach for extracting plots automatically from corpora and show how they can be used to generate stories.

4.2 Entity Graph Extraction

In order to create a plot we must first obtain information about the entities that will appear in the story, the actions they participate in, their ordering and finally how the entities can interact. Information about each entity is represented as a directed graph which we explain below. Throughout this thesis we have emphasised the virtues of extracting knowledge automatically from corpora. Along the same lines, we will outline an approach for building entity-specific schemas, which we shall call *entity graphs*, from knowledge inherent in corpora.

Our algorithm processes each document in the corpus, one at a time, operates over dependency structures and assumes that entity mentions have been resolved.[1] Documents are processed with RASP (Briscoe and Carroll, 2002), a broad coverage dependency parser, and the OPENNLP[2] coreference resolution engine. However, any dependency parser or coreference tool could serve our purpose. In addition, we assume that entities have been sense disambiguated. Our sense disambiguation method is relatively naïve, we default to the first sense in WordNet (Fellbaum, 1998). Each document in the corpus is processed in turn and from the dependency trees, events are identified along with their subjects and objects. For each entity in the text we now have a chain of events in which that entity was present. We assume that linear order corresponds to the temporal order. This is a gross simplification as it is well known that temporal relationships between events are not limited to precedence, they may

[1] Following from Chapter 3 entities are single word token nouns, and actions are lemmatised verbs.
[2] See `http://opennlp.sourceforge.net/`

The dragon holds the princess in a cave. The prince slays the dragon. The princess loves the prince. The kingdom rejoices. The prince asks the king's permission. The prince marries the princess in the temple. She has a baby.

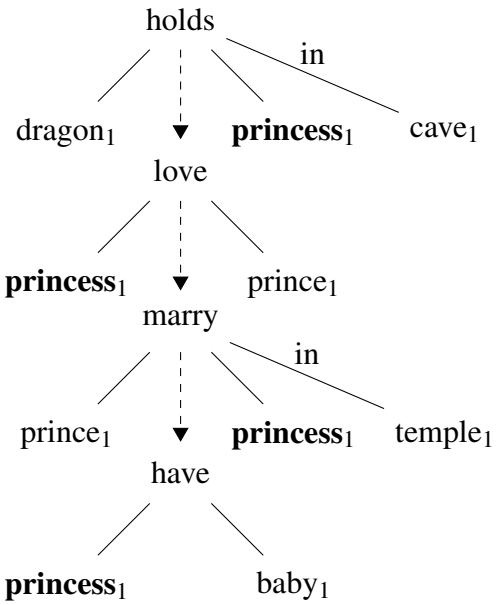


Figure 4.1: Example of graph construction for the entity *princess*. Dashed arrows represent directed edges between nodes. Each entity is shown with a subscript indicating its sense index in WordNet.

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. Each edge E is also assigned a weight w to represent the likelihood that a transition from the previous node will follow this edge. We discuss how weights are calculated and used in Section 4.2.1. In Figures 4.1 and 4.2 we give an example of how the graph for the entity *princess*, is constructed from a hypothetical corpus consisting of two documents. For simplicity we have omitted the weights on the graph's edges and used bold font to highlight the graph's topic entity. From the first document in Figure 4.1 a graph for *princess* is extracted with each node in the graph corresponding to an action with which *princess* is attested. As a result of coreference resolution, the

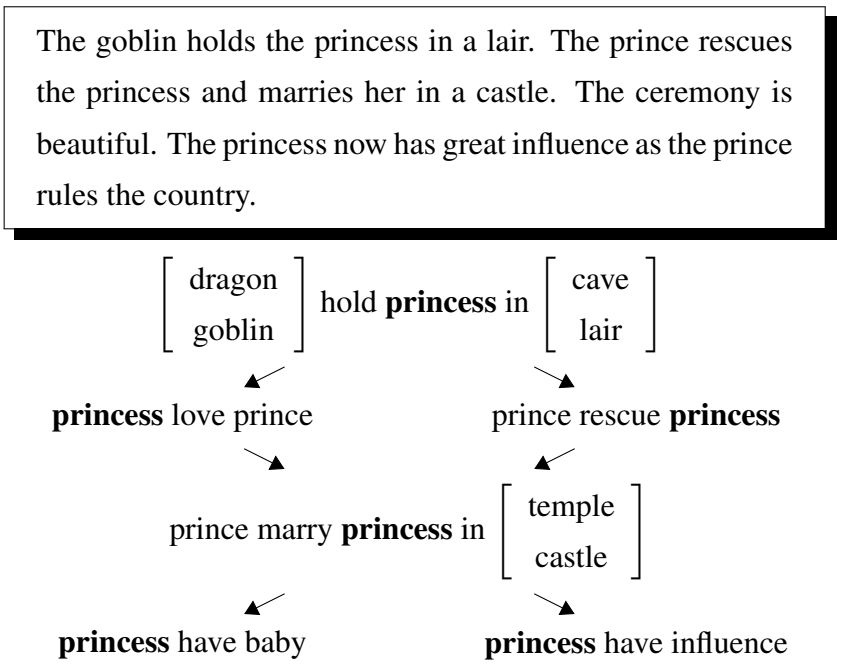


Figure 4.2: Example of an entity graph constructed for *princess* from two documents.

word forms *she* and *her* are recorded as *princess*. We also record the senses of any other entities participating in the action. WordNet sense indices are shown with each entity in the example, as we always take the first sense these will all be 1 and as such are omitted from the following examples. Each action is represented as a dependency tree which we show in Figure 4.1 but linearise for simplicity in the following examples. It should be clear that not every sentence in a document will be associated with each entity. Sentences may contain multiple events resulting in many more nodes. The first sentence in the text gives rise to the first node in the graph, the third sentence to the second node, and so on. Note that the second sentence is not present in the graph as it is not about the *princess*. Each node contains only the lemmatised dependency information as shown in Figure 4.1, grammatical information in the original text (e.g., verb tense and noun number, prepositions and particles) is discarded[3] and only reassigned during surface realisation (see Section 4.2.2; prepositions and particles are shown in the examples for clarity).

As mentioned above, entities are assigned the first sense in WordNet. The reason

[3] As with the development of the content selection database in Section 3.2.2, our focus is on building a model of the relationships between entities and actions. By removing prepositions and particles we reduce the number of possible actions that entities can be associated with, decreasing the number of low frequency events. Grammatical structure can later be reconstructed by the sentence planner and surface realisation modules.

we do not simply record the entity word forms is that we do not want to constrain, at this early stage, the number of possible lexical items that the story generator could use when creating a story from a plot. By recording sense information we defer the decisions with regards to which entities will appear in the final story. Using the first WordNet sense is admittedly a naïve approach and will not always assign the correct senses in the context of the text being processed. Works of fiction, such as those in our fairy tale corpus may lead to problems as the senses which appear will not necessarily have the same distributions as those in non-fiction documents, which WordNet is more apt to represent. Another issue is that co-reference resolution is not error-free (especially on fictional texts as most systems are trained on news text) and may leave pronouns unresolved. Rather than discarding actions with pronouns or any of their argument slots, we introduce two dummy senses, *object* and *person*. We use the *object* sense when encountering inanimate pronouns and *person* otherwise. Interestingly, fairy tales (as well as some other genres of fiction) rely heavily on anthropomorphism. This means that some story entities, although not actually being *people*, can still be seen to perform actions that would indicate they were. An example of this is the fairy tale "Puss in Boots" where the main character performs the acts of *speaking* and *wearing clothes*, although being a *cat*. For example, if the action *princess greet puss* appeared in the corpus as *princess greet him*, this would be recorded in our graph as *princess greet person*. When generating a story from a plot containing this node, the generator will look for an entity to fill the argument *person*, possibly leading to the action *princess greet prince*.

The entity graph is extended whenever a new sequence of associated events is found in a new document. In our example, the second document (see Figure 4.2) also contains a series of events in which the *princess* participates. Our goal when building an entity graph is not to simply record a collection of event chains as this would mean the graph would only encode stories it had seen before, hindering creativity. By creating a global entity graph, new action progressions can lead to entirely new stories. This practically means merging graphs together that have at least one node in common. Before adding a new node to the entity graph we compare it with the nodes currently in the graph to see if a similar node already exists. Nodes are merged only if they have the same verb and similar arguments, with the graph entity (i.e., *princess*) appearing in the same argument slot. In our example the nodes *goblin hold* ***princess*** *in lair* and *dragon hold* ***princess*** *in cave* can be merged as they contain the same verb and number of similar arguments. Combining these into a single node results in two different actions that the

princess can participate in after being *held*, namely to be rescued by a prince or to love him.

We stated above that two nodes can merge only if they contain similar arguments. In our example *castle* and *temple* can be merged as they both represent a place in which the *princess* is married. Nodes within graphs can therefore represent lists of senses. Note that we did not merge the nodes *have baby* and *have influence* despite the fact that they are both associated with *princess* because they are semantically dissimilar. In sum, entities other than the entity for which the graph is created do not correspond to word forms. Rather, they are grouped into semantically coherent classes. The later allows us to create many different stories from the same plot. In our example, the *princess* could also get married in a *mansion*, *palace* or *cathedral* as long as the system considers them semantically similar to *castle* and *temple*.

In order for us to group semantically similar senses we need a method of measuring how related they are. There are many ways of doing this using WordNet (Fellbaum, 1998) or distributional similarity (e.g., Brody and Lapata, 2008, 2009). Several different sense similarity measures have been proposed based on WordNet and an outline and comparison for several of these are presented in Budanitsky and Hirst (2001). In order to judge similarity we will focus solely on the hypernym (x is a y) relationships. Figa and Tarau (2003) also use WordNet to create what they call *story projections* which generalised stories by substituting entities with their hypernyms. They argue that patterns in stories will become more apparent by viewing stories by the types of entities that occur within actions rather than the exact entity. These projections are intended to be used by story understanding systems to build an overview of what a story is about. It is obvious though that crucial information about the story will be lost if we carry this procedure too far, as all characters in a story can eventually be projected to being objects (the fact that objects perform actions on other objects does not really provide us with much information). Although they also looked at abstracting other word types than nouns, such as verbs, we will focus exclusively on the former.

To judge the similarity of two senses we adopted the measure in Wu and Palmer (1994). It provides an efficient online method of calculating similarity which the system requires when searching the story space. This measure uses the paths between senses in the WordNet hierarchy. The assumption is that two related nodes will both have a shared super-node on their paths from the root node. Although Wu and Palmer (1994) only applied their measure to the similarity of verbs, the structural aspect of the measure can easily be adapted to the ontology of nouns as well. We can assume that

two similar noun senses will share a common super-node in their hypernym trees. The similarity of two senses $S1$ and $S2$ is thus:

$$SenseSim(s1,s2) = \frac{2*n3}{n1+n2+2*n3} \tag{4.1}$$

Where sense $s3$ is a super-node of $s1$ and $s2$. $n1$ is the number of nodes on the path from $s1$ to $s3$, $n2$ is the number of nodes on the path from $s2$ to $s3$ and $n3$ is the number of nodes between the root and $s3$. Resnik (1999) reformulates this equation in terms of the depth of each sense, i.e., distance from the root of the taxonomy:

$$SenseSim(s1,s2) = \frac{2*d(s3)}{d(s1)+d(s2)} \tag{4.2}$$

Where $d(s3)$ is the depth of the maximally specific superclass $s3$, of $s1$ and $s2$, and $d(s1)$ and $d(s2)$ are the depths of $s1$ and $s2$ through $s3$. Our implementation uses Resnik's (1999) formulation as sense depths are computationally easier to calculate. An advantage of this measure is that it always returns a value of similarity between 0 and 1 (1 meaning the senses are identical and 0 completely dissimilar), allowing us to define a numerical limit for what the system considers similar. Through empirical observation we decided that a score of 0.6 or above could be considered similar. Although the above measure will never return a score of 0, as the WordNet root is at depth 1, a score of 0 is automatically given when two senses do not share a common hypernym. For example, the senses for *prince* and *hope* have no super-ordinate sense in common.

When considering a new node for merging, we may be comparing a single sense against a list of senses which have already been deemed similar. In our example we grouped together *castle* and *temple* but the next document may contain the same action with the sense *cathedral*. For merging to take place we stipulate that the new sense must be scored as similar against all the senses in the list. This is to ensure that there is no drift in the meaning of that particular action. In the event there is a list of senses to be evaluated against we take the average sense similarity score compared to each sense. However, it should be noted that the order in which the texts and actions are processed can affect the way these cliques of senses are formed, in turn affecting the final entity graph.

The graph construction algorithm terminates when graphs like the one shown in Figure 4.2 have been created for all entities in the corpus. We will next outline a story generation system that utilises the entity graphs to make document level decisions when generating. As with the generate-and-rank system, we will assume that

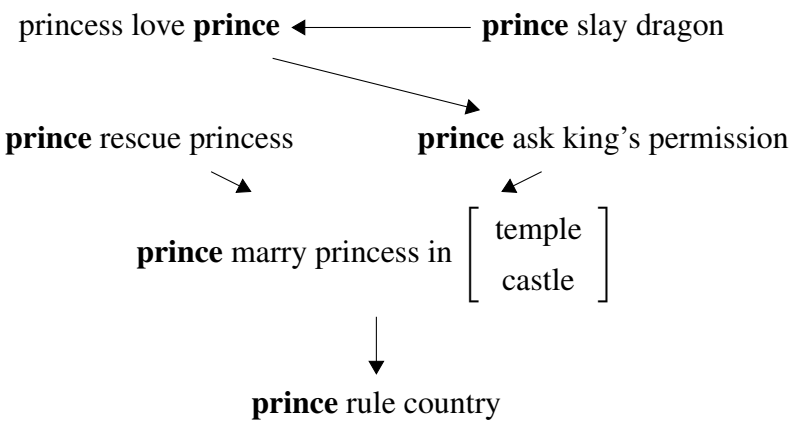


Figure 4.3: Example entity graph for the entity *prince*.

a user supplies the system with a sentence containing a couple of entities, the story protagonists. The goal of the system will then be to take this input sentence and use it to instantiate several plots. These plots will allow the system to search the space of possible stories more efficiently as the structure of the story will already be outlined within each plot. In addition to document planning, the system will make use of the content selection, sentence planning and surface realisation modules described in the Chapter 3.

4.2.1 Building a Story Plot

Our entity graphs effectively describe the characteristics of a single entity, its actions and interactions with other entities. However, we argue that for a story to be coherent we need to maximise the interaction between the story protagonists, i.e., the main characters in the story. In the previous chapter we explained how this could be achieved through the use of a discourse history. This, however, did not give preference to any particular entity, treating the story protagonists equally with all other entities in the story. Here, we create plots in which the focus is on the story protagonists supplied in the input sentence. For each of our protagonists we have an entity graph. We produce a plot by merging these together. We call the resulting graph a *plot graph*. As an example we will consider creating plots for the input sentence *the princess loves the prince* which requires the combination of the graphs for *princess* and *prince* shown in Figures 4.2 and 4.3 respectively to give the plot graph shown in Figure 4.4.

The process of merging two entity graphs is very similar to the way individual entity graphs are acquired. However, there are a couple of additional constraints on

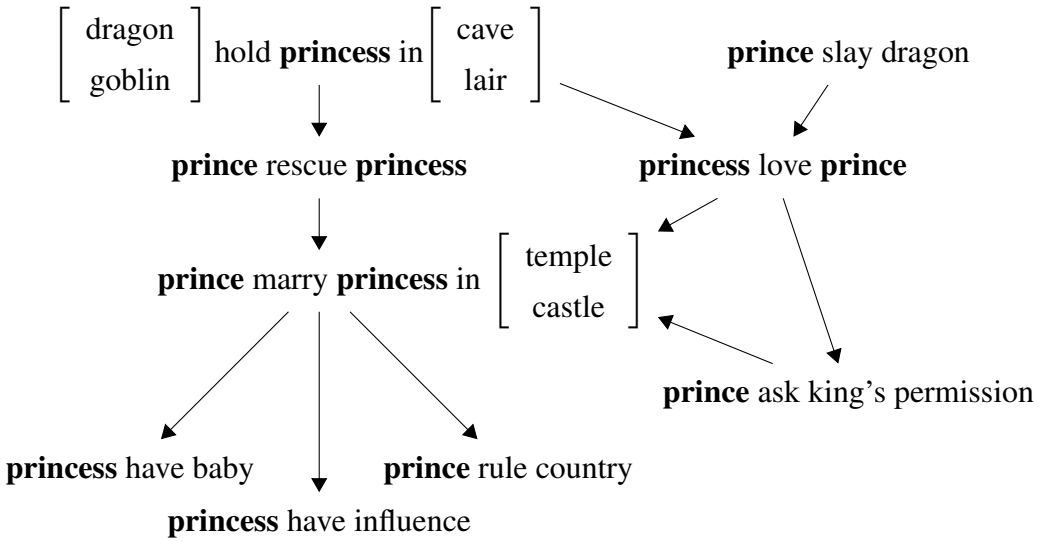


Figure 4.4: Example plot graph for the input sentence *the princess loves the prince*.

which nodes can be merged. First, we must be careful when merging nodes containing both protagonists, ensuring that they do not result in a node in which they assume the same role, (e.g., *[**prince**, **princess**] cries*). However, if these appear in different roles, then the resulting merger creates a new node while retaining the original nodes from both entity graphs. The entity senses that appear alongside the protagonists in these actions have the possibility of introducing additional entities to the story, so the system should be able to decide whether or not to introduce them when creating a plot. The newly created node maintains a focus on the protagonists which we expect to increase coherence. Consider the plot graph created from the two documents shown in Figure 4.5. Here, merging the nodes ***prince*** *dance with duchess* and *dauphin dance with* ***princess*** would result in a new node ***prince*** *dance with* ***princess*** which has the same edges as both of the original nodes. However, we also maintain the original nodes so that it is still possible for a *duchess* or a *dauphin* to appear in a story created from this plot. Essentially, this stops us from restricting the entities that can appear in our stories.

In addition to depicting order, the edges of the plot graph are weighted, indicating preference for particular action sequences. We would expect most actions to contain only a few possible transitions, however, it is easy to foresee that without proper weighting, commonly occurring actions (such as *get*, *be* and *have*) would become the predominant nodes for any entity, losing the characterisation the graphs are intended to capture. In keeping with the previous content selection module outlined in Sec-

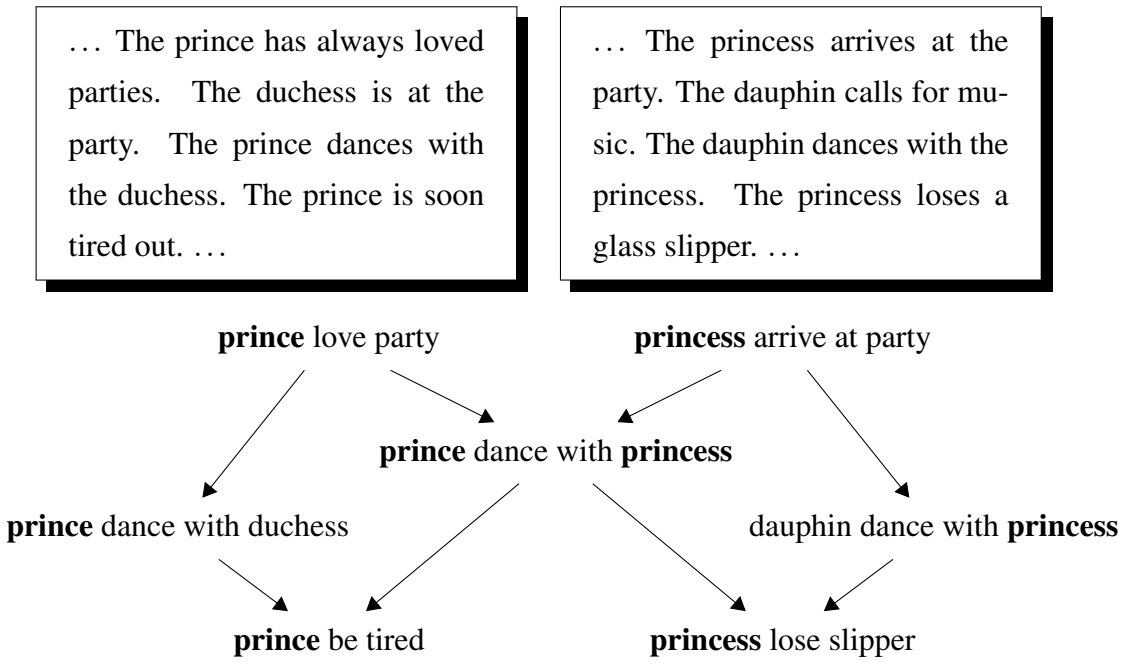


Figure 4.5: Example of node merging from two documents. The node ***prince*** *dance with* ***princess*** is created through the merging of the entity graphs for *prince* and *princess*.

tion 3.2.2, we weight these transitions with Mutual Information (MI, Lin, 1998) scores. These weights are easily calculated after the construction of the graph from counts of each nodes's occurrences in the corpus. Once again we are giving precedence to events that are more characteristic of the entities.

Once the plot graph has been created, a depth first search (see Corman et al., 2001, Section 22.3) is initiated from the node representing the action within the input sentence. The search locates all paths which will represent a story of the desired length, supplied as an input parameter (cycles are disallowed). We also disregard any nodes that do not have one of the two story protagonists as its subject (this is a stipulation made in the generate-and-rank system to increase coherence). The result of this search procedure is a collection of plots, representing underspecified stories. Assuming we wish to generate a story consisting of three sentences, the graph in Figure 4.4 would create four plots. These are (***princess*** *loves* ***prince***, ***prince*** *marry* ***princess*** *in [castle, temple],* ***princess*** *have* ***influence***), (***princess*** *loves* ***prince***, ***prince*** *marry* ***princess*** *in [castle, temple],* ***princess*** *have* ***baby***), (***princess*** *loves* ***prince***, ***prince*** *marry* ***princess*** *in [castle, temple],* ***prince*** *rule country*), and (***princess*** *loves* ***prince***, ***prince*** *ask king's permission,* ***prince*** *marry* ***princess*** *in [castle, temple]*).

What happens if our search reaches a dead end? This is a distinct possibility and

what is even more possible is that the input sentence, supplied by the user, will not match any of the nodes in the plot graph. Using the generate-and-rank system we can generate sentences that follow a given node, or the input sentence. We can thus obtain a set of sentences that are likely to follow these actions based on the content selection database. These generated sentences can then be compared against the nodes of the plot graph to see if they are represented. Once again we must ensure that we match a sentence to nodes with similar entity senses. Once a match has been found the search procedure can be continued from the identified node. By backing-off to the generate-and-rank approach we are allowing the story generator to use its knowledge of the corpus to infer one-off edges in the plot graph that it has not seen before, stopping the search from becoming trapped. We can expect that the sentences returned by the generate-and-rank system will feature actions strongly associated with the protagonists, highlighting advantageous nodes in the graph from which to continue building the plot.

4.2.2 Sentence Planning

The sentence planner is interleaved with the story planner and influences the final structure of each sentence in the story. Using the sentence planner described in Section 3.2.3 the actions depicted in the graph are given a structure that the story generation process can manipulate. The sentence planner uses a phrase structure grammar which can be represented as a tree in which nodes contain lexical items and edges represent dependencies. Lexical resources made available by Korhonen et al. (2006) and Grishman et al. (1994) indicate the rules applicable for a given verb. Grammar rules must contain the same number of noun arguments as there are entities in the action. The nodes in our entity graph are made up of entities and actions, i.e., nouns and verbs (in the example prepositions were added for clarity). However, a sentence is clearly going to contain more than just nouns and verbs. The rules outlined within the sentence planner, also allow for inserting adverbs, adjectives and prepositions. We describe how adverbs and adjectives are selected in Section 4.3. When generating a plot, we postpone, until surface realisation, the decision on how exactly the sentence should be rendered but it is crucial that we find which rules can be applied to the action. Each node contains one verb and then one to three nouns. A search of all grammar rules finds the rules suited to the action we are considering. This can also function as an early sanity check to ensure that the verb is allowed the number of arguments it appears with.

The nodes in our graph represent only a single action, corresponding to a single clause. The realisation of these nodes would therefore result in rather short sentences. As we discovered in the previous chapter (see Section 3.3.2), there is a significant correlation between interest and sentence length which was reflected in the low interest scores given to the Best-selection system which tended towards using the simplest sentence structures when generating stories. To avoid generating short sentences we combine pairs of nodes within the same graph by looking at intrasentential verb-verb co-occurrences in the training corpus. The grammar rules applicable for the matrix verb must contain a dependent subclause. 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. Also, we only turn an action into a subclause if its subject entity is same as that of the previous action.

4.3 Generating Stories from Plots

The plots that we generate are essentially schemas representing a large number of possible stories. Each node in the plot graph contains a list of sentence templates assigned by the sentence planner, each of which may contain lexical nodes that must be satisfied. These may be verbs, adjectives, adverbs or nouns (represented as senses). Using the content selection database described in Section 3.2.2 the most appropriate verbs, adverbs and adjectives are found based on the dependencies in the sentence (see Algorithm B.6 in Appendix B). The case for nouns is slightly different. All references to the protagonists are assigned the same word form used in the input sentence. For all other entities, the decision on which word form to select depends on the sense information. We know from the content selection database which entities are strongly suited to roles within an action (e.g., that *dogs bark* and *princesses are married*), based on their Mutual Information scores calculated from the corpus and use this information to select the word-forms for the final story. For example, consider again the node ***prince** marry **princess** in [castle, temple]*. The protagonists will be assigned the word forms *prince* and *princess*, however, we still need to select an appropriate word form for the final argument *[castle, temple]*. We need to select the word form from our co-occurrence database, but this may not contain either *church* or *temple*. As we discuss in Section 3.2.2 we discard all verb-noun occurrences that occur in the corpus less than a certain number of times. In the event that they are not present, we look for other

entities that co-occur with *marry* that have a similar sense to that of *temple* and *castle*. For example, these could include *princess*, *temple*, *haste*, *cathedral*, *house*, *mansion*, *queen*, *forest*. These candidate words can be ordered by calculating their similarity to the sense of *castle* and *temple*. This would then gives rise to the following ordered list: *temple*, *mansion*, *cathedral*, *house*, *forest*, *princess*, *queen* and *haste*. Senses deemed considerably dissimilar are then dropped, e.g., *princess*, *queen* and *haste*.

Note that, simply enumerating exhaustively all of the stories represented by a plot would be completely infeasible. A typical graph can contain hundreds of nodes which themselves represent thousands of sentences (especially for commonly occurring nouns such as *get*, *be* and *say*). Searching all combinations of sentences within a plot would lead to the generation of millions of stories. To get around this we limit the *branching factor* of the search. The MI scores on the edges of the plot graph allow us to prescribe an order for the children before we search them. As we are dealing with a depth first search we take the N best children that result in complete plots.

As well as limiting the number of plots we generate, we must also limit the number of lexical items for consideration in those arguments of the sentence templates not yet satisfied. We can therefore use the MI scores in the content selection database to limit the number of choices for consideration for verbs, adjectives and adverbs. As for nouns, we first list all nouns in the content selection database that appear with the given verb in the corpus. Then, we order these using the semantic similarity score, as demonstrated above, and select the top N nouns for use in the stories.

As a result, each plot we generate produces a set of complete stories, by enumerating all possible versions of each sentence, which can either be realised into text for presentation to the user or stored for further analysis.

4.4 Examining the Plots

We now examine the capabilities of the plot generation system by looking at some of the plots and the stories they can generate. We used the Andrew Lang fairy tale corpus for training and the same content selection, sentence planning and surface realisation modules as outlined in Chapter 3. Entity graphs were collected for the entire fairy tale corpus, discarding any that contained less than 10 nodes. There were 667 entity graphs in total (this is 43% of the entities in the corpus) with an average of 61.04 (SD 106.76) nodes. We used clustering rate (or transitivity, Newman, 2003) – the number of triangles in the graph sets of three vertices each of which is connected to each of

Entity	Number of Nodes	Number of connected graphs	Maximum chain length	% of bidirectional nodes
prince	933	3	929	0.89
princess	811	2	808	0.83
dog	219	8	202	0.71
cat	163	6	142	0.72
queen	436	3	433	0.83
bird	272	13	242	0.58
king	1076	2	1076	0.9
monster	49	11	9	0.47
daughter	350	6	340	0.68
stepmother	58	3	51	0.57

Table 4.1: Properties of selected entity graphs extracted from the fairy tale corpus. The number of nodes indicates the total number of unique nodes in the graph. Number of connected graphs indicates the number of unconnected subgraphs. Maximum chain length is the number of nodes in the largest connected subgraph. We also show the percentage of nodes that contain incoming and outgoing nodes.

the others – as a means of measuring whether nodes in the plot graphs are densely connected. This measure is inspired by social networks, in which you would expect the friend of your friend to also be your own friend. We calculate clustering rate using following formula.

$$C = \frac{3 * \text{Number of distinct triangles}}{\text{Number of distinct paths of length three}} \tag{4.3}$$

The average clustering rate is 0.027 (SD 0.076) which indicates that our graphs are substantially connected. A breakdown of the graphs for some of the most commonly occurring entities is shown in Table 4.1. As well as giving the total number of nodes for each entity we show the number of unconnected graphs, the length of the largest of these graphs and the percentage of nodes that have incoming and outgoing edges. It is encouraging to see that the majority of nodes created for each entity appear within a single connected graph and have both incoming and outgoing edges which will reduce the number of instances where the plot search reaches a dead end. We can see though that entities that appear less frequently in the corpus may have problems due to short disjoint chains, e.g., *monster* and *bird*.

Entity	Number of Nodes	Number of connected graphs	Maximum chain length	% of bidirectional nodes	Nodes gained in merging
prince ∪ princess	1826	3	1821	0.87	82
prince ∪ queen	1421	4	1415	0.87	50
king ∪ queen	1574	4	1567	0.89	60
king ∪ princess	1967	3	1962	0.88	80
prince ∪ dog	1054	7	1160	0.86	21
dog ∪ cat	386	13	349	0.71	4
queen ∪ bird	727	14	694	0.74	17
king ∪ monster	1131	7	1112	0.85	6
king ∪ daughter	1499	6	1489	0.86	73
stepmother ∪ daughter	412	7	397	0.67	4

Table 4.2: Properties of selected plot graphs extracted from the fairy tale corpus. The number of nodes indicates the total number of unique nodes in the graph. Number of connected graphs indicates the number of connected subgraphs. Maximum chain length is the number of nodes in the largest connected subgraph. We show the percentage of nodes that contain incoming and outgoing nodes. The last column shows how many new nodes were created during the merging process.

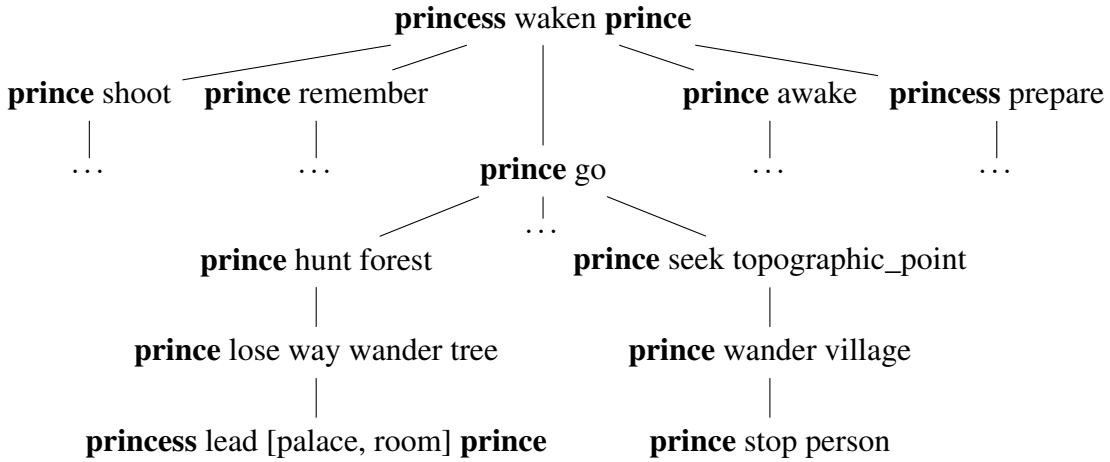


Figure 4.6: A section of the search tree generated, by the system trained on the fairy tale corpus, for the sentence *the princess wakens the prince*.

Table 4.2 illustrates the results when combining two entity graphs together to form a plot graph. Entities that appear frequently in the corpus (e.g., *prince* and *princess*) lead to a large number (in this case 82) of new nodes that are created from mergers in the entity graphs. However, some entity pairs are difficult to combine. For example, although the entity graph for *king* contains a large number of nodes, only 6 new nodes are created in the merger with *monster*.

We next inspect some of the plots our system produces for the entities *prince* and *princess*. Specifically, we supplied the system with the sentence *the princess wakens the prince* and asked it to produce stories of length 5. Figure 4.6 shows a sample of these plots. For this example we kept the branching factor to 5, so only 5 children could be considered from each node (as each sentence after the input sentence can be comprised of up to 2 nodes, this places an upper limit of 390,625 plots). In total there were 112 plots generated for this particular input sentence.

We must next examine the stories these plots are capable of producing. The number of possible choices for each unsatisfied lexical node was set to 5. In total, the 112 plots generated 10,624 stories. Taking one of the plots in the above example we will show how the story generator satisfies lexical choice decisions. We start with a list of plot nodes.

princess waken **prince**, **prince** go, **prince** hunt forest, **prince** lose way wander tree, **princess** lead [palace, room] **prince**. (4.4)

Some nodes may at first seem to make little sense, such as, ***prince*** *lose way wander tree*. However, this is because we have not yet added structural information to the sentence or resolved lexical variables. We next proceed to do this; for the entity *way* we select the words *way*, *strength*, *shape* and *ring*. These are the the most similar to *way's* WordNet sense[4] and co-occur with *lose*.

> **princess** waken **prince**, **prince** go, **prince** hunt [forest], **prince** lose (4.5)
> [way, strength, shape, ring] wander [tree, giant, friend, wizard, fellow],
> **princess** lead [room, castle, palace, house, cottage] **prince**.

This represents many different stories depending on the permutations of lexical choices we make. An example of one of the stories we could realise from this example is the following, depicting a scene in which the prince once being saved by the princess then becomes lost.

> The princess wakens the prince. The prince goes. The prince hunts in the forest. The prince loses the way wandering among the trees. The princess leads the castle in place of the prince.

From this example there are several points for consideration. First we can see that the 5 sentence story has been constructed from 6 plot nodes. The nodes *(**prince** lose way, **prince** wander tree)* have been aggregated into a single sentence by structuring the second node as a subclause of the first. It is clear that this results in more natural stories than simply enumerating each graph node as a single sentence. Although not shown in our example above, we also encountered plots in which the nodes *(**prince** go, **prince** hunt forest)* were aggregated to create the sentence *the prince goes to hunt in the forest*. There are also interesting implications rising from the selection of entities. In confining word forms to similar senses we are trying to narrow what may often be a large selection of words in the content selection database. The node ***prince*** *hunt forest* only considers the word form *forest* as an object. However, the content selection database actually contains 46 different entities for this argument. What it comes down to is capturing the intention of the action. The node in our example is describing to us an event where the prince is hunting in a certain place. The verb hunt, however, could also have been used to describe the prince hunting for a particular entity, either simply to find them or for sport. Other entities in the content selection database would

[4]Sense 1: manner, mode, style, way, fashion – (how something is done or how it happens; "her dignified manner"; "his rapid manner of talking"; "their nomadic mode of existence"; "in the characteristic New York style"; "a lonely way of life"; "in an abrasive fashion").

The princess wakens the prince. The prince awakes. The prince tells an aunt about asking for the woman. The prince picks up a pebble. The prince takes the property in the ring.
The knight fights the dragon. The knight rides. The knight is to do for men out of the plain. The knight meets with the road. The poor knight shakes the head.
The fairy saves the princess. The princess goes home. The princess tells of the king. The princess marries a woman. The princess picks the garden out of the pebble.
The witch captures the girl. The witch shouts. The witch becomes the body. The ugly witch takes a piece. The ugly witch holds a canopy.

Table 4.3: Example of stories created by the plot based system for the input sentences; *the princess wakens the prince*, *the knight fights the dragon*, *the fairy saves the princess* and *the witch captures the girl*.

have been more suited to these alternative meanings, e.g., *tiger*, *stranger*, *treasure* and *monster*. In this example, the sense information ensures that the chosen lexicalisations conform to the original meaning of the node's action.

In Chapter 3 we described how a lexicalised model of interest and an entity grid model of local coherence could be used to discern the single best story created by the generate-and-rank system. We applied the same approach to the stories resulting from our plots, with examples shown in Table 4.3 for a selection of input sentences. The story generated for *the princess wakens the prince* is the first in the table. One of the most peculiar parts of this story is the sentence *the prince picks up a pebble.* It simply comes out of nowhere and does not make much sense. This sentence is a result of the graph node ***prince*** *pick object*; the system simply does not have enough information as to what it is that the prince is supposed to be picking. The objects attested with *pick* include *diamond*, *girl* and *treasure*, which make more sense in the context of the story, but the system can only select from those with the most similar sense to that of *object*. The words that were selected when generating stories for this plot node are *ground*, *goods*, *stone*, *piece* and *pebble*. We described earlier how we use Wu and Palmer (1994) to score the similarity of two senses. This measure is based on the path length between the sense nodes in WordNet, counting nodes closer together as more semantically similar. This means that when comparing words from the content selection database, to assume a lexical argument, the system will select those closer to *object* and *person*. These senses are, however, rather abstract, indicating a category of

senses rather than a particular entity. The word *pebble* is selected by the system rather than *diamond* as the former has a shorter path to the sense *object*.

pebble
=>rock, stone
=>natural object
=>**object**, physical object (4.6)

diamond
=>jewel, gem, precious stone
=>jewelry, jewellery
=>adornment
=>decoration, ornament, ornamentation
=>artifact, artefact
=>**object**, physical object (4.7)

Our similarity measure will select those senses closest to the target sense, as those are the ones it considers most similar. There is obviously a trade-off between using an abstract sense place holder, such as *object* or *person*. However, unless we want to discount these actions completely from the plot then there is no other alternative for actions in which not all of the entities mentioned have been resolved. We will discuss in the next chapter how the problem can be alleviated by removing the limits on the number of entities to be considered in an action. This problem is common, resurfacing in other generated stories, e.g., ***princess*** *marry person* becomes *the princess marries a woman* and ***witch*** *become object* becomes *the witch become a body*.

Despite these problems in lexical choice, the generated stories are promising. We can see logical action sequences appearing, such as *the princess wakens the prince, the prince awakes*. We must also consider the fact that the number of complete stories considered by the generator was on average around 10,000. Each sentence was generated once and reused if the node appeared in more than one plot, thus reducing computation time. This is a stark contrast to the generate-and-rank system where each generated sentence was unique to each story under construction. On average the generate-and-rank system created around 500,000 sentences, many of which ended up being identical.[5]

[5]Unlike the plot-based system, the generate-and-rank system employs a discourse history when generating sentences for a particular story. This means the sentences created at a particular depth of the search tree cannot be reused as the discourse history for that node is unique.

However, there are still some caveats with our approach. Although the search is much more directed than it was in the generate-and-rank approach, with plots outlining the expected actions, the system still lacks document level control over how lexical choice decisions are made. Each action represented in a plot can contain a very large number of representations, outlined in the grammar rules of the sentence planner. Although the use of sense information has improved our ability to order the candidates for lexical choice, we find that still only a small proportion can be searched, which may not necessarily lead to the best story. These lexical choice decisions are made in isolation with no document level control. A coherent text is more likely to have recurring entities, but the plot based system has no mechanism to ensure any entity other than the topic entities reappear.

It is also clear that taking the default sense for each word form is not optimal. One example is the word *queen*. In the fairy tale domain, the word *queen* refers to a female monarch. Unfortunately, this is not the default sense in the current version of WordNet[6] and as a result our system interprets *queen*[7] to be the single egg laying female of an insect colony. Surprisingly, this has not caused too many problems, as the sense for *insect* and *person* are closely linked through the sense *organism*. Other notable erroneous sense assignments were for the words *scorpion*[8] and *tiger*[9] which were classified as descendants of the sense *person*. This means that when the system is looking for a *person* to take the object role in a given action, it is more likely to select *scorpion* or *tiger* than *queen*. It is clear that the system requires the ability to perform word sense disambiguation on the corpus whilst extracting the entity graphs.

The use of the senses *object* and *person* allowed us to keep actions that were encountered in the corpus with pronouns as arguments. These senses however tell us little about the type of entity we should be using in these roles when it comes to turn the plot into a story. An example is the action ***princess*** *put object* where candidates for the sense *object* are the word forms *ground*, *piece*, *body*, *spit* or *nest*. One option would be to maintain the senses *object* and *person* only until the node is merged and a different sense for this slot is found. For example, it may be better that the action ***princess*** *recognise [object, person, husband]* be recorded as ***princess*** *recognise husband* which narrows the range of entities that could assume the object role of the action. We have to

[6] At the time of writing 3.0.2.

[7] queen – (the only fertile female in a colony of social insects such as bees and ants and termites; its function is to lay eggs)

[8] Scorpio, Scorpion – ((astrology) a person who is born while the sun is in Scorpio)

[9] tiger – (a fierce or audacious person; “he’s a tiger on the tennis court”; “it aroused the tiger in me”)

bear in mind though that *husband* might not make sense in context for every *recognise* action the *princess* performs.

Problems also occur due to the lack of ordering when constructing entity graphs. Counts are used to calculate Mutual Information scores but these may not give the best picture overall of how entities behave. A better approach may be to employ the clustering method used in Chambers and Jurafsky (2009) where the addition of nodes within a narrative chain depends on how well it fits with the overall chain. Our approach records all events associated with an entity in the entity graph. Some of these events could be omitted as they add little information to the plot. Events such as *getting*, *saying* and *asking* appear frequently in the plots but rarely produce sentences that contribute to an overall narrative. This is because these actions are often found in dialogue, which our system cannot represent. The ordering of actions are also likely to be incorrect in some cases. Strategies to correctly determine the temporal ordering of actions have previously been described in Bramsen et al. (2006) and Chambers and Jurafsky (2008).

We gave an example of how the aggregation of graph nodes can create more interesting and natural sentence structures. However, we have only outlined a very simplistic method of aggregation which will not always work in practice. There are several considerations to be made when performing aggregation. The first is how do the two clauses relate to one another? Our system is agnostic with regards to this information.

4.5 Summary of Chapter

In this chapter, we explained how story plots can be automatically extracted from corpora and used to generate stories. These plots represent story plans, something the generate-and-rank generation approach did not have access to. Graphs are created for each entity in the corpus indicating the progressions of actions it participated in, the order these actions occur and which other types of entity it is seen to interact with. These graphs are subsequently combined to create plots representing stories based on a user supplied input sentence. From these plots, the story generation system can make lexical choice decisions based on the types of entity attested within the plot actions in the original text. By searching through these possible stories the system can then select the one best story to present as an output.

As we mentioned above, one of the biggest problems with plot-based generation is that it cannot make lexical choice decisions whilst considering their impact on the

document as a whole. We shall see in the next chapter how the stories generated by the plot based system can be viewed not as a final product of a story generation system but as a starting point of a more elaborate search procedure. We will show that using evolutionary search techniques, stories can be optimised to overcome problems with lexical node satisfactions. We will also show that evolutionary search techniques provide a strong platform for exploring the story space, removing the limitations imposed by the generate-and-rank approach proposed in the previous chapter. In addition, we can remove the need for an explicit model of interest, by allowing interesting stories to develop through the optimisation of established story plots.

Chapter 5

Evolutionary Search for Story Generation

The generate-and-rank system presented in Chapter 3 can produce a large number of potential stories. However, in order to be computationally feasible it can only explore a small portion of the search space. Limitations are set on the number of sentences the system can produce at each depth of the search. This means restricting the number of actions and entities that appear in the final story. By limiting the search in this manner, the system may never encounter the sentences that would result in the best story overall. In the previous chapter we discussed how this problem may be alleviated by introducing document level controls over the generation process, which we operationalised using plots. The later allow us to create an outline for a story which we no longer need to build sentence by sentence. Each plot however, still potentially represents a large number of stories that we need to search in order to find the best one. Unfortunately, an exhaustive search is also not computationally feasible. In this chapter, we propose to use story plots as a starting point from which to explore the search space advantageously so as to arrive at one 'good enough' story.

The search task can be viewed as an optimisation problem. We already have the ability to produce stories which we now must improve upon along a set of criteria. We propose that evolutionary search techniques are well suited for this type of search problem as they allow the search space to be explored stochastically. In particular, we will use genetic algorithms (GAs) a well-known technique for finding approximate (or exact) solutions to optimisation problems. The basic idea behind GAs is based on "natural selection" and the Darwinian principle of the survival of the fittest (Mitchell, 1998). In this chapter we present a story generation system that utilises GAs to evolve

an optimal story. Below, we introduce the basic GA procedure and show how it can be adapted to our task. We start by reviewing related work that has used evolutionary search techniques for natural language generation.

5.1 Related Work

Mellish et al. (1998) pioneered the use of GAs in natural language generation and subsequent research has built upon their work, especially in the field of document structuring. They propose a system for generating textual descriptions of museum artefacts from a list of facts and relationships between them. The task is to find the best document structure that includes all the facts about an artefact whilst also being coherent. They use Rhetorical Structure Theory (RST, Mann and Thompson, 1987) to define document structure, with leaf nodes representing the facts and higher levels denoting how they relate (e.g., that one fact can contrast or elaborate another). To find the optimal RS tree they advocate genetic algorithms as an alternative to exhaustively searching all possible descriptions. They outline a measure of document coherence based on the structure of the RS tree and the ordering of entities and facts. For each artefact, there are a set of the facts relating to it represented as snippets of English text, and the known rhetorical relationships between those facts. The GA search starts with a population of random document trees, these are created by loosely following sequences of facts where consecutive facts mention the same entity. This population then undergoes mutation and crossover. Mutation occurs by randomly assigning a fact a new position among the tree's leaves. Crossover takes two parent trees and produces a single child by taking a subset of facts from the first parent and inserting it into the second parent, removing duplicates as each fact must appear at most once in the final document structure. The result of this search is a document structure for a particular artefact, however, this system was not integrated with a surface realiser, so the final document contains only the 'canned text' of each of the input facts.

Cheng and Mellish (2000) extend this work by focussing on the interaction of aggregation and text planning, using a GA to search for the best aggregated document that satisfies coherence constraints. They extend the fitness function of Mellish et al. (1998) to include features for local coherence, embedding and semantic parataxis[1]. To

[1]Two facts are said to have a semantic parataxic relationship if they are both children of a shared multi-nuclear semantic relationship (e.g., *sequence* and *contrast*). These facts are related but are considered of equal importance in the sentence. In contrast, a semantic hypotaxic relationship is indicated by nucleus-satellite relationship in the RS-tree (e.g., *clause*) making one fact dependant on the other.

evaluate they scored the orderings of the human authored descriptions of the artefacts and compared them against the scores of their system generated orderings. They found that the scores for the machine authored texts often came very close to that of the human authored ones, although no formal evaluation was carried out to see if human evaluators preferred one over the other.

Karamanis and Manurung (2002) also utilise GAs to search for the best ordering of facts about museum artefacts. However, rather then using RS trees to evaluate document structures as in Mellish et al. (1998), they simply use the ordering of the facts themselves. They argue that for a produced text to be coherent, it must retain *continuity*, i.e., each utterance should refer to at least one entity in the utterance that preceded it. To evaluate each candidate solution, they use a fitness function that counts the number of continuity preservations. They mutate members of the population by either randomly permuting the ordering, swapping two facts or repositioning a single fact. Their crossover procedure is the same as in Mellish et al. (1998). Once the search finishes, the ordering of the facts represents a description which is realised using 'canned text' associated with each fact. Their results show that it is possible for GAs to find a global optimum solution using only the principle of continuity in order to create a coherent ordering of facts, at least in the museum artefact domain.

Unlike Mellish et al. (1998) who use GAs as a means for optimising the implementation of a content planner, Duboue and McKeown (2002) employ evolutionary algorithms in order to *learn* a content planner itself. In particular, they wanted to use search to produce a planner similar to that of MAGIC (McKeown et al., 1997). The MAGIC system utilises a manually developed planner to generate post cardiac-surgery medical reports or briefings from raw data collected in the operating room. By training a content planner using GAs it was hoped to overcome the inflexible, consistent nature, of the generated texts resulting from the manually developed planner. They operationalise the content planner as a tree consisting of operators which identify what information from the semantic input to use and how it should be structured. These operators are either structure-defining (discourse or topic level) or data-defining (atomic level). An example of an atomic operator is the age of the patient, whereas the structure-defining operators form a discourse plan of how the atomic operators should be placed in the document. Their task is to optimise the tree-like structures in order to find a planner that results in documents similar to human authored reports for the same surgical operations. Starting with an initial population of random planner trees, they evolve this population using two corpus-based fitness functions. The first fitness

function assess to what extent the placement of atomic operators violates ordering constraints learned for the domain. The second fitness function then evaluates the plans by comparing the average similarity to the reports produced automatically against those authored by the physicians for the same patient. The process of generating a new population of planners involved three mutation operators and a single crossover operation. Mutations occur by inserting, deleting and combining nodes. The crossover operation creates new planner trees by combining the structures of two plans selected from the population. Duboue and McKeown (2002) find that the best plan returned by the GA is more similar in structure to manually constructed MAGIC planner than a randomly created planner. This result shows that GAs have the potential to train Natural Language Generation components that are comparable to those constructed manually by humans.

In a related task, Manurung (2003) explore the use of GAs in order to generate poetry. They developed a NLG component, MCGONAGALL, that takes as its input a non-hierarchical semantic representation (e.g., *john(j), mary(m), loves(l, j, m)*) that is to be expressed poetically. The system encodes each possible representation of the semantic input as a lexicalised tree-adjoining grammar. A GA search is then employed to evolve these grammar structures, optimising in terms of *meaningfulness* (semantic faithfulness), *grammaticality*, and *poeticness* (metre conformance). They were able to show through empirical studies that as a proof of concept MCGONAGALL was capable of optimising text to match designated meters, such as haiku and limerick, as well as correctly representing the underlying semantic meaning. However, they found that the results suffered when optimizing on more than one evaluation metric at a time.

5.2 Genetic Algorithms

We have taken the simple canonical GA outlined in Mitchell (1998) as the basis for the GA search we describe below. The GA search procedure centres around a population which contains a number of individuals (or solutions). These individuals are each represented by a genetic string (e.g., a population of chromosomes). The search begins with an initial population that is randomly created and contains a predefined number of individuals. The job of the GA is to evolve from this population a new and better population by breeding the individuals based on their fitness. An individual's fitness is evaluated according to an objective function (also called a fitness function). Individuals that are selected to be parents undergo *crossover* (also called recombination)

and *mutation* in order to develop the new population. The motivation here is that fit parents will produce superior offspring improving the overall fitness of the population. The algorithm thus identifies the individuals with the optimising 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 with natural adaptation.

One of the key strengths of GAs as a search technique is that they contain an element of randomness. If we were to always select the fittest individuals to breed, then we would quickly reach a point where our population converges at a local optimum. However by introducing a random element in the application of the selection, crossover and mutation operators we can force the genetic algorithm to keep exploring the search space and stop it from becoming trapped. We also argue that by breaking from a strictly deterministic method of story generation, we can add more creativity to our stories as we entertain story candidates that systems following a more traditional architecture could not have considered.

The input to our GA-based story generator is similar to the generate-and-rank based system presented in Chapter 3. The user provides a sentence (e.g., *the princess loves the prince*), which contains two story protagonists (the main characters the story will focus on), and the desired story length. Unlike the generate-and-rank approach this generator relies on a story planner for creating meaningful stories. The GA-based system differs from the generate-and-rank system in two important ways. Firstly, it 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 (see Chapter 4). Secondly, our search procedure is simpler and more global; the generate-and-rank system searches for the best story twice (it first find the N-best stories and then subsequently re-ranks them based on coherence and interest), whereas our genetic algorithm explores the space of possible stories once.

5.2.1 Initial Population

Genetic algorithms traditionally start with a randomised population of individuals. However, rather than initialising the GA with random stories, we start from possi-

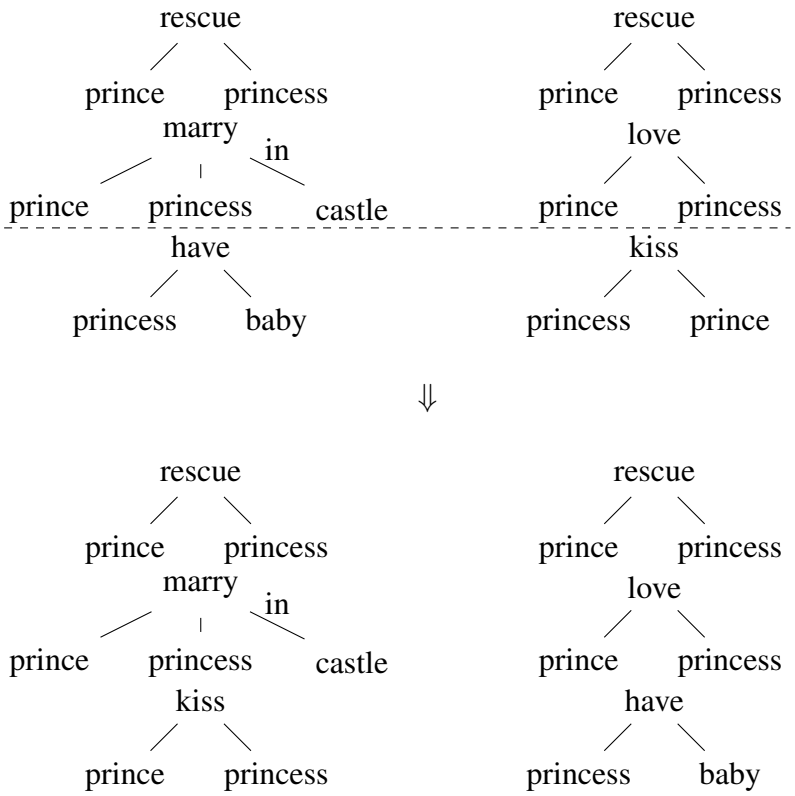


Figure 5.1: Example of the genetic algorithm crossover operator as applied to story structures. Here two stories undergo crossover at a single point, indicated by the dashed line, resulting in two children which are a recombination of the parents.

ble story plots instantiated by the input sentence. These plots in turn give rise to many different stories. By randomly sampling this pool of stories we can create an initial population of a desired size that is random in nature but contains stories with a narrative. Each individual within our population is a story, represented as a list of dependency trees each of which corresponds to a sentence. Each of these sentence trees consists of nodes representing lexical items and edges representing dependencies.

5.2.2 Crossover

In order to produce the next generation of the population, a selection process is used that will pick two members of the current population based on their fitness. There is then a small probability (known as the crossover rate) that these two individuals will undergo mating and produce offspring. If they do not undergo mating then they are simply copied straight into the new population (we cannot expect each population to

be completely different from the last). However, if they do produce children, then these children will be the result of a recombination of the parents. Before the offspring can be created, one index in the sentence ordering (although potentially more) is assigned along the length of the parent chromosomes. To create a child, we copy the chromosome up to this point in the first parent and then after this point in the second (and visa-versa). Each child now contains some genetic material from each parent. As our solutions are represented by an ordered graph of dependency trees, our crossover point will represent depth in the ordered list. The crossover point must be the same depth for both parents to ensure that stories remain at the desired length. We assume that the crossover point occurs after the input sentence but we use a uniform distribution to decide where to place it.

Figure 5.1 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 stories are created by swapping their last sentences.

5.2.3 Mutation

In the event that two selected candidate solutions produce children, these children may then undergo mutation before entering the next generation. Mutation allows for adaptation and search within the GA; without this operation we would only produce recombinations of stories already present in the population. To explore a larger space of possible stories, we must be able to mutate genetic material in the population into new solutions. Mutation, however, has a very disruptive effect on the population, as it has the ability to completely alter a chromosome, so it must be used sparingly. Before entering the population, each child has a small probability of undergoing one or more mutations. Mutation can occur on any verb, noun, adverb, or adjective in the story. 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 (similar lexical items co-occur with the parent node in the training corpus, see Figure 5.2a). 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 5.2c). This new sentence must have one of the story protagonists as its subject to retain coherence. If it is a subclause, then it is replaced with a randomly generated clause, headed by a verb in the training corpus that co-occurred with the matrix verb (Figure 5.2d). The sen-

tence planner selects a template tree and fills it using the content selection database to generate random clauses. Mutation may also change the order of any two sentences in the list in the hope that this will increase the story's coherence or create some element of surprise (see Figure 5.2b).

5.2.4 Selection

The process by which we select individuals for possible mating is of primary importance in our story generation setting. Obviously the emphasis is on the 'survival of the fittest', yet as we have already explained we do not wish to concentrate solely on the fittest individuals as they may trap the search within a local maximum. Potentially advantageous genetic material may be contained within the chromosome of a less fit individual, yet through recombination this material may eventually lead to a superior child. Selection methods must therefore tend towards choosing fitter individuals but not totally discount less fit ones. We can do this by using 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. To ensure that through this somewhat random process of selection we do not loose the fittest members of the current generation we employ some *elitism* by allowing the top 1% of solutions to be copied straight from one generation to the next.

One problem faced by the GA is that not all solutions will be valid. Specifically, we consider invalid stories to contain repetitive actions. This was not a problem for the generate-and-rank system as it built up each sentence in a very directed manner, whereas, the GA has the ability to make alterations at any point in a story through crossover and mutation. We enforce validity by ensuring that a *subj-verb* pair only appears once in the story.[2] In the cases where repetition occurs, the story is assigned a low fitness, without however being discarded, as it may still contain some useful material. For example when we perform crossover on the stories *(princess love prince, princess see castle, prince ride horse, prince save kingdom)* and *(princess love prince, princess cry, princess see castle, princess go on quest)*, it could produce the story *(princess love prince, princess see castle, princess see castle, princess go on quest)*. This new story will be given a low score as there is a sentence repetition, however, the some of the sentences could be useful in future crossover operations, (e.g., *princess*

[2] As each of the actions in the story has one of the story protagonist as its subject, we can assume sentences will be distinct enough that we do not need to check *verb-object* pairs also.

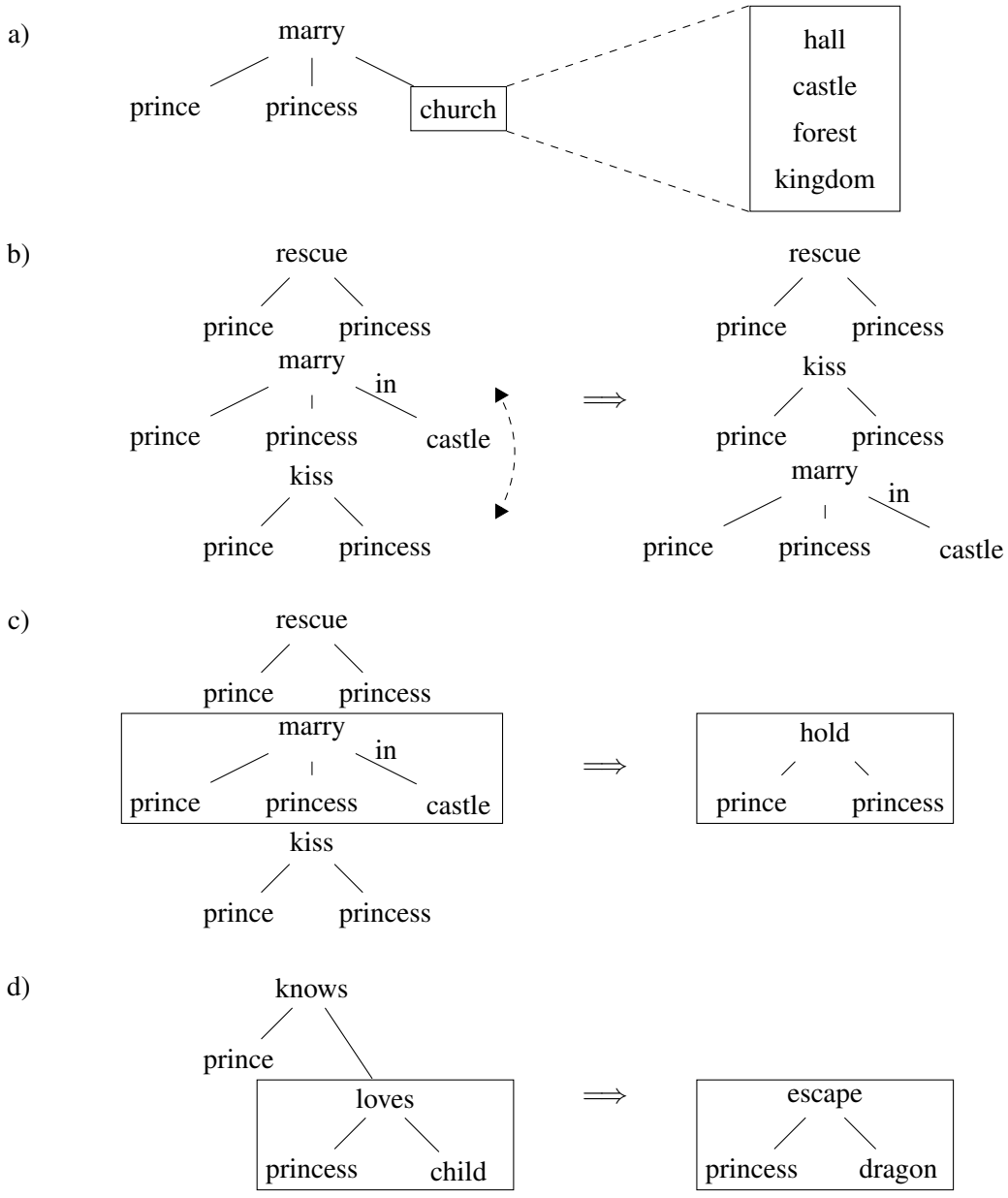


Figure 5.2: Example of genetic algorithm mutation operators as they are applied to story structures: a) mutation of a lexical node, *church* can be replaced from a list of semantically related candidates; b) sentences can be switched under mutation to create a potentially more coherent structure; c) if the matrix verb undergoes mutation then, a random sentence is generated to replace it; d) if the verb chosen for mutation is the head of a subclause, then a random subclause replaces it.

see castle, princess go on quest).

As a GA search progresses, the number of individuals in the population remains constant but the number of unique solutions they represent will decrease as the population converges over a suitable solution. This results from the selection procedure selecting the same individual within a single generation more than once and placing it directly into the next generation, producing clones. The presence of identical solutions in the population will increase the selection pressure in favour of the fittest solution, which exerts itself over the population by replicating itself in this manner. This selection pressure aids the population in reaching convergence, however, there is a trade-off between the speed at which a convergence is sought and allowing the GA to explore a wider pool of solutions. Our system treats each of the solutions in the population as an individual, even if they are clones of other solutions.

For many search problems the GA is particularly advantageous as the landscape defined by the fitness function contains an optimum solution to be reached. We share the view of Karamanis and Manurung (2002) that in natural language generation tasks there does not need to be a global optimum solution, there simply needs to be a solution that is coherent enough to be understood.

In the next section we will discuss some of the fitness functions that can be used by the selection procedure to evaluate our story population.

5.2.5 Fitness Functions

The fitness function evaluates and scores each individual in the population. In a traditional GA, the fitness function deals with one optimisation objective. It is possible to optimise several objectives either using a voting model or more sophisticated methods such as Pareto ranking (Goldberg, 1989). Any measure of story quality used as part of a fitness function must be automatic and efficient, as the GA may contain a large number of candidate stories that require scoring. We follow previous work (Mellish et al., 1998; Karamanis and Manurung, 2002) which focuses on fitness function that scored candidates based on their coherence. In Chapter 3 we motivated a scoring function based on interest and approximated by 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. However, although an interest scoring function made sense for the generate-and-rank system as a means of selecting unusual stories, in the GA-based system it seems redundant. Interesting stories emerge natu-

rally through the operations of crossover and mutation, and we can assume that story plots will already be interesting as they contain sequences of actions consistent with the training corpus. In essence, we are exploiting the nature of the GA search to develop the interest of our stories at the document level rather than at the lexical level.

In contrast to previous work (Mellish et al., 1998; Cheng and Mellish, 2000) our fitness functions are not based on the well-formedness of RST trees. Karamanis and Manurung (2002) show that coherent texts can also be produced using a similar and potentially more robust fitness function based the number of *continuity* violations within a text. Continuity violations can be easily calculated by the entities present in each utterance with those of the previous utterance. We will now outline several measures for evaluating text coherence that are easily calculated for our stories. We have previously discussed a measure of local coherence based on the entity grid representation (Section 3.3.3). Also, Clarke and Lapata (2007) in their work on sentence compression, use measures derived from Centering Theory (Grosz et al., 1995) and Lexical Chains (Morris and Hirst, 1991) to evaluate the coherence of their compressed documents.

Entity Grid The local coherence of a text can be evaluated using a representation based on entity grids (Barzilay and Lapata, 2008). This approach represents each document as a 2-dimensional array in which the columns corresponds 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. Central to their approach is the idea that the distribution of entities in a locally coherent text will exhibit certain regularities. From a corpus we can learn the probabilities of such transition sequences and then use them to calculate the probability of the sentence transitions in a given text. This approach was successfully used in Barzilay and Lapata (2008) to learn a ranking function for sentence ordering. We describe, in detail, in Section 3.3.3 how the entity grid model was trained. An example entity grid is shown in Figure 3.7. In each sentence we record if an entity is a subject (S), direct object (O), indirect object (X) or not present (–). Although Barzilay and Lapata (2008) used the symbol X for an entity present in the sentence that was neither a subject or an object, we will use it to represent an indirect object.

Centering Theory A central assumption in Centering Theory (CT, Grosz et al., 1995) is that the focus of a discourse revolves around a centred entity. It is also assumed that

The $\boxed{\text{prince}_1}$ loves the princess$_1$. The $\boxed{\text{princess}_1}$ sees a dragon$_2$ near the city. The $\boxed{\text{prince}_1}$ draws his sword$_3$ and goes to meet the dragon$_2$. The $\boxed{\text{prince}_1}$ slays the dragon$_2$ with the weapon$_3$. The $\boxed{\text{princess}_1}$ kisses the prince$_1$.

Figure 5.3: An example story with centers highlighted in boxes and entities belonging to lexical chains shown with subscripts; entities with the same number are in the same chain (e.g., *prince*, *princess*).

texts whose utterances share a common center are more coherent than texts where the center shifts from one utterance to the next and that salient entities are more likely to be centers. Salience is determined by different factors such as the entity's syntactic role, whether it is a subject, and whether it is prenominalised or not. Assuming that utterances correspond to sentences (Miltsakaki and Kukich, 2000), we can calculate for each sentence U_n:

- **Preferred center**, CpU_n, the subject of the matrix verb in sentence U_n.
- **Backward-looking center**, CbU_n, the most highly ranked entity in sentence U_{n-1} which is also realised in sentence U_n.
- **Forward-looking center**, CfU_n, the set of entities that appear in sentence U_n ranked by their salience. The ranking of Cf will allow the Cp of this sentence and the Cb of the following sentence to be identified.

The salience of each entity in the set of forward looking centers is calculated by assuming that subjects are ranked higher than objects, which in turn are ranked higher than all other entities.

There are currently several different coherence metrics that use CT as their basis. Karamanis et al. (2009) discuss and evaluate a selection of centering based measures for information ordering in NLG (including the entity grid shown above). They found that ensuring each utterance had a CbU_n provided a baseline that was difficult to outperform when incorporating additional centering features. However, they also explain that its performance is not suitable for it to be used as a measure for sentence ordering on its own. As we are focussing on the task of content selection in conjunction

with sentence ordering, we decided to focus on the centering based measure described in Kibble and Power (2004) in order to provide additional granularity to the level of coherence each text could be assigned. Kibble and Power (2004) describe a measure that scores the transitions from one utterance to the next in a text, and show that it can be used to select an optimal realisation from a selection of different text plans, for the same document. From the sentence trees that form our stories we know exactly which entities appear in each sentence and the role they play in each action. We take each sentence, U_n, in a story and calculate the preferred center (CpU_n), the backward-looking center (CbU_n) and the forward looking center (CfU_n). To score a story we look at violations of salience, cohesion, cheapness and continuity which are defined in Kibble and Power (2004) as follows;

- **Salience violation**: An utterance U_n violates salience if $CbU_n \neq CpU_n$. This defect is assessed only on utterances that have a backward-looking center.
- **Cohesion violation**: A transition $\langle U_{n-1},\ U_n \rangle$ violates cohesion if $Cb(U_n) \neq Cb(U_{n-1})$. This defect is not recorded when either U_n or U_{n-1} has no Cb.
- **Cheapness violation**: A transition $\langle U_{n-1},\ U_n \rangle$ violates cheapness if $CbU_n \neq CpU_{n-1}$. This defect is assessed only on utterances that have a backwards-looking center.
- **Continuity violation**: This defect is recorded for any utterance with no Cb except the first proposition in the sequence (which by definition cannot have a Cb).

We also follow their work by assigning each salience, cohesion and cheapness violation a score of 1 and each continuity violation a score of 3. Continuity violations are seen as more sever as they represent an expensive *rough shift*, e.g., the entities in U_n are all different from those in U_{n-1}. The sum of all violations gives an idea of how well a story adheres to the principles of CT. The stories with the lowest scores are expected to have the greatest local coherence. Kibble and Power (2004) note that the weight given to the violations may differ depending on the type of text being evaluated (i.e., different genres, spoken vs. written), however we will use their weights as they still provide an insight into how well a document adheres to the principles of CT. We will now show how the story in Figure 5.3 is evaluated by the Kibble and Power (2004) centering measure. The first step is to calculate the preferred center (CpU_n),

1 The prince loves the princess.
`Cf(prince, princess)`
`Cp=prince`
2 The princess sees a dragon near the city.
`Cf(princess, dragon, city)`
`Cb=princess Cp=princess`
3 The prince draws his sword and goes to meet the dragon.
`Cf(prince, sword, dragon)`
`Cb=dragon Cp=prince`
4 The prince slays the dragon with the weapon.
`Cf(prince, dragon, weapon)`
`Cb=prince Cp=prince`
5 The princess kisses the prince.
`Cf(princess, prince)`
`Cb=prince Cp=princess`

Figure 5.4: The preferred center (Cp), the backward-looking center (Cb) and the forward looking centers (Cf) for each sentence (U_n) of the story in Figure 5.3.

the backward-looking center (CbU_n) and the forward looking centers (CfU_n) for each sentence. These are shown in Figure 5.4.

We then record any violations that occur between entity transitions which in our example are: salience in sentences 3 and 5 (as $Cb \neq Cp$), cohesion in transitions 2-3 and 3-4 (as $CbU_n \neq CbU_{n-1}$) and cheapness in transition 1-2 and 2-3 (as $CbU_n \neq CpU_{n-1}$). This would give the above story a violation score of 6. We normalise the scores by story length by bearing in mind that the maximum number of violations for each utterance after the initial utterance is 3, either salience+cohesion+cheapness violations or a continuity violation.

Lexical Chains As we discuss above, Centering Theory describes how a locally coherent text will maintain the discourse focus on a centered discourse entity. However, the *cohesion* of the utterances may rely on more than just the centered entity. For example, the discourse, *'the prince has a football, the prince loves chicken, the prince visits France'*, does not violate any centering constraints, as the *prince* is the focus of each sentence, but this text is not very coherent. *Lexical cohesion* takes into account

the relatedness of lexical items in the story and their distribution. Halliday and Hasan (1976) coined the term to explain the fact that coherent texts tend to contain more related terms and phrases than incoherent ones. As we saw in the text above, the lexical items *football*, *chicken* and *France* are unrelated and the story does not make much sense. However, consider the story, *'the prince has a puppy, the prince likes dogs, the prince does not like cats'*, which appears much more coherent as the lexical items, *puppy*, *dog* and *cat* are semantically related.

We can view coherent documents as those containing more related terms. In particular, we will use *lexical chains* (Morris and Hirst, 1991) which represent lexical cohesion through semantically related words. Using WordNet (Fellbaum, 1998), relationships such as synonymy, hyponomy and mernomyny, can be identified for noun entities in the text. These relationships along with entity repetition then allow chains of nouns to be built throughout the text. Lexical chains allow us to quickly view the coherence of a text by viewing their distribution. Texts where the chains form dense clusters will have a higher degree of coherence as the words are lexically related whereas chains with sparse clusters will indicate a lack of coherence. Lexical chains have been successfully used in the areas of sentence compression (Clarke and Lapata, 2007) and document summarisation (Barzilay and Elhadad, 1997), as they can identify topic boundaries: entities within a topic will relate to one another stronger than they will with entities of another topic. Documents will often contain multiple topics and producing a document summary requires deciding which of these to include whilst retaining the most important information from the document. Of the topics in a document, those represented by the largest lexical chains are likely to reveal what a topic is about and so will play an important role in any summary produced. Whereas, shorter chains are more likely to represent topics that have been included to provide supplementary information, so their omission from a summary is unlikely to detract from the message the document is intended to convey.

To build lexical chains, we follow the procedure outlined in Barzilay and Elhadad (1997) that is inspired by the work of Hirst and St-Onge (1998).

1. Select a set of candidate words (typically all words in the document that appear in WordNet).

2. For each candidate word, find the appropriate chain relying on a relatedness criterion among members of the chains.

3. If a chain is found, insert the word into the chain.

Candidate words are those nouns found in WordNet. There are three types of relationship used in determining relatedness between members of a chain, which also depend on the typographical distance between the nouns in the document. These are, *extra-strong* (between repetitions of a word, no limit on distance between occurrences), *strong* (between words connected by a relationship in WordNet, within a window of seven sentences) and *medium-strong* (where the path length between the synsets of the words is greater than 1, within a window of three sentences).

Hirst and St-Onge (1998) utilise a greedy strategy when building chains, using an iterative process assigning each word to a chain in which the strongest of the above relationships can be applied, or adding it to a new chain. This greedy strategy leads to inconsistent sense allocations as the sense selected in order for the word to fit in with the current chains may not be the correct sense if when considering all the words in the document. Barzilay and Elhadad (1997) improve upon their implementation by introducing word sense disambiguation. Rather than applying and updating the sense of each word based on the current set of chains, they build the set of chains resulting from each sense of each word. By recording all possible chains they can select the set of chains that result in the strongest relationships based on the words from the whole document.

To give a numerical value indicating the strength of a lexical chain in a story, we use the scoring function proposed in Barzilay and Elhadad (1997). From the set of chains created for a document, they select the strongest of these chains as indicators of the key topics for use in a documents summary. To select the strongest chains they show that good indicators of a chain's strength are:

Length: The number of occurrences of members of the chain.

Homogeneity index: 1 - the number of distinct occurrences divided by chain length.

These are combined to give the following measure of chain strength:

$$Score(Chain) = Length * Homogeneity \tag{5.1}$$

For each story we calculate the average lexical chain strength from each of the identified chains. For the example shown in Figure 5.3 there are three chains; *{prince, princess}*, *{dragon}* and *{sword, weapon}* (*castle* is not semantically related to any other word in the story). We start by scoring each of the chains using equation 5.1. The chain *{prince, princess}* has a length of 7 and contains 2 distinct occurrences. Its

score is therefore, $7 * (1 - (2/7)) = 5$. The score for the chain *{dragon}* is 1 and for *{sword, weapon}*, 0. This gives an average chain score of 2.

Applying the Fitness Functions Each of the fitness functions above evaluates a story candidate in a different way, using a different numerical scale. A simple voting system is implemented to combine the scores when more than one fitness function is to be applied (see Algorithm B.7 in Appendix B). Each story is ranked by each of the fitness functions and then these ranking are combined to create an overall ranking for the population. This method of scoring treats each of the fitness functions with an equal weighting (i.e., one fitness function does not have more influence on selection of candidates than another). We use descending dense ranking (e.g., 100, 99, 99, 98) so as not to favour too strongly those solutions that are clones of one another (see Algorithm B.8 in Appendix B). The ranks of each member of the population are then used to decide the proportion of weighting each candidate is assigned during roulette-wheel selection.

In Section 5.3.2 we will evaluate the range of fitness functions available from the combination of these three measures of local coherence.

5.2.6 Surface Realisation

Once the search has reached a set number of generations, the fitness function is used to select the overall fittest story. This story is then transformed into English text through surface realisation. The realiser 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 realisation (Knight and Hatzivassiloglou, 1995). Ideally, the realiser should also select an appropriate tense for the sentence. However, we make the simplifying assumption that all sentences are in the present tense (see Section 3.2.4 for more details).

5.3 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.

5.3.1 Corpus Data

The generator was trained on 437 stories from the Andrew Lang fairy tales collection, the same corpus presented in Section 3.4.4. The corpus contains 15,789 word tokens, the average story length being 125.18 sentences. We discarded tokens that did not appear in the Children's Printed Word Database[3], a database of printed word frequencies as read by children aged between five and nine. From this corpus we extracted entity graphs (see Section 4.2) for 667 entities in total (this is 0.43% of the entities in the corpus), disregarding any graph that contained less than 10 nodes as too small. The graphs had on average 61.04 (SD 106.76) nodes, with an average clustering rate[4] of 0.027 (SD 0.076) which indicates that they are substantially connected.

5.3.2 Search Parameters

Considerable latitude is available when selecting parameters for the GA. These involve the population size, crossover, mutation rates, and fitness functions. To evaluate which setting was best, we asked two human evaluators to judge (on a 1–5 scale) stories produced with a population size of 1,000, 5,000, and 10,000, a crossover rate likelihood of 0.1, 0.4, and 0.6 and a mutation rate likelihood of 0.001, 0.01, and 0.1. We used 7 different fitness functions. As we have used the entity grid previously we decided to see if using additional coherence models would approve the output; entity grid, centering, lexical chains, entity grid + centering, entity grid + lexical chains, centering + lexical chains, and the combination of all three. For each run of the system a limit was set to 5,000 generations. We produced stories of length five for three different input sentences resulting in a total of 567 ($27 \times 7 \times 3$) stories that were presented to the evaluators.

The human ratings revealed that the best stories were produced for a population size of 10,000, a crossover rate of 0.1, a mutation rate of 0.1 and a fitness function consisting of the entity grids alone. Compared to previous work (e.g., Karamanis and Manurung, 2002) our crossover rate may seem low and the mutation rate high. In fact

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

[4] Clustering rate (or transitivity), Newman (2003), is the number of triangles in the graph — sets of three vertices each of which is connected to each of the others. The clustering rate is calculated using Equation 4.3.

Goldberg (1989) recommends that the mutation rate should be inversely proportional to the size of the population. However, it makes intuitive sense in our case, 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 that may improve overall coherence and interest. The evaluation showed that the best stories were generated when the entity grid was used on its own. An explanation for this could be that the system finds it too difficult to optimise in more than one dimension. The fitness landscape created by using multiple coherence metrics may simply be too ill-defined. It is also possible that lexical chains and centering measures are not well as well suited to our task as the entity grid approach. Our stories are very short, containing only 5 sentences and a small number of entities. Our centering based measure could only evaluate the story based on the centered entity of each sentence, and our lexical chain measure can only evaluate those entities that it can place within a chain. This means that neither of these measures will necessarily evaluate the usage of all of the entities within a story, although the entity grid will.

5.3.3 Evaluation

To evaluate the performance of the GA system (GA-based), we compared the stories it created against those of the following 1) the generate-and-rank system (Rank-based) described in Chapter 3 and 2) the plots as a deterministic system, without GA optimisation (Plot-based). In Chapter 4 we outlined how a plot graph can be generated for a given input sentence. Edges within this graph are weighted using Mutual Information (MI, Lin, 1998), and so traversing the graph whilst preferring the highest scoring edges will result in a single, most likely, plot from the graph. Each plot represents a set of stories containing differing lexical values for entities other than the protagonists (the main characters of the story supplied in the input sentence). These supporting entities are selected from the content selection database and are ranked based on their semantic similarity to the senses in the plot. We limit the number of entities that can be considered for each lexical variable in the plot to 5. The resulting stories are ranked using the entity grid coherence metric in order to select the best one. In addition, we included a baseline (Human-sentences) which randomly selects sentences from the training corpus of human authored fairy tales. We limited the set of possible sentences to those that contained one or both of the story protagonists (i.e., entities in the input sentence)

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*
Human-sentences	3.10	2.23*	2.20*

Table 5.1: Human evaluation results: mean story ratings for four story generators; $^{*}: p < 0.05$, $^{**}: p < 0.01$, significantly different from GA-based system.

and with a length of 12 words or less, as this was the maximum length of the sentences generated by the Rank-based system.

Each system generated stories for 12 input sentences, resulting in 48 (4×12) stories for evaluation. The sentences were created using commonly occurring entities in the fairy tales corpus (e.g., *the child watches the bird*, *the queen controls the dragon*, *the wizard casts the spell*). These 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 (56 in total) 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 favourably interesting stories, stories that were comprehensible and overall grammatical. The experiment was conducted remotely over the web using the WEBEXP[5] (Keller et al., 2009) experimental software. Instructions and materials are included in Appendix A.

5.3.4 Results

Our results are summarised in Table 5.1 which lists the average human ratings for each system. 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 5.1 for fluency, coherence, and interest.

In terms of interest, the GA-based system is significantly better than the rank-based, plot-based and human-sentence ones (using a Post-hoc Tukey test, $\alpha < 0.05$). With re-

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

gard to fluency, the Rank-based system is significantly worse than the rest ($\alpha < 0.01$). Interestingly, the sentences generated by the GA-based and plot-based systems are as fluent as those written by the human authors. Recall that the human-sentences system, simply selects sentences from the training corpus. Finally, the GA-based system is significantly more coherent than the rank-based and human-sentences 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 human-sentences 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 judgements by the grammaticality of the stories.

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 5.2 for the input sentences *the emperor rules the kingdom* and *the child watches the bird*.

The stories generated by the GA-based system appear to be coherent. The story protagonists interact in the majority of the story actions. This is because the mutation operator has the option of swapping any entity in a story for a protagonist anywhere in the story, something the generate-and-rank approach could not do, due to the way it constructs stories, sentence by sentence. It is unknown exactly how disruptive the introduction of new sentences through the mutation process is, however, the action sequences that appear in the examples of the plot-based system are, on inspection, relatively similar to those generated by the GA-based system, leading us to believe that their disruption is quite low.

The rank-based system and GA-based system both differ on their treatment of story interest whilst generating stories. The rank-based system evaluates each story using a lexical model of interest which it uses to select between different story variations as it generates each story one sentence at a time. The GA-based system, however, does not make use of an explicit model of interest and utilises its recombination and mutation functions in order to generate interesting stories from complete stories in the population of story plots. From the results we can see that the document level treatment of interest used by the GA outperforms the lexicalised treatment by the rank-based

GA-based	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-based	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-based	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.
Human-sentences	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 5.2: Stories generated by a system that uses plots and genetic search (GA-based), a system that uses only plots (Plot-based), the generate-and-rank system outlined in Chapter 3 (Rank-based) and a system that randomly pastes together sentences from the training corpus (Human-sentences).

system. As we discussed in Section 3.4.4, the use of a lexicalised model for interest when generating sentences and stories tends to result in overly long sentences in which the words used may be selected based on their perceived interest rather than on how well they complement the story as a whole. Interest in the GA's stories, however, develops from the treatment of the document as a whole, where recombination and mutation serve to change limited portions of an already established story, with the resulting mutations being required to maintain or improve the coherence of that story for it to receive a competitive fitness. One future possibility is to investigate how both lexical and document level interest could be combined. For instance, using weightings we could give preference to those options supplied to the mutation operator that have higher perceived lexical interest. This could potentially encourage the use of interesting words in the story without impacting on the evaluation of document level interest maintained by the fitness functions.

The use of GAs is also promising for scaling the generation system presented here. For instance, a story of length 10 is no more difficult to produce than a story of length 5. As the GA accepts stories from plots as its inputs, it is relatively straightforward to create the initial population. The GA operators then work identically to how they would for a shorter story. This is very different from the generate-and-rank approach where sentences had a very strong dependence on the discourse history. The plot-based system makes no use of discourse history, but rather uses sense information from action roles, to make lexical choice decisions. As the lexical items for each node of the plot graph are selected independent from one other, the sentences representing these nodes are only generated once. Indeed, whilst the plot generation system, generating stories of length 5, tends to produce sentences in the thousand, the generate-and-rank approach was in the hundreds of thousands.

When we generate plots for use in our system, we add sense information to outline which entities are to assume roles in each action. This information is used by the plot generation system to satisfy underspecified action slots. The sense information can sometimes be misleading (for example, the first sense for *scorpion* in WordNet represents a person born under the start-sign Scorpio). Lexical selection problems may however have resulted from the plot generation system's inability to make content selection decisions with document quality in mind. The mutation process used by the GA when replacing an entity node will select at random from those supplied by the content selection database. For common verbs, such as *give*, this may mean effectively selecting from hundreds of possible entities. One option to improve efficiency is to

extend the use of sense information to the mutation process so that a bias can be made towards similar words, whilst also allowing for dissimilar words to be chosen, just not as often.

Our experimental evaluation revealed that the combination of plots with genetic search leads to interesting stories even without an explicit interest rating module. However, there are some issues that come to light by evaluating stories on their coherence alone. To start with, we can only judge story quality based on the position of the entities in the story. This is useful but it tells us little about what these entities are, or if the actions performed are in fact interesting. A coherence-based fitness function will also have problems discriminating between stories in which the entity transition sequences are the same and as it is unlexicalised it does not asses whether entities fit together in a story. It is possible that an improved model of interest together with a coherence-based fitness function could provide this information. This interest model would help discern between equally coherent stories by evaluating the combination of their lexical and syntactic features. Another argument is that the coherence model completely ignores the contribution of adverbs and adjectives. These are still selected by the system but as there is no way to judge their impact on the story, they are simply chosen at random from those available.

We have not investigate in full the nature of the GA search technique as it can be applied to our domain. Our approach assumed that the fitness function remains static throughout the search whereas it could be profitable to develop a dynamic model of fitness. There is a trade-off between exploration and exploitation. The search should be initially exploitative to allow potentially rich areas of the search space to be located. Yet, this should eventually turn to exploitation, which is known as the "killer instinct": having found a lucrative area the search forces convergence upon the fittest solution in that area. Exploration is maintained in our approach though our use of roulette wheel selection, we allow weak candidates to participate in the mating process, although not as often as the fitter candidates. Another selection method we could have used is tournament selection. In tournament selection, a certain number of candidates are selected at random and then the fittest individual wins the tournament. Miller and Goldberg (1995) show that this selection process can be used to affect the selection pressure through the varying of the tournament sizes.

We compared a selection of fitness functions that evaluated coherence in the generated stories. These included measures based on the entity grid, centering and lexical chains and their combination. On the one hand, we found that system output suffered

when utilising a fitness function comprised of more than one of these measures. The intended specialisation of each fitness function is diluted when not considered in isolation. In gaining a consensus opinion from several fitness function, we lose the ability to effectively critique the stories along the desired dimensions. On the other hand, using a fitness function based on the entity grid measure alone gives many different stories the same score because they have the same distribution of entities. One option we did not consider was to employ a two-tier fitness function as in Duboue and McKeown (2002). Our GA search could then evolve the individuals in the population along different coherence measures one after another. This option would also allow us to consider the interest model as a fitness function on its own. The system could then search for the most coherent of the most interesting stories. It would also allow the addition of further evaluation models, for example, in this study we did not include a model of Latent Semantic Analysis (LSA, Foltz et al., 1998) which has been shown to be an effective predictor of coherence.

5.4 Summary of Chapter

In this chapter we have introduced evolutionary search, GAs in particular, as a means to improve the quality of the stories generated by our system. Specifically, we used stories described by plots and then applied a GA search to improve the overall coherence and flow of the story. Unlike the generate-and-rank system, the search was applied to entire stories, thereby allowing document level control to guide the story progression.

Having developed an improved method for searching the story space, we will next look at how the system can be extended. It is clear that the language used in the stories is not very natural, so we will consider referring expressions and how they can be included to improve the quality of the generated stories. We will also consider how common sense knowledge (Liu and Davenport, 2004) can be used to improve the stories. Another consideration for our approach is its portability, we will explore the impact of using a different training corpus and how stories produced by a system trained on a corpus representing a different domain, namely, news texts. We will also introduce a story completion task rather than generation an entire story from scratch. This task is more in line with how the story generator would be used in an educational setting, e.g., as an aid to students while writing their own stories.

Chapter 6

Exploring the Modularity and Portability of the Story Generation System

In the preceding chapters, we motivated a trainable story generation system. The strength of such a system lies in its portability and modular structure. The former allows it to be trained on new domains, whereas the latter enables the integration of new data sources. In this chapter we propose additional components for integration into our system. These components highlight possible areas of future work and showcase the extensibility of the pilot system. Specifically, we will look at the addition of a components for generating referring expressions and incorporating commonsense knowledge (Singh and Barry, 2003). These are added as post-processing tasks and are intended to enhance the stories generated by the GA-based system. We shall also explore the system's capability for retraining on a new domain. Finally we will introduce, a new generation task, namely automatically finishing incomplete stories.

6.1 Generating Referring Terms

One problem facing the story generation system we have described in this thesis is the quality of the text it is able to produce. In particular, we find a large amount of repetition as each entity in a story is represented by the same surface form throughout the text. Consider for example, one of the stories created by the GA-based system.

> 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 first thing we notice is that the text used to communicate this story is not very natural and it is unlikely that a human writer would have expressed the narrative in this way. To make the text more human like, it requires the use of pronouns, e.g., *the emperor* can be referred to as *he* after the first mention and *the kingdom* as *it*. Also, the story communicates no descriptive information about these entities as the current implementation always maps each entity to a single noun. However, writing a narrative requires more than simply stating each event that takes place. Take the following excerpt from the Andrew Lang fairy tale corpus (see Chapter 3.4.1) as an example:

> Many, many thousand years ago there lived a mighty King whom heaven had blessed with a clever and beautiful son. When he was only ten years old the boy was cleverer than all the King's counsellors put together, and when he was twenty he was the greatest hero in the whole kingdom.

There is clearly a gap between the human authored prose of the fairy tales and our system generated stories. Here, we discuss how pronouns can be introduced into our stories (i.e., replacing *King* with *he*) together with the generation of more elaborate referring expressions (e.g., describing the *king* as *mighty* or adding that the *counsellors* belonged to the *king*).

In the NLG pipeline (Reiter and Dale, 2000) the task of generating referring expressions falls under the purview of the microplanner. The latter is responsible for selecting the best representation for each mention of an entity within the text. Any system which generates referring expressions must also differentiate between the initial and subsequent mentions of each entity. It must consider if an entity should be represented by a simple or complex noun phrase depending on the level of detail about that entity, whether a pronoun can be used to replace the noun phrase, and finally, if the entity has a name that could be used to represent it. The system must also ensure that the reader knows which entity is being refereed to, taking care to watch for possible distractors, and making sure that the reader is not confronted with too much redundancy, resulting from the inclusion of information about entities that serves no purpose. Many approaches to generating referring expressions have been proposed in the past with particular attention to the task of pronominalisation, a selected few are discussed below.

STORYBOOK (Callaway and Lester, 2002a) incorporates an algorithm for making pronominalisation decisions as outlined in Callaway and Lester (2002b). Their approach assumes that the input to the pronominalisation component is a discourse tree

representing the narrative. This discourse tree has discourse relationships as its internal nodes and the leaves are individual sentential elements organised semantically. In order to make decisions about when to use pronouns, the algorithm must consider several factors about each entity and the rhetorical structure it appears in, including, the sentential distance and number of other entity references that have been encountered since its last occurrence, and whether the containing rhetorical structure marks a shift in topic. The algorithm clearly needs access to a discourse history for each entity to make such decisions. Callaway and Lester (2002a) conducted a human evaluation study to determine the effects of removing the components for *discourse history* (selecting definite or indefinite articles, allowing contextual references and pronominalisation), *lexical choice* (replacing repetitious entity and event mention) and *revision* (allowing revisions in sentence planning, e.g., aggregation) from STORYBOOK. They found that the removal of the module for discourse history was most detrimental to the perceived quality of the narratives, especially in terms of flow and readability. Their results show that unnatural use of referring expressions can greatly affect the quality of a generated narrative.

McCoy and Strube (1999) also propose an algorithm for making decisions on when an entity mention in a text should be a pronoun or a noun phrase. Similar to Callaway and Lester (2002a) the distance between entity mentions is considered. In particular, they found that pronouns are generally used for subsequent reference to an entity within a sentence and that a definite description is used if the last mention was several sentence previously. Their algorithm also handles multi-thread documents (a feature prevalent in the newspaper documents they were working with) where attention must be paid to whether the previous mention of the entity is within or without the current story line. One of the biggest problems for any pronominalisation algorithm is determining whether or not the use of a pronoun would create ambiguity over which entity it refers to. McCoy and Strube (1999) only use pronouns when the reference is unambiguous, which requires comparing entities in the previous sentence and the preceding text of the current sentence and comparing the gender and number of each to the current mention. If there is a competing entity present for which these features match then the use of a pronoun would lead to ambiguity. Evaluation of their algorithm on three documents from the *New York Times*, shown that it correctly identifies where pronouns should be applied 84.7% of the time.

Dale and Reiter (1995) propose the Incremental Algorithm for generating referring expressions. The input to their algorithm is an entity to be referenced along with its

known attributes (e.g., entity e_1 has the attribute set $catagory = car$, $colour = red$, $size = small$) and the set of other entities in the scene labelled with their attributes. The algorithm then iterates through the attributes for the focal entity and selects one-by-one attributes for inclusion in the reference that succeed in eliminating at least one entity from the list of distractors in the scene. Once a set of attributes is selected that results in an unambiguous reference for the entity, the algorithm terminates. The Incremental Algorithm was designed to loosely comply with Grice's maxims (Grice, 1975) of conversational implicature which state that referring expressions should be accurate, contain as few attributes as possible, be as short as possible and not add any more information than the hearer requires. Although Dale and Reiter (1995) also outline several approaches which strongly adhere to these maxims, they found that they have greater computational complexity in comparison to the Incremental Algorithm and do not necessary help explain the psychological aspects of how human speakers construct referring expressions. The Incremental Algorithm represents a much simpler model and by not placing the constraint of brevity on the generated reference is closer to human generated referring expressions.

Krahmer et al. (2003) propose a graph based method for deciding which properties of entities within a text should be included in the referring expressions. Their approach is similar to that of Dale and Reiter's (1995) although they seek to overcome the limitations placed upon the Incremental Algorithm by its inability to perform backtracking (once an attribute has been selected it cannot be removed). Their goal is to create a description of a scene in which the minimal amount of information is included to avoid ambiguity. To accomplish this, they encode the scene as a graph in which each vertex represents an entity, each edge between two distinct vertices is a relationship and an edge starting and ending on the same vertex denotes an attribute. The problem of deciding which relationships and attributes to include is reformulated as finding the best subgraph for each vertex which could not refer to any other distractors. For each vertex, a subgraph, initially the vertex itself, is iteratively constructed by adding edges until a representation without distractors is found. However, a cost function is used which stipulates that adding an edge to the subgraph will increase its cost. This ensures a minimal representation for each entity in the scene. Viethen et al. (2008) expand upon Krahmer et al.'s (2003) approach by considering the fact that humans do not always refer to objects using the most minimal description. Their approach is to learn a dynamic cost function by examining the frequency of properties that appear in hu-

man authored descriptions of scenes in the TUNA corpus[1] (Gatt et al., 2007). Adding certain edges to a subgraph under consideration may then not incur an additional cost if they were often used redundantly by the human authors.

Recent work by Hervas and Finlayson (2010) explores further the use of descriptive rather than distinctive referring expressions. They create an annotated corpus of referring expressions gathered from a selection of folk tales and news texts and find that around 18% of the annotated referring expressions were descriptive. Their research highlights the need for referring expression generation to move away from its current focus on generating purely distinct references.

Siddharthan and McKeown (2005) propose a method of acquiring and generating referring expressions for people by exploiting the redundancy in multilingual document summarisation. For this task (outlined in the DUC 2004 Multilingual summarisation task of the Document Understanding Conference[2]) 25 sets of documents in Arabic were to be summarised, with 2 machine translations of the document set, in English, being provided. Siddharthan and McKeown (2005) proposed that by being able to identify and extract the referring expressions from these documents they could also be generated in order to improve the quality of the machine translated text. For entities in the translated documents they marked up the semantic attributes for *role*, *organization*, *country*, *state*, *location*, *temporal modifier* (e.g., former, new) and also *person name*. As it can be assumed that entities with similar (although not identical) reference attributes will be the same across documents on the same topic, these entity references can be combined to create a single set of attributes for each entity. The set of attributes for each entity is then pruned, leaving the most likely reference for that entity in each of the semantic attributes shown above. From analysis of news texts Siddharthan and McKeown (2005) extracted a set of template phrasal structures based on an entity's *role* (for example, an expression referring to an ambassador may be structured as *'COUNTRY ambassador PERSON'*). In order to generate a referring expression, they then selected the best fitting template based on the entity's semantic attribute set, using those attributes to fill the template. To evaluate, they compared their model for generated referring expressions against two baselines, one which selected the first reference used for a given person in the document and another that selected a reference at random. They found that their generated reference achieved a higher BLEU score[3] than the two

[1] See `http://www.csd.abdn.ac.uk/research/tuna/`

[2] `http://duc.nist.gov`

[3] BLEU scores are often used in machine translation to judge the similarity of a machine translated document to that of human authored translations for the same document.

baselines when compared to the referring expressions in manual translations, of the same document set.

Each of the approaches described above requires a significant amount of data about the entities and the context in which they are being generated (Siddharthan and McKeown (2005) being a notable exception). Firstly, the generation of pronouns requires knowledge about the gender and number of every entity which is typically hand-coded. Secondly, approaches for generating distinct and descriptive referring expressions require knowledge about each entity's attributes and the relationships between entities. We shall now outline a procedure for generating referring expressions from data that can be automatically leveraged from a corpus.

6.1.1 Corpus-based Generation of Referring Expressions

Our referring expressions component performs post-processing on the story plans generated by the GA-based system, before they undergo surface realisation. The input is thus a story, comprising a list of dependency trees, each of which represents a sentence. The goal is to search for a more natural and descriptive representation for this story as a whole. Unlike the approaches to generating referring expressions described above, our system deals with simple stories, that are short with no shift in topic or dialogue, so we need only consider local constraints. Also, our story specifications do not make use of rhetorical relationships so we need only consider the positions of the entity mention and not its current role in the text. And finally, each of our stories assumes that there is no referential ambiguity, e.g., in a story about a *prince* there is only one *prince* and all mentions of the word *prince* refer to him.

We decompose the task into three distinct parts. Firstly we consider building a descriptive reference for the first reference to an entity, secondly we indicate which of the subsequent references can be replaced with pronouns and lastly, as in STORYBOOK's revision module (Callaway and Lester, 2002a) we replace repetitive references with semantically similar references (e.g., synonyms and hypernyms). We start by describing how the data required can be leveraged from the corpus before outlining the generation procedure.

6.1.2 Building Referring Expressions

The first step we must take is to decide which attributes can be applied to each entity. We start by looking at four types of linguistic attribute that can be easily leveraged

entity: princess	
verb:role	*description*
strut:subject	'vain princess' (4.83) 'proud princess' (2.48) 'poor princess' (0.95)
dance:subject	'wonderful princess' (4.76) 'beautiful princess' (1.52)
suffer:subject	'princess of Lombardy' (4.12) 'lost princess' (2.58) 'young princess' (1.84) 'lovely princess' (0.68)
forget:object	'weeping princess' (4.22) 'world's princess' (4.22) 'silent princess' (3.93) 'beautiful princess' (0.82)
threaten:object	'elder princess' (4.05) 'other princess' (3.13) 'eldest princess' (1.61)
expect:object	'princess Celandine' (3.45) 'princess of the island' (3.45) 'charming princess' (1.5) 'lovely princess' (0.45) 'poor princess' (0.44)

Table 6.1: Example of verb-reference relationships for the entity *princess*. Verbs are shown with their role (subject or object) and references are shown with their MI scores.

from the corpus. These are prepositional attachments (*the prince of England*), adjectives (*the happy prince*), possessives (*the prince's horse*), and appositives, including names (*prince George*). Each of these relationships is straightforward to extract from a dependency parse tree. For each entity we can then ascribe its possible attributes. However, we must also consider that we may not always want to describe an entity the same way in every story. In order to add characterisation to our stories we need to consider the type of character that the entity portrays. To do this we shall make two assumptions. Firstly, that for each document in the corpus there is only one character type represented by each entity word form, i.e., there is only one set of characteristics for a prince in the given text. Secondly, that the characteristics of a given entity are associated with the actions that they perform, e.g., a *good prince* will *save* and *protect* whilst an *evil prince* will *terrorise* and *kidnap*. We process the documents in the Andrew Lang fairy tale corpus (see Section 3.4.1) one as a time. For each entity in a document we extract its attributes and the actions it performs. We then record the relationships between actions and attributes. For each action we also take note of whether the entity was the subject or the object, i.e., descriptions of *the prince who has been kidnapped* and *the prince who is kidnapping* may differ. As we continue to process the

corpus, the actions performed will continue to be associated with additional attributes. It is also clear that we must weight the likelihood that an attribute is associated with an action. We use Mutual Information (MI, Lin, 1998) to score the relationships, which allows us to decide which attributes are most characteristic of a given action for an entity.

Selecting the best attributes for an entity then becomes a case of ranking each attribute by its MI scores for a set of actions. Selected examples of verbs and associated descriptions for the entity *princess* are shown in Table 6.1. Given the set of actions for an entity, those attributes with the highest average MI are selected. If there are no attributes attested with the actions, then we shall not generate a descriptive reference. From the top 5 attributes, we select at most 1 of each type, except for adjectives for which we allow at most 2. This gives us at most 5 references that can be combined with an entity's noun to produce a descriptive reference. If one of the references is a compound noun that indicates a name then we incorporate it directly into the reference. For each of the other attributes we perform a search to select which to include. We generate all possible combinations of reference that can be produced (ordering is only a consideration for selecting adjectives) and then score the realised version of the reference with our language model (see Section 3.2.4). This score indicates the naturalness of the reference with respect to human authored text. The highest scoring attributes are then incorporated into the story tree for the first mention of that entity.

The next step is to calculate for each entity whether they are masculine, feminine or neuter. To do this we use a database constructed by Charniak and Elsner (2009) which gives the probability of a given noun falling into one of these categories. Their work focusses on the use of Estimation Maximisation (EM) to obtain parameters for a system that resolves pronoun anaphora. One such parameter is the probability of gender given a noun. The probabilities they derive can thus be used to select the most likely gender for each noun.[4] For example, the likelihood of *prince* being male is 0.943, female 0.026 and neuter 0.031.

6.1.3 Making Reference Decisions

We now outline a simple algorithm for generating referring expressions. On the first mention of an entity we attempt to generate a descriptive reference from the actions

[4]Their system and the Ge corpus (a portion of the Penn Wall Street Journal Tree-bank which was used by Ge et al., 1998) on which it was trained are available online at `http://bllip.cs.brown.edu/download/emPronoun.tar.gz`

Algorithm 6.1 Pronoun Generation Condition

if $r \in C \wedge \neg(\exists\, x \in C$ such that $x \neq r \wedge \langle p_x, g_x, n_x\rangle = \langle p_r, g_r, n_r\rangle)$ **then**
  use a pronoun
else
  use a nonpronomial reference
end if

- C is the set of entities mentioned in the previous utterance.
- r is the internal symbol corresponding to the intended referent.
- $\langle p_r, g_r, n_r\rangle$ is a triple representing the grammatical properties of person, number and gender of r.

Figure 6.1: Reiter and Dale (2000)'s conservative pronoun generation algorithm

in the story in which the entity participates. For all subsequent mentions of an entity we decide whether to use the non-descriptive noun phrase, just the noun token, a semantically similar noun phrase or a pronoun. To decide if a pronoun can be used, we implement the conservative pronoun resolution algorithm described in Reiter and Dale (2000) and shown in Figure 6.1. This algorithm uses each entity's number, gender and person in the current and previous sentence to decide whether or not a pronoun can be used. This algorithm will only produce pronouns when the entity in question has already been encountered in the current or previous sentence and does not share attributes with any other entity in that same window which would result in an ambiguous pronoun. The number for each entity can be determined by preprocessing each sentence with the realiser to find the best representation of the sentence before pronominalisation takes place. In our stories, every entity is referred to in the third person.

The last stage in our algorithm is to monitor the types of reference used for each entity and replace references that are overused with semantically similar ones. We do this if there have been three consecutive references for the same entity that are not descriptive or a pronoun. To select a semantically similar noun to replace the entity with, we must first determine the correct sense for the entity. This can be achieved using lexical chains which we describe in Section 5.2.5. In particular, we use the procedure for generating lexical chains described in Barzilay and Elhadad (1997). A

result of generating lexical chains for the story is that the most likely entity senses will be selected to form those chains. Once we know an entity's sense, WordNet (Fellbaum, 1998) can be used to select all possible synonyms and hypernyms. However, as we must ensure the reference is not ambiguous we must also ensure that the new reference produced cannot also describe any of the other entities in the text. Using the semantic similarity score outlined in Wu and Palmer (1994) (described in detail in Section 4.2) we select the first single word reference (thus excluding compound nouns such as *male-monarch*) from the hypernym tree that is not considered semantically similar to any other entity sense in the story. This allows us to break the monotony that may occur when two entities with similar attributes but of different classes (e.g., a *prince* and a *chef*) appear together in a story preventing the use of pronouns.

6.1.4 Examples

We shall now give examples of story revisions produced by the referring expression component on both hand-written stories and those generated by the GA system (see Chapter 5). Hand-written stories (simple fairy tale stories by an adult author) can be parsed by the system into the same structural representation (an ordered list of dependency trees) that is the input to the surface realiser. Examples of hand-written stories are shown in Table 6.2 (stories A through D) and examples for the GA-system in Table 6.2 (stories E & F). For each example, the original story is given with the revised entity references generated by the component shown in bold and each reference indexed to indicate the entity they refer to.

These stories highlight several successes and failures of our prototype referring expression component. Stories A and B demonstrate the ability to generate descriptive references for several entities. We see that *the prince* has become *the young prince* or *the prince of the East* depending on the actions he performs. We also see that the system can now select names for some characters, such as *the great dragon Fafnir*. The entities that are not introduced with a descriptive reference are those for which the system does not have enough information regarding their attributes. From these stories, it is clear that the conservative pronoun algorithm is also successful. For example, *the princess* is correctly referred to as *she* and *her*. The system also avoids ambiguous pronouns. In story C we have two stereotypically male protagonists, a *prince* and a *knight*. Using a pronoun to describe either of these entities would confuse the reader as to which of them is being referred to. However, it is clear that this makes the remainder

A	The prince presents the princess to the duke. The duke leads the princess to the palace. The prince leaves the palace hunting the dragon. The duke distracts the princess with the birds. The princess welcomes the prince riding home on a horse.
	[**The young prince**]$_1$ presents [**the princess of love**]$_2$ to [the duke]$_3$. [The duke]$_3$ leads [**her**]$_2$ to [**the palace of the underworld**]$_4$. [The prince]$_1$ leaves [**it**]$_4$ hunting [the dragon]$_5$. [The duke]$_3$ distracts [the princess]$_2$ with [the birds]$_6$. [**She**]$_2$ welcomes [the prince]$_1$ riding home on [**a good horse**]$_7$.
B	The prince searches for the princess. The dragon guards the princess. The prince has a horse the dragon. The prince fights the dragon. The horse awakens the princess with a neigh. The prince saves the princess.
t	[**The prince of the East**]$_1$ searches for [**the unfortunate princess**]$_2$. [**The great dragon Fafnir**]$_3$ guards [**her**]$_2$. [The prince]$_1$ has [**a white horse**]$_4$ following [**it**]$_3$. [The prince]$_1$ fights [**it**]$_3$. [The horse]$_4$ awakens [the princess]$_2$ with [a neigh]$_5$. [The prince]$_1$ saves [**her**]$_2$.
C	The prince offends the knight. The knight demands an apology from the prince. The prince ignores the knight. The knight draws the sword on the prince. The knight duels with the prince.
	[**The handsome prince**]$_1$ offends [**the young knight**]$_2$. [The knight]$_2$ demands [an apology]$_3$ from [the prince]$_1$. [The prince]$_1$ ignores [the knight]$_2$. [The knight]$_2$ draws [**the robber's sword**]$_4$ on [the prince]$_1$. [The knight]$_2$ duels with [the prince]$_1$.
D	The chef offends the young knight. The knight demands an apology from the chef. The chef ignores the knight. The knight draws the sword on the chef. The knight duels with the chef.
	[The chef]$_1$ offends [**the young knight**]$_2$. [The knight]$_2$ demands [an apology]$_3$ from [the chef]$_1$. [The chef]$_1$ ignores [the knight]$_2$. [The knight]$_2$ draws [**the robber's sword**]$_4$ on [**the cook**]$_1$. [**The aristocrat**]$_2$ duels with [the chef]$_1$.

Table 6.2: Examples of hand-written stories with revisions performed by the referring expressions component. Revisions are shown in bold font and references are indexed by entity.

E	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 young emperor**]$_1$ rules [**the kingdom of the fairy Blue**]$_2$. [**It**]$_2$ holds on to [**him**]$_1$. [**He**]$_1$ rides out of [**it**]$_2$. [**It**]$_2$ speaks out against [**him**]$_1$. [**He**]$_1$ lies.
F	The child watches the birds. The birds weep for the child. The child begs the birds listen. The birds dress up the child. The child grows up.
	[**The poor child**]$_1$ watches [the birds]$_2$. [**They**]$_2$ weep for [**her**]$_1$. [**She**]$_1$ begs [**they**]$_2$ listen. [**They**]$_2$ dress up [**her**]$_1$. [**She**]$_1$ grows up.

Table 6.3: Examples of GA-system stories with revisions performed by the referring expressions component. Revisions are shown in bold font and references are indexed by entity.

of the story rather repetitive. Story D, shows a successful attempt to overcome this by utilising semantically similar nouns to refer to entities when pronominalisation is not appropriate. In this story we once again encounter two male entities where one of their mentions has been altered (e.g., *knight* becomes *aristocrat* and chef becomes *cook*). This approach was not viable in story C as *prince* and *knight* are considered too semantically similar, i.e., the term *aristocrat* could apply equally to both of them creating an ambiguous reference. Finally, stories E and F show the result of applying the referring expression component to the GA-based system. We see that the stories become much shorter due to the reduced number of entities, although the text appears more natural.

These stories also highlight some caveats with our approach. In story F the sentence *the birds dress up the child* is transformed to *the birds dress up her*. It is clear that the process of generating pronouns has an impact on the structure of the sentence in certain cases. Allowing further revisions to sentence ordering with the use of a language model ranker may solve this problem. Another problem highlighted is that although we can determine the most likely gender for entities within the corpus, this does not necessarily correspond with their usage in the corpus. This is once again one of the peculiarities of fictional writing in which personification is used to give animals and objects human characteristics. In story B, for example, although the assertion that the *dragon* be referred to as *it* may be technically correct, by describing the *dragon*

using a male sounding name in the previous sentence (the name *Fafnir* is associated with the entity *dragon* in the fairy tale corpus), we then expect that the correct pronoun would be *him*. The gender probabilities, from Charniak and Elsner (2009), record a 0.65 likelihood that the gender of *dragon* is neuter and 0.34 likelihood that it is male. In such situations, it may be more beneficial to use the proper name, where possible, when determining gender and making more use of that name in references after the initial description.

The algorithm we have described above is rather simplistic and plays heavily on the reduced structural nature of the stories our system is capable of producing. It is clear that further work is required in ascertaining the attributes for given entities from corpora. One approach would be to train a model using shallow text features that can make classification decisions for which type of referring expression to use for each entity mention based on examples obtained from corpora. Charniak and Elsner (2009) have already shown that features can be determined for the task of anaphora resolution. Selecting attributes for descriptive references is also a difficult task, requiring the system to determine the correct amount of information about an entity being described to the reader. Recent work by Bergsma et al. (2010) outlines how classification tasks in generation can benefit from using web-scale data and allowing portability. They also show that their model is capable of effectively ordering adjectives lists, without relying on language models.

6.2 Commonsense Knowledge

Another possible extension to the pilot system is to incorporate additional knowledge bases to the generation process. One such knowledge base is ConceptNet[5] (Liu and Davenport, 2004; Singh et al., 2004) which consists of a database of commonsense knowledge automatically extracted from human written sentences that were collected as part of the Open Mind Commonsense project (Singh and Barry, 2003). The database has already been utilised by other story generation systems. MAKEBELIEVE (Liu and Singh, 2002) generated short stories by following the causal chain resulting from a supplied input action. In particular, they focus on commonsense relationships that encode cause-and-effect assertions, such as, *a consequence of bringing in a verdict is that the defendant is nervous*. A user supplies the initial sentence to the system and a story is generated through the successive applications of causal chains. A global

[5]Available at `http://web.media.mit.edu/~hugo/conceptnet/`

Relation	*Example*
CapableOf	The *man's hand* is capable of *hold pipe*.
LoacationOf	The location of the *sculpture* is *at gallery opening*.
EffectOf	The effect of *ride bike* is *accident*.
MotivationOf	The motivation of *buy umbrella* is to *keep dry*.
UsedFor	A *computer* is used for *compute*.
SubeventOf	A subevent of *see artefact* is *learn about history*.
FirstSubeventOf	The first sub event of *start fire* is *light match*.
LastSubeventOf	The last sub event of *wake up in morning* is *stretch*.
PrerequisiteEventOf	A prerequsite of *read letter* is *open envelope*.
CapableOfReceivingAction	A *weight* can receive the action *measure with scale*.
DesireOf	The *person* desires *not be unappreciated*.
DesirousEffectOf	The *book* wants to have the effect of *learn*.
PropertyOf	A *lcd monitor* has the property *flat*.

Table 6.4: Examples of relationships in the ConceptNet database.

manager constantly evaluates the story being produced to ensure there are no cycles. In the event that no further causal chains can be found, the user may be prompted to supply the next sentence. MAKEBELIEVE also makes use of WordNet (Fellbaum, 1998) to overcome problems with matching word forms in sentences and rules, using semantic distance to locate semantically related entities.

Solis et al. (2009) also utilise ConceptNet in PICTUREBOOKS, a story generation system that generates short children's stories to describe the scene in a picture constructed by the user. Although their system generates stories in a structured manner, from 11 predefined themes, it also contains a database of concepts derived from ConceptNet that can be used to enhance the stories. They make use of the commonsence knowledge as a semantic ontology in order to link entities, attributes and events. For example, knowing that a character is performing the action *play* the ontology can inform the system of a possible location for that event. The ontology is also used in a manner similar to MAKEBELIEVE in order to build paths between desired story events, showing the causal steps that can lead from one event to the other.

6.2.1 Integrating Commonsense Knowledge

We created a prototype component for integrating common-sense information into the stories generated by the pilot system. It works by incorporating sentence fragments extracted from ConceptNet into the generated stories. Specifically, we add this as a post-processing step after the story plan is generated but before surface realisation takes place. The purpose of this post-processing task is to introduce additional information into the generated stories with an eye to making them more entertaining, or simply to aid in justifying the events that place.

We started by extracting each concept from the database. The ConceptNet database contains 1.6 million assertions of commonsense knowledge, linking together 300,000 nodes which represent entities and facts. These assertions fall into 20 different categories, 13 of which we shall focus on in this chapter. Those 7 categories that we omit are simple semantic relationships, e.g., *a dog is a canine* or *a wheel is part of a car*. The relationships we used along with examples are shown in Table 6.4. In total the 13 relationship types consist of 24,981 unique commonsense assertions. Each assertion in the ConceptNet database, consists of a rule type and then two sentence fragments describing the two concepts, the antecedent and consequent, that it connects. For example, *(MotivationOf "buy fresh fruit and vegetable" "eat fresh fruit and vegetable")*. To transform these database entries into a format compatible with our system we parsed them with RASP (Briscoe and Carroll, 2002) and record the POS tags and dependency structures. It is clear that problems will arise due to ambiguity as we are parsings sentence fragments rather than complete sentences. For example, without a sentence from which to determine context, the word *play* could easily be a noun or a verb. We evaluated the performance of the RASP on the 200 of the concept sentences and found that it achieved a precision[6] of 0.9 in terms of correct POS tagging. From the dependency parse of each relationship's antecedent we selected the word at the root of the parse to be the indexing term. For example, the root for *buy fresh fruit and vegetable* will be the verb *buy*. The consequent of each rule is then parsed by our system's sentence planner (see Section 3.2.3) to build a dependency tree using the rules from our subcatagorisation grammar.

Once the story generator has produced a story plan, the commonsense module can start to search through the database of rules to find those which are applicable. For this prototype, we assume that each sentence in the story that has one or more applicable

[6]Precision is defined here as the number of correctly classified word types.

Original Sentence: *The prince marries the princess*.		
keyword	Commonsense rule	Enhanced Sentence
prince	CapableOf *succeed king*	*The prince, who succeeds the king, marries the princess.*
	LocationOf *in castle*	*The prince, who is in the castle, marries the princess.*
	LocationOf *in England*	*The prince, who is in England, marries the princess.*
princess	CapableOf *kiss frog*	*The prince marries the princess, who kisses the frogs.*
	CapableOf *live in castle*	*The prince marries the princess, who lives in the castle.*
marry	MotivationOf *start family*	*The prince wants to start a family. The prince marries the princess.*

Table 6.5: An example of the variations created by the commonsense module for the sentence *The prince marries the princess*.

entries in the commonsense database must have one of those rules applied to it. Ideally, we would want to search all possible combinations of rule applications to find the best overall, but as this is a prototype, we restrict it to one rule per sentence. The entries in the commonsense database can be split into two categories, namely those that apply to entities and those that apply to actions. We therefore look for those relationships pertaining to the nouns and verbs in each sentence. These two sets will also vary in the way they can be applied to the story. Entries pertaining to entities provide descriptions, giving an additional clause containing information on the characteristics of that entity, whereas relationships pertaining to actions will include motivations or consequences, involving the insertion of a new sentence either before or after the sentence, respectively. To reduce the amount of search conducted we restrict the application of entity based rules to the first mention of an entity in the story. This can be likened to a description used to introduce a new entity in a story.

For each relationship type we developed a template for its application (e.g., the rule *(LocationOf entity location)* will become *entity be in location*). In general, these templates consist of a simple phrase for the root of the sentence to which the consequent's dependency tree is attached (i.e., the root of a *location* sentence will always be

entity be in location, but *location* may represent a larger dependency tree). As mentioned earlier, the relationships we consider fall into two categories, those pertaining to nouns and verbs. The templates used for noun rules are dependant clauses to be embedded into a sentence whereas those for verbs will become new sentences. The process works one sentence at a time. The nouns and verbs are first identified and used to lookup the database. For each entry in the database that is associated with one of the nouns, the templates are used to construct a clause with the noun as the subject of its matrix verb. This clause will be attached to the noun's node in the dependency tree for the sentence. For each verb a similar process applies: templates are used to construct sentences for each applicable database relationship. Rather than being integrated into the current sentence, this new sentence will either be inserted before or after it in the story, depending on the type of rule used to construct the new sentence. It is likely that there will be more than one possible sentence revision or additional sentences for inclusion in the current story. The system must therefore search for the best one to use in the final version and we do this using a language model (see Section 3.2.4). This process continues for each sentence in the original story, the final updated version of the story is then sent to the surface realiser to undergo surface realisation. Limited pronominalisation is also performed when dependant clauses are generated for entities. For example, compare *the prince, the prince succeeds the king, is in the castle* with *the prince, who succeeds the king, is in the castle*. Another possibility is that the enhanced story could then undergo further post-processing in terms of the referring expressions component described in the previous section.

In order to make full use of the data, we utilise WordNet (Fellbaum, 1998) to search for terms semantically related to those in the text that do not have associated entries in the database. For nouns we search first through synonyms and then hypernyms. Therefore, if the entity *prince* had no associated relationships it could inherit those of *aristocrat*. This is useful as most of the concepts center around more abstract entities, the most common being *person* and *object*. In order to reduce the problem of ambiguity, we first perform word sense disambiguation upon the story using lexical chains (Barzilay and Elhadad, 1997) as described in the previous section.

6.2.2 Examples

Table 6.5 shows the resulting variations for the sentence *the prince marries the princess.* Here, we see examples of both how knowledge about entities, e.g., *the prince is in the*

castle, and how the motivation for an action is conjectured, e.g., the *prince marries* because *he wants to have a family*. We can also see from this example that both everyday and fictional expectations are encoded in the commonsense database Although we can expect in real life to find *princes* in *castles*, we would not really expect to find *princesses kissing frogs* outside of the fairy tale domain. This shows that human commonsense knowledge is not restricted to the real world, making such a database suitable for the production of fiction.

To examine the commonsense module, we used several hand-written stories along with several stories generated by the GA-based system (see Chapter 5). Figure 6.6 shows the original version of each story along with a revised version generated by the component, the revisions are shown in bold font. Story A is hand-written whilst B and C were generated by the GA. The stories generated by the GA system usually contain only two entities, so we also included a hand-written story, parsed by the system on input, to evaluate the effect of introducing new entities throughout the story. In story A a new entity is introduced in each sentence and an entity-based relationship is included in each line. Although several of these seem appropriate, e.g., *the duke who is in the country*, others seem out of place, such as *the palace which can be divided into rooms*. Stories B and C, were generated by the GA-based system and show how the component is capable of extending stories by adding sentences indicating motivation or effect. Several of these inserts help add context to the sentence. For instance, having the sentence *the emperor wants to go* before *the emperor rides out of the kingdom* renders the narrative more natural. However, others seem to complicate the reasoning behind actions performed by the story entities. For example, *the kingdom wants to lose consciousness* is inserted before *the kingdom holds on to the emperor* as our current approach does not take the context into account when searching the database and the information that the system is attempting to express is that *holding your breath leads to you losing consciousness*. Clearly more work is needed in deciding which commonsense relationships are appropriate for which context.

From these examples we can see how integrating off-the-shelf knowledge sources has the potential to enhance the stories that the pilot system system is capable of creating. For example, using commonsense knowledge could improve characterisation by providing additional information about story entities and attempt to explain the reasons and the consequences of the actions they perform. However, our pilot component only illustrates how commonsense information could be incorporated as a post-processing task, the information contained in the database was not used to generate the stories,

A	The prince presents the princess to the duke. The duke leads the princess to the palace. The prince leaves the palace hunting the dragon. The duke distracts the princess with the birds. The princess welcomes the prince riding home on a horse.
	The prince presents the princess to the duke**, who is in the country**. The duke leads the princess to the palace**, which can be divided into rooms**. The prince leaves the palace hunting the dragon**, which is fictional**. The duke distracts the princess with the birds**, which are in the air**. The princess welcomes the prince riding home on a horse**, who is in the country**.
B	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 emperor**, who is at the school,** rules the kingdom. **The kingdom wants to lose consciousness.** The kingdom holds on to the emperor. **The emperor wants to go.** The emperor rides out of the kingdom. **The kingdom wants to illustrate the point.** The kingdom speaks out against the emperor. **The emperor wants to have a rest.** The emperor lies.
C	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.
	The child**, who is in the school,** watches the bird. The bird weeps for the child. **The bird wants to hear.** The child begs the bird to listen. The bird dresses up the child. **The child wants to eat.** The child grows up.

Table 6.6: Examples of stories before and after processing with the commonsense component. Inserts generated by the component are shown in bold font.

only to ornament them. Being able to integrate this knowledge at the generation stage could provide additional benefits, as it would allow causal chains to be integrated into the stories from the start, thus resulting in more believable stories (as shown in MAKEBELIEVE (Liu and Singh, 2002)).

Another aspect to consider is the portability of the commonsense database. Does the commonsense information stored in ConceptNet really transcend genres? Is it fair, for instance, to say in a fairy tale setting that dragons are fictional, or that real-life princesses kiss frogs, or is this likely to spoil the story? Currently, we have used a language model when selecting between possible sentence variations, but a better approach may be to include a measure of the information content and the probability that the inserted information conforms to the expectations of the story and genre.

6.3 Exploring Portability

Throughout this thesis we argued that our story generation system has a high degree of portability due to the fact that it can construct its own knowledge base directly from text. This allows it to develop new knowledge bases for new corpora or even new domains. To illustrate the portability of the system, we retrained our system on a corpus from a new domain, namely news texts, and evaluate the stories it generates. Retraining the system requires constructing a new content selection database and story plot database. We did this on a corpus consisting of 485 documents from the British National Corpus[7] that have been designated *world news*. These texts are representative of articles from newspapers, relating to social and geopolitical topics (i.e., they do not contain business or finance topics). The average length of a news text is 1,399.45 sentences (SD 2,036.21) with an average sentence length of 24.34 words (SD 15.7) and there are approximately 16,700,000 word tokens present in the corpus consisting of 184,455 word types. Each document was parsed using RASP (Briscoe and Carroll, 2002). The parser computes part of speech (POS) tags for each word and then through syntactic analysis outputs the grammatical relations that appear within the text. As with the fairy tales, we discarded word tokens that did not appear in the Children's Printed Word Database[8] and only retained relationships that appeared at least 5 times in the corpus and had a MI (Mutual Information (Lin, 1998)) score greater than 0. We also only stored common-nouns, proper-nouns and pronouns were discarded; we also

[7]Available from `http://www.natcorp.ox.ac.uk/`

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

Noun	Subject	Object	Adjective	Total
people	7214 (301)	4474 (261)	3051 (155)	14739
man	4906 (358)	3906 (266)	2592 (205)	11404
government	5067 (283)	3138 (228)	3105 (58)	11310
minister	4826 (218)	2816 (143)	3323 (55)	10965
way	2139 (237)	6353 (248)	1868 (167)	10360
year	2062 (220)	6099 (275)	1812 (82)	9973
member	4228 (255)	3462 (155)	1469 (110)	9159
part	1125 (161)	5644 (163)	1923 (121)	8692
state	1723 (194)	2578 (225)	3877 (81)	8178
country	2150 (183)	3622 (255)	2001 (75)	7773

Table 6.7: Properties of relationships for the most frequent entities in the news text corpus. The number of occurrences for each relationship is reported with the number of tokens it relates to shown in parentheses.

split compound nouns. Verbs were stored in their lemmatised word form.

As expected, news texts are substantially different from fairy tales. In total, the news text corpus contains 3,058 noun tokens, 1,358 verbs, 300 adverbs and 1,019 adjectives. In Table 6.7 we see the 10 entities that appear most frequently in the corpus, and in Table 6.8 the top 10 verbs. Tables 6.7 and 6.8 also show the number and types of relationships for each of these most frequent words. For example, the entity *people* appears 7,214 times as the subject of 301 verbs, and the verb *be* appears 65,283 times with 1,380 different subjects. The size of the database trained on the news texts is therefore substantially larger than that trained on fairy tales. For example, *be* is the most frequent verb for both corpora, yet its occurrence in the news text corpus is approximately 10 times that of the fairy tales. Although the fairy tale corpus contained frequent references to many character entities such as, *king*, *prince* and *princess*, the news text corpus refers more to institutions, e.g., *state*, *government* and *country*. However, many of the predominant fairy tale entities also reoccur in news text, or rather their real world counterparts do. Table 6.9 shows the top and bottom subject, object and adjective relationships for the entities *king*, *prince* and *princess*, which we presented in Table 3.6 for the fairy tale corpus. Looking at specific entities within the database we can see how well the MI scores capture the stereotypical relationships within the corpus. In Table 6.9 we see that many of the relationships are just as we

Verb	Subject	Object	Adverb	Total
be	65283 (1380)	107066 (1205)	19936 (101)	192285
have	9654 (783)	25035 (876)	4806 (73)	39495
take	5705 (638)	16610 (624)	2731 (82)	25046
make	6333 (740)	13957 (817)	2574 (93)	22864
give	3658 (601)	11057 (699)	1520 (87)	16235
go	4215 (618)	5873 (688)	2597 (95)	12685
come	4287 (677)	6241 (637)	1870 (98)	12398
see	2356 (517)	6828 (799)	2116 (64)	11300
become	3303 (628)	5761 (555)	1680 (60)	10744
say	4433 (305)	3321 (525)	1541 (60)	9295

Table 6.8: Properties of relationships for the most frequent verbs in the news text corpus. The number of occurrences for each relationship is reported with the number of tokens it relates to shown in parentheses.

would expect from real-life royalty, for instance, the *king* can *knight* and *reward* whilst the princess can be *booked* or *cared* for. Changing the domain of the training text has therefore not damaged the potential of the system to generate coherent narratives. We also retrained the database of story plots. Entity graphs were collected for the news text corpus, discarding any that contained less than 10 nodes. There were 1,740 entity graphs in total (this is 57% of the entities in the content selection database) with an average of 112.76 (SD 172.61) nodes. The average clustering rate[9] is 0.031 (SD 0.067).

6.3.1 Examples

Using the retrained knowledge bases, we ran both the Rank-based system and the GA-based system, as described in Chapters 3 & 5 respectively, to gain insight into the different types of story the systems are capable of producing. We used a range of different input sentences, including some containing entities that are predominant in the news text corpus, e.g., *the parliament investigates the minister* and *the army surrounds the city*. For the Rank-based system, we used the setup described in Section 3.4.4,

[9] Also known as transitivity (Newman, 2003). It is the number of triangles in the graph sets of three vertices each of which is connected to each of the others. It measures whether nodes in the plot graphs are densely connected.

Noun	Subject	Object	Adjective
king	pardon (5.0)	style (4.6)	conquering (4.2)
	grumble (5.0)	crown (4.5)	danish (4.0)
	reward (4.4)	pardon (4.5)	norman (3.9)
	disgust (4.4)	reign (4.2)	fearful (3.9)
	knight (4.1)	dine (3.6)	saxon (3.9)
	...	...	...
	stay (0.14)	need (0.07)	eastern (0.26)
	sit (0.14)	leave (0.06)	bad (0.22)
	take (0.13)	welcome (0.04)	roman (0.14)
	come (0.01)	lie (0.02)	various (0.11)
	add (0.01)	talk (0.01)	european (0.04)
prince	pin (4.1)	style (6.3)	murdered (4.5)
	shelter (4.1)	bend (4.3)	lay (4.1)
	pronounce (4.1)	strip (3.6)	keen (3.3)
	punish (3.6)	tear (3.4)	christian (3.3)
	wander (3.5)	wish (3.2)	powerful (3.2)
	...	...	...
	need (0.45)	pay (0.38)	english (0.75)
	tell (0.44)	come (0.34)	american (0.62)
	do (0.36)	receive (0.28)	old (0.23)
	have (0.28)	see (0.26)	local (0.16)
	work (0.01)	be (0.11)	european (0.1)
princess	fast (7.2)	please (5.5)	split (6.5)
	nod (5.3)	book (5.1)	deaf (5.8)
	cheer (5.2)	slip (4.1)	scots (5.3)
	crown (5.0)	care (4.1)	pretty (5.2)
	tour (4.8)	beat (3.8)	muslim (4.7)
	...	...	...
	appear (0.58)	lead (1.2)	english (2.4)
	lead (0.37)	get (0.92)	special (2.3)
	get (0.35)	provide (0.77)	young (1.9)
	begin (0.2)	become (0.66)	british (1.3)
	see (0.15)	go (0.64)	new (0.46)

Table 6.9: Examples of the highest and lowest scoring relationships (with MI scores shown in parentheses) for character entities in the news text corpus.

Rank-based	GA-based
The parliament investigates the minister. The minister introduces the order in the streaming. The minister meets for a year in the order. The minister sours the happiness for a year. The minister plans a crisis for a year.	The parliament investigates the minister. The minister blocks parliament. The parliament presses the minister. The minister steps out of the parliament. The parliament honours.
The president signs the bill. The bill seeks the sustenance from the president. The president agrees with the relay on the bill. The bill orders that the eagle circles the kill around the globe. The president honours the ponies for the deal vetting the people with a copy.	The president signs the bill. The bill aims for the president. The president calls for the bill. The bill concerns the president. The president offers to control.
The queen marries the king. The king orders the property into giving. The king lives in the property for the year. The king places an advert out of the property. The king raises the money out of the pup.	The queen marries the king. The king has granted the queen. The queen faithfully sends for the king. The king crowns the queen. Surely, the queen lives.

Table 6.10: Examples of stories generated by the Rank-based and GA-based systems trained on a corpus of world news texts from the British National Corpus.

this includes a scoring measure applying simultaneously entity grid coherence and our interest model trained on human interest ratings of Æsop's fables. A limit of 5 choices was imposed on the system when making selection decisions and a beam of size 500 was used at each level of the story search. For the GA-based system the parameters used were the same as for our experiments in Section 5.3. We used a population of 10,000 stories initialised from plots, at a crossover rate of 0.1, a mutation rate of 0.1 and with a fitness function based on entity grids. Each run of the system was for 5,000 generations and each story 5 sentences long.

Table 6.10 shows a selection of stories generated by each system. As we can see the language is very different from that of the fairy tales. The actions that appear are much more formal, as is expected from a newspaper article, e.g., *the minister blocks* and *the president honours*. We once again see the difference in the stories generated by the Rank-based and GA-based systems. The interest model used by the Rank-based system has selected words which it regards as interesting, e.g., *happiness* and *kill*, although it seems to have had less of an impact as the resulting sentences are not quite as extreme as those produced from the fairy tale corpus. The reason for this could be that the features of predominant words in the news text corpus are generally different from those in the fairy tales and fables, i.e, it is less likely for highly imaginative words to be present. There are, however, still instances where the interest model leads to complex sentence structures, e.g., *the bill orders that the eagle circles the kill around the globe*. In contrast, the stories generated by the GA are shorter with fewer entities and they retain a clear progression of actions. We see in one of the examples in Table 6.10 a story in which a conflict occurs between a *minister* and *parliament*. It is clear from these examples that the system can generate stories that are not tied to the corpus on which it is trained. We can easily retrain on new corpora, producing stories that are indicative of new genres, yet are similar in structure to those it has generated before. We can see that the content selection decisions made using the retrained knowledge bases helps capture the language expectations of the new genre.

6.4 Completing Unfinished Stories

We have mentioned that one of the practical uses of story generation is for inclusion in interactive tutoring systems, giving prompts for students as they are writing their own stories. To give an example of our system's potential for such integration, we present a new task in which the system is presented with an incomplete story for which it has

to generate the missing sentences. This task is different from generating a whole story as the sentences created by the system must be compatible with the human authored portions of the story.

The input to the system is a story containing 6 sentences of which the 4th and 5th are missing. The human-authored sentences are parsed by the system and represented as dependency trees, blank sentences in the input are those the system must generate. We will complete the stories with the GA-based system (see Chapter 5). The process of generating the initial population however will be different as the sentences used for building the plot graph (see Chapter 4) and initialising the search of that graph will be different. As before we use the first sentence in the story to identify the protagonists and create a plot graph by merging together their entity graphs. However, as we already have several sentences of the story supplied by the user, we need to start searching the plot graph from the last sentence before the missing lines. As each node in the plot graph encodes a unique action, there will only be one node which represents that sentence and it is from this node that we begin a search of the plot graph. The GA-based system starts with a population of 10,000 stories initialised from the possible plot, a crossover rate of 0.1 a mutation rate of 0.1 and each run of the system was for 5,000 generations. The system we used was trained on the Andrew Lang fairy tale corpus (see Section 3.4.1). Restrictions are placed on the GA's operators so that only the missing sentences can be mutated and crossover can only be applied between those sentences. We investigated the effect of two different fitness functions on the types of story the GA could produce. We have shown previously that the entity grid measure is capable of producing short interesting stories, however, the task presented here is different from generating a story from scratch. Rather than finding those sentences that fit together sequentially, the story completion task also requires the system to consider globally how the new sentences will fit in with the already established context that has been provided by the user. We propose that lexical chains (see Section 5.2.5) provide a suitable fitness function for this task as they compare the placement of entities throughout the document, scoring highly, those that are more semantically related. To this end the system generates two stories for each incomplete story, one using entity grids as a fitness function and the other lexical chains.

Table 6.11 shows example stories with the system generated lines in bold font. From these stories we see that the system is capable of generating new stories to appear in an already determined context. From inspection it appears that the lexical chains fitness function is more suited to this task than the entity grid. One possibility for this is

Entity Grid	Lexical Chains
The prince loves the princess. The prince introduces the princess to the queen. The queen hates the princess. **The princess recognises the horse. The princess means for the horse to prefer.** The princess forgives the queen.	The prince loves the princess. The prince introduces the princess to the queen. The queen hates the princess. **The princess tells the queen when the kingdom saves the prince. The prince writes to the princess that the queen gives up.** The princess forgives the queen.
The queen catches the bird. The bird sings a song. The queen takes the bird to the court. **The bird teaches the manners that the court makes up. The bird reminds the day when the boat stops off at the door.** The bird flies away.	The queen catches the bird. The bird sings a song. The queen takes the bird to the court. **The queen discovers that the bird hides out. The bird helps with the queen.** The bird flies away.
The knight rides the horse. The knight brings back the treasure. The horse is very tired. **The horse speaks about the subject when the wizard enters the room. The horse tells the knight about the room surprising the attendant.** The knight escapes on the horse.	The knight rides the horse. The knight brings back the treasure. The horse is very tired. **The knight goes to fetch the horse. The knight lets up on the horse.** The knight escapes on the horse.

Table 6.11: Output of the story completion task generated by the GA system using entity grid and lexical chain fitness functions. New sentences generated by the system are shown in bold font.

that the entity grid only considers the transitions between pairs of adjacent sentences whereas the lexical chains consider entity relations more globally. This allows the lexical chains to capture longer distance relationships than that of the entity grid. As we can see in the story generated by the lexical chain GA for the story beginning *the prince loves the princess*, the newly generated stories contain all three of the entities that appear in the human authored sentences. The entity grid GA however has difficulty in evaluating all of the entities that appear in the sentence, from those longer sentences that are generated from story plots: there may be no fitness increase in changing the sentence's structure. These stories are good examples of how the knowledge contained in the system's database can be used to hypothesise missing story content. By providing a context for the story generation, the system has the ability to better define the entities that will appear in the story and the sequence of actions to develop.

6.5 Summary of Chapter

In this chapter we motivated profitable areas of future work arising from the extensible and modular platform of the pilot story generation system. We presented a series of modular extensions, including two post-processing components, one for generating referring expressions and another of integrating commonsense knowledge into the generated story. The referring expressions component highlights the potential for incorporating automatically generated descriptive references, pronouns and semantically similar noun phrases to the pilot system's stories. It utilises a database of descriptions that have been extracted from the training corpus allowing further characterisation of the story entities. The commonsense component illustrates how additional information could be integrated into the story to provide further details about story entities and actions. We also explored the portability of the system by retraining it on a corpus of news texts, representing a new domain. In addition, we showed how the pilot system can be used for a new task, i.e., finishing incomplete stories. Overall, we have shown that our system provides an excellent platform for developing extensions to improve the quality of the generated stories and for use in new tasks and domains.

Chapter 7

Conclusions and Future Directions

In this chapter we summarise the main findings and contributions of the thesis and indicate possible areas of future research.

7.1 Main Findings

In this thesis we have proposed a novel approach to story generation, outlining an end-to-end system which stochastically realises the components of the natural language pipeline. An outline of our major findings and contributions is given below:

1. We have developed an end-to-end system for generating short narratives which consists of trainable components. The strength of our system lies in its simplicity, as it does not require large amounts of hand-annotated data, thus reducing the work load of the system developer. In particular, we developed components for content selection, sentence planning, document planning and surface realisation and have shown how they can be easily integrated to form an end-to-end natural language generation system. We have described an approach to extracting a knowledge base from a corpus that encodes the relationships between entities and actions in order to capture the content of a given corpus. From this knowledge base, sentences and stories can be constructed based on the likelihoods of entities and their interactivity appearing in the corpus. Our approach is therefore bottom-up, a story is generated from the simplest lexical structures, in contrast to previous story generation systems which have a top-down approach to document planning. We have presented an evaluation method that, for the first time, allows us to establish the state of the art in the field. The story generation system we

have developed is thus state-of-the-art and can be used as a suitable baseline for comparison against future developments.

2. We have formulated the story generation task as a search problem. In particular, we present a viable bottom-up story generator that does not rely on rhetorical or other document level structures. Our system finds the best story overall by searching through possible sentences and the stories that result from their combination. Searching this large space is made possible through evaluation functions that use shallow document features. This allows the system to perform on-line evaluation of a large number of possible stories. We have shown that local coherence can be used to judge the quality of generated stories, making use of the entity grid approach proposed in Barzilay and Lapata (2008).

3. In addition to evaluating coherence of our generated texts, we have proposed a novel model for evaluating story interest. This model is trained using human interest judgements for well known human authored stories and also uses shallow document features, making it suitable for automatic evaluation. Using publicly available psychological resources the model can be trained to identify those features that indicate whether a story is interesting. We have shown that in its early stages such a model has the ability to discern between lexical choice decisions that improve story quality.

4. We have compared two different search methodologies for traversing the story space. The first is an over-generate and rank approach which creates sentences for stories sequentially, maintaining a cache of the best stories, selected by the evaluation models, which the system will then extend by generating the next sentence. This search approach considers both partial and complete stories throughout the generation procedure. We also outline a genetic algorithm search approach which constantly maintains and optimises a pool of complete stories. Optimisation of the stories occurs through the repeated application of crossover and mutation operators on individual story candidates that are selected based on their fitness (as operationalised by their local coherence). We have shown experimentally that the human participants found the stories generated by the GA-based system more interesting than those generated by the rank-based approach.

5. We reported the first work to induce story plots from corpora and to use these

as a basis for generating novel stories. We show that from a given corpus it is possible to build a graph which encodes that expected action sequences of a given entity. Story plots are developed from the overlap resulting from the merger of two (or more) such graphs, in particular from those actions in which the story protagonists interact. We used WordNet (Fellbaum, 1998) to aid in the graph merging by identifying non-identical yet semantically similar entities. Each story plot we generate represents a schema encoding a large number of possible stories depending on those supplementary entities that will appear in the final story.

6. Additionally, we have shown that our system provides an excellent platform for future work. We presented several pilot components and new tasks that highlight the extensibility, scalability and portability of our story generation system. In particular, we have shown how attributes pertaining to story entities can be extracted and used to model different character stereotypes based on the actions they perform in the corpus. These attributes can then be incorporated into a given story to provide descriptions of the story entities, making the final text more ornate. Pronominalisation can also be incorporated by identifying the gender of each entity, using attributes derived from pronoun resolution (Charniak and Elsner, 2009).

7.2 Future Research Directions

In this thesis we have focussed on a modular story generation system in which a bottom-up, data lean approach was championed. We have developed a system that represents a first attempt at approaching the task of story generation from this angle. There are several points to be considered for the development of such a system, 1) the level of document structuring used to represent each story, 2) how knowledge-bases for the story generator can be developed from available corpora, 3) the methodology for generating and searching the space of possible stories that can be encoded in the knowledge bases, and 4) what off-the shelf components are available to reduce the work load of the developer. This thesis has sought to answer these questions but future research is required to improve the quality of the stories that can be produced.

Most generation systems use rhetorical relations for document structuring. Adding rhetorical knowledge would allow our system to consider the way in which phrases

relate to one another, so that their placement in the final text can be decided. This would also allow for aggregation, which is currently out of the system's reach as it would require the relationships between phrases to be labelled for the correct lexical connectives to be selected. The challenge here is in automatically extracting the rhetorical knowledge from the corpus and learning the relationships that occur between story propositions. Despite initial efforts at rhetorical parsing the results are still quite poor (Marcu, 2000; Soricut and Marcu, 2003).

This thesis proposed that stories can be evaluated based on interest and local coherence. We defined interest in terms of shallow document features, yet it could be argued that this interpretation of interest is too simplistic and that there are other document level features should be considered when evaluating a story's interest. Pérez y Pérez and Sharples's (2001) story generation system MEXICA, for instance, records the level of tension resulting from the application of sequences of actions. Bae and Young (2008) propose a system that produces surprise using foreshadowing and flashbacks, by manipulating the action sequence of the story. However, any document level evaluation of interest needs to be automatic to allow the assessment of large numbers of candidate stories. MEXICA required that the level of tension be explicitly encoded by the human developer and Bae and Young (2008) utilised planning rules to identify the relationships between actions. Developing automatic evaluation models for interest in terms of suspense, humour, imagery, conflict, and so on, requires more sophisticated methods than we have presented here, and requires, potentially, extensive human involvement. However, their development would open the door to potentially new applications, such as poetry or screenplay generation.

Future work may also focus on how document level evaluation could be incorporated to allow stories to be generated that communicate specific messages or meet specific communication goals. Such stories may include parables or forms of poetry. This would represent an ability beyond that of the system we have presented in the thesis. Recent work has already shown the potential for future development along these lines. Manurung (2003) developed an evolutionary search approach for generating a poetic representation for a given document content specification. One option would be to incorporate their work as a post-processing step that would provide the potential for presenting our generated stories in new media.

Developing automatic methods for instilling morals and other communicative goals represents a much more difficult task as it requires us to know exactly how the contents of a story culminate to express them. In order to generate such stories automatically,

we would first need to develop a method of analysing stories to extract, not only the set characters and actions, but also the changing states of the story world and each character's beliefs, goals and feelings. These are not necessarily obvious from the text of a story, as they will be implied and require substantial world knowledge in order to predict them. An additional challenge would then be in designing a database to store the extracted document structures. One requirement is that the extracted information would have to be general enough so that it could be used in different ways to create novel stories (not simply retelling the same stories that were in the corpus). The concept would ideally be similar to the problem solving TRAMs used by Turner (1992), which encode solutions to specific problems that are then abstracted so they can be used in other genres.

One of the main strengths of the system we have proposed is its modular structure which allows additional components to be integrated, as well as new knowledge to be incorporated into the stories. Recent years have seen the development of several psychological and linguistic knowledge bases which have the potential to enhance the quality of our generated stories. The FrameNet project (Baker et al., 1998) is one such source of knowledge. Its function is to gather together semantic knowledge about real world actions, stipulating relationships (e.g., one action inherits from another or causes another action) and orderings between them. Each action frame also encodes information about the entities that participate, e.g., *driving* involves a *driver* and a *vehicle* with possible *cargo*. The knowledge contained in FrameNet could be used by the system to make better decisions about which actions to include in a story, using semantic knowledge to develop causation relationships between story actions and ensure that story entities appear in the appropriate semantic roles. Similarly, having the ability to determine semantic information in a document would allow the system to construct more accurately the relationships between story entities, building profiles that can be called upon to make inferences about their behaviours. The semantic networks produced by ASKNET (Harrington and Clark, 2008) contain nodes representing entities in a text and labelled relationships between them. To construct their networks, they start by performing Named Entity Recognition identifying entities and their semantic categories, e.g., *person*, *organisation*, *location*. Then using BOXER (Bos et al., 2004) they extract the document's underlying semantic representation in first order logic. ASKNET can then translate this output into one or more networks by identifying which variables in the semantic representation refer to one another. Through the application of spreading activation theory they are able to successfully combine network fragments, allowing

long distance semantic relationships between entities to be established. These semantic networks can potentially be used to generate improved story plots, through inference over networks containing intended story protagonists.

Another improvement to the current system could be the incorporation of an explicit model of temporal ordering (Bramsen et al., 2006; Chambers and Jurafsky, 2008). Human authored texts do not always represent events in chronological ordering, e.g., one character may tell another what they saw the day before. In order to generate more elaborate stories, the system will require the ability to deduce the actual ordering of events in the original corpus as well as the ability to produce an appropriate action ordering in the stories it generates.

Due to their potential for integration in interactive systems, agent-based approaches to story generation are very much the focus of current research. Ideally, the data-lean approach we have described in this thesis could be extended to provide knowledge bases for such systems too. Agent-based systems are attractive not only for their interactivity but also in that they allow a creative means of developing stories in which logical action decisions by story characters are separate from their internal planning mechanisms and world knowledge. These systems also provide capabilities currently out of the reach of our system, including the use of dialogue, through the transferral of story-world knowledge, and also emotion. However, there is one major hurdle that must be overcome before development of such a system could begin. An automatic method of leveraging the knowledge required for an agent-based rule base from the corpus would be required. Story agents require access to planning rules but these are difficult to learn from written text, at least in an unsupervised fashion, without access to annotated data. Human writers do not always state explicitly all the preconditions and implications a performed action will have, the author relies on human real-life experience to fill-in the missing information. This makes it difficult for planning rules, to be written automatically. However, one avenue of research may be the use of inference systems for planning that can perform automated planning on incomplete world knowledge (Eiter et al., 2000) using only those propositions that can be extracted from the corpus.

Appendix A

Æsop's Fables Evaluation

This appendix contains the instructions presented to participants in our elicitation study on Æsop's Fables (see Section 3.3.2) and the evaluation studies for comparing the output story generation systems (see Chapter 3 and Chapter 5).

A.1 Instructions for Elicitation Study of Æsop's Fables

In this experiment you will be asked to judge a set of short stories. These stories are fables and you may be familiar with some of them. A fable is a brief, succinct story, that features animals, plants, inanimate objects, or forces of nature which are anthropomorphised (given human qualities), and that illustrates a moral lesson (a "moral") which may sometimes be expressed explicitly at the end of the story. After reading each story you will be presented with a number of questions asking for your opinion of the story you just read. Some of these questions will require a numerical answer based on your judgements while others will require you to give a textual response.
The questions you will be asked are shown below:

- How believable were the characters? Rate from 1 (hard to believe) to 7 (very believable).
- How strong was the plot of the story? Rate from 1 (little or no plot) to 7 (very strong plot)
- How easy was the story to follow? Rate from 1 (very hard to understand) to 7 (very easy to understand)

- How novel were the events in the story? Rate from 1 (the events were repetitive and clichéd) to 7 (the events were novel and exciting).
- How interesting was the story? Rate from 1 (very boring) to 7 (very exciting).
- Once you have answered these questions you will be asked to explain briefly whether you liked the story or not.

A.1.1 Examples

Consider the following story.

> A wolf resolved to disguise himself in order that he might prey upon a flock of sheep without fear of detection. So he clothed himself in a sheepskin, and slipped among the sheep when they were out at pasture. He completely deceived the shepherd, and when the flock was penned for the night he was shut in with the rest. But that very night as it happened, the shepherd, requiring a supply of mutton for the table, laid hands on the wolf in mistake for a sheep, and killed him with his knife on the spot.

Below are scores that would be given for this story.

- **How believable are the characters?** You may give this question a high number (e.g., 7 or 6) as the characters assumed roles that we expect them to. The wolf is trying to eat the sheep and resorts to deception to accomplish this.
- **How strong is the plot of the story?** Here, you may consider a relatively high number (e.g., 6 or 5) as there is a clear plot outline and progression of events. The wolf disguises himself, deceives the shepherd, and then gets killed.
- **How easy is the story to follow?** Again a high number (e.g., 6 or 7) would be given here as the story is very easy to understand and progresses in a logical order.
- **How novel are the events in the story?** The story introduces some novel actions. The wolf pretends to be a sheep and is successful at it. There is also an interesting twist in the end. The deception bring about the wolf's downfall. So, you'd give this question a high number (e.g., 6 or 7).
- **How interesting is the story?** The approach taken by the wolf to get the sheep is an interesting solution to his problems which unfortunately later ends up causing him more problems. We can also read from this story an underlying

message as the wolf ends up getting what he deserves. This seems interesting so we would give a high score (e.g., 6 or 7).

- **Please explain briefly whether you liked the story.** Here you may say that you liked this story because of the twist in the end. Or because the underlying message and plot are clear and the story interesting.

There will also be stories you may want to assign lower scores. Take the following example.

> A fir-tree was boasting to a bramble, and said, somewhat contemptuously, "You poor creature, you are of no use whatever. Now, look at me: I am useful for all sorts of things, particularly when men build houses; they can't do without me then." But the bramble replied, "Ah, that's all very well: but you wait till they come with axes and saws to cut you down, and then you'll wish you were a bramble and not a fir."

- **How believable are the characters?** Here, you may find the characters less believable in comparison to the previous story. For example, having an argument is not an activity we picture for plants. Also a fir-tree and a bramble are an unusual choice of characters as opposed the wolf and the shepherd. So, you could give this question a low score (e.g., 2 or 3).

- **How strong is the plot of the story?** This story does not have much of a plot. Two plants discuss which one is more useful. There is no event progression and no actions are undertaken by the two main characters. So, you could give 3 or 2 as an answer to the question.

- **How easy are the story to follow?** Here you could give a relatively high score (e.g., 6 or 5) as the story is easy enough to understand and the each segment of dialogue clearly follows from the last.

- **How novel are the events in the story?** A score of 1 would be appropriate as there is very little action in this story.

- **How interesting is the story?** This story is not very exciting.] Although there is an underlying message, the story itself does not deliver this message very convincingly and can hardly capture our attention. So this question would receive a low score (e.g., 2 or 1).

- **Please explain briefly whether you liked the story.** Here, you may say that you did not like the story. The characters were boring and unrealistic. The plot and the actions within the story were weak and did not help maintain interest throughout the story. However, there was a clear message and it was easy to understand.

It is possible that you may find a story performs better or worse in each of the evaluation criteria so do not feel constrained to give a story the same score for each of the questions.

A.1.2 Procedure

When you start the experiment below you will be asked to enter your personal details. Next, you will be presented with 8 short stories to evaluate in the manner described above. Once you have completed the questions for a story, click the button at the bottom of that page to advance to the next story.

A.1.3 Examples of Human Rated Fables

Figure A.1 gives two fables that have been rated by human evaluators during our elicitation study. The fable at the top of the page was rated highly in terms of interest whilst the other was given low ratings. Included are the comments written by each reviewer for that fable.

A.2 Evaluation of the Story Generation Systems

In this experiment you will be asked to read a set of short computer generated stories. Each story will be 5 sentences long and will contain only a couple of characters. After reading each story you will assess its quality along three dimensions: fluency, coherence and interest. For each dimension you will provide a rating on a scale from 1 to 5.

Fluency refers to the individual sentences of the story, whether they are grammatical and in well-formed English or just gibberish.

- If the sentences are grammatical, the you should rate the story high in terms of fluency.

Story

A trumpeter marched into battle in the van of the army and put courage into his comrades by his warlike tunes. Being captured by the enemy, he begged for his life, and said, “Do not put me to death; I have killed no one: indeed, I have no weapons, but carry with me only my trumpet here.” But his captors replied, “That is only the more reason why we should take your life; for, though you do not fight yourself, you stir up others to do so.”

Comments

“Good moral about those who encourage being as much to blame.”
“This story is likeable in the conclusion, for the captor grasped the truth so thoroughly, that the trumpeter was the encouragement for the nearby troops.”
“Yes, I thought it was thought provoking with an interesting message.”

Story

A deer said to her fawn, who was now well grown and strong, “My son, nature has given you a powerful body and a stout pair of horns, and I can’t think why you are such a coward as to run away from the hounds.” Just then they both heard the sound of a pack in full cry, but at a considerable distance. “You stay where you are,” said the deer; “never mind me”: and with that she ran off as fast as her legs could carry her.

Comments

“I didn’t like it because a mother would never leave her offspring to die.”
“I liked this story because it played on the the vulnerable nature of deer but no action really happened in this story.”
“Very dull, and a bit pointless.”

Figure A.1: Example of a fable that received high scoring judgements of interest and a fable that received low scoring judgements of interest from the participants. Also presented is a selection of comments written by the participants about each story.

- If the sentences are something like word salad, then you should give the story a low number.

Coherence refers to the overall story and how comprehensible it is.

- If the text is almost impossible to understand, then you should give it a low number.
- If the text is readily comprehensible, well-organised and doesn't require any effort on the reader's part, then you should give it a high number.

Finally, interest reflects whether you found the story exciting or boring.

- Use high numbers if you find the story original.
- On the other hand it the story is rather tedious, you should use low numbers.

A.2.1 Examples

Suppose you were given the following story:

> The prince fights the dragon.
> The dragon burns the shield.
> The prince raises the sword.
> The prince slays the dragon.
> The prince saves the princess from the tower.

Then, you may rate it high in terms of fluency (e.g., 4 or 5) as the sentences are well-formed and overall grammatical. This story would be also given a high coherence number (e.g., 4 or 5) as it progresses in a logical order, focuses on the main characters and generally makes sense. You may give this story a medium score as far as interest is concerned (e.g., 3), as it is familiar and describes a rather predicable series of events. However, if you are not familiar with the story at all, you may give it a high number. Now, take the following example:

> The prince fights the dragon.
> The prince knows of the fairy out of the horse.
> The fairy up bakes a cake.
> The prince flies to the river.
> The prince reads the dragon.

This story is much harder to read than the previous example. It contains unusual sentence structures and the individual words do not make sense together. So you would rate this story low in terms of fluency (e.g., 1 or 2). Taken in its entirety, this story also makes little sense. It is unclear why the characters are behaving as they are and there is no strong progression of actions. The story seems rather unfocused, the dragon appears in the first sentence and then only again at the end. Overall the story is not coherent and would receive a low score (e.g., 1 or 2). This story is not very interesting either as it is impossible to picture exactly what is happening. The disjoint action sequences are too confusing to capture the reader's attention. So interest here would receive a low rating (e.g., 1 or 2).

A story can receive high ratings on some dimension and low ratings on others; it is not necessary that your ratings for fluency, coherence, and interest are all either high or low.

A.2.2 Procedure

When you start the experiment below you will be asked to enter your personal details. Next, you will be presented with 12 short stories to evaluate in the manner described above. Once you have completed your rating, click the button at the bottom of that page to advance to the next story.

Appendix B

Algorithm Psudocode

Algorithm B.1 Simple story generation algorithm

```
nounList → k nouns supplied by the user
storyLength → i supplied by the user
discourseHistory → holds ordered verb list for each entity in each story
actionGraph → graph holding verb-verb relationships
choiceLimit → limit on number of selection choices
storyCache → partial/completed stories
while number of sentences < storyLength do
  newStoryCache → ∅
  for all story in storyCache do
    for all protagonist in nounlist do
      matrixVerbs → chooseMatrixVerb(protagonist, actionGraph, discourseHis-
      tory(story))
      for all verb in matrixVerbs do
        sentenceTemplates → selectTemplates(verb, choiceLimit)
        for all t in sentenceTemplates do
          sentences → fillTemplate(t, choiceLimit, discourseHistory)
          for all sent in sentences do
            story.add(sent)
          end for
        end for
      end for
    end for
  end for
  storyCache → newStoryCache
end while
select highest scoring story and realise
```

Algorithm B.2 Building the discourse history

```
storyCache → list of partial/completed stories
discourseHistory <story, <noun, verbList> > → ∅ verbList contains tuples of verb
and relationship type (subject or object)
for all story depth d do
  for all story in storyCache do
    sentences → generatedSentences(story, d)
    for all s in sentences do
      for all n in s.nouns do
        verbList → discourseHistory<story, n>
        verbList.add(n.parentVerb, n.relationship)
      end for
    end for
  end for
end for
```

Algorithm B.3 Building the mutual information database

```
contentDB → database of relationships with assigned scores
relationships → list of all relationships of form relationship(word1, word2, relation-
shipType)
limit → selected threshold for relationship occurrences
for all r in relationships do
  if count(r) > limit then
    r.MI = calculateMutualInformation(r)
    if r.MI > 0 then
      contentDB.add(r)
    end if
  end if
end for
```

Algorithm B.4 Selecting objects for verbs

```
contentDB → database of relationships with assigned scores
subject → noun subject the clause
verb → verb root of the clause
n → number of objects for the verb (1 or 2)
object1List → list of selected lexemes for first object
choiceLimit→ limit on values to return
if n = 2 then
    object2List → list of selected lexemes for second object
    if contentDB contains relationship(subject, verb, $o1FromDB, $o2FromDB)
    then
        object1List = selectFirst(choiceLimit, $o1FromDB)
        object2List = selectFirst(choiceLimit, $o2FromDB)
    else if contentDB contains relationship(subject, verb, $object1) then
        object1List = selectFirst(choiceLimit, $o1FromDB)
        objectFromDB = contentDB.selectMIObjects(verb)
        object2List = selectFirst(choiceLimit, objectFromDB)
    else
        objectFromDB = contentDB.selectMIObjects(verb)
        object1List = selectFirst(choiceLimit, objectFromDB)
        object2List = selectFirst(choiceLimit, objectFromDB)
    end if
else
    if contentDB contains relationship(subject, verb, $object1) then
        object1List = selectFirst(choiceLimit, $o1FromDB)
    else
        objectFromDB = contentDB.selectMIObjects(verb)
        object1List = selectFirst(choiceLimit, objectFromDB)
    end if
end if
```

Algorithm B.5 Combined coherence and interest ranking

```
stories → list of generated stories
for all s in stories do
    s.interestScore = scoreInterest(s)
    s.coherenceScore = scoreCoherence(s)
end for
for all s in stories do
    s.rankInterest = denseRank(interestScore, stories)
    s.rankCohherence = denseRank(coherenceScore, stories)
end for
for all s in stories do
    s.rank = denseRank(rankInterest + rankCoherence, stories)
end for
```

Algorithm B.6 Generating story plots

```
storyPlot → list of x clause nodes n
stories → list of stories ∅
wordLimit → limit of selection choices
stories.add(sentence(n_0))
for all nodes n_j where n>0 do
    sentences → ∅
    for all sentence template sTemp in in n_j do
        for all lexeme lex in sTemp do
            if lexeme.word = ε then
                if lexeme.type = noun then
                    selectSimilarNouns(lex.sense, limit)
                else
                    selectTopMI(sTemp, limit)
                end if
            end if
        end for
        sentences.append(sentence)
    end for
    newStories = ε
    for all story in stories do
        for all sentence in sentences do
            newStories.add( story.add(sTemp.sentences) )
        end for
    end for
    stories → newStories
end for
```

Algorithm B.7 Fitness function ranking

stories → list of generated stories
fitnessFunctions → set of x scoring functions
for all s in stories **do**
  for all f in fitnessFunctions **do**
    s.score(f) = *scoreFitness*(s, f)
  end for
end for
for all s in stories **do**
  for all f in fitnessFunctions **do**
    s.rank(f) = *denseRank*(f, stories)
  end for
end for
for all s in stories **do**
  s.rank = *denseRank*(sum(s.rank(f_0), ... ,s.rank(f_x)), stories)
end for

Algorithm B.8 Dense ranking (*denseRank*)

scoringFunction → value of each item to use for comparison
itemList → list of x items i with assigned scores
order itemList by scoringFunction sorting order descending
i_0.rank = x
for all i_j in itemList **do**
  if i_j.score < i_{j-1}.score **then**
    i_j.rank = i_{j-1}.rank - 1
  else
    i_j.rank = i_{j-1}.rank
  end if
end for

Bibliography

E. Althaus, N. Karamanis, and A. Koller. Computing locally coherent discourses. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 247–254, Barcelona, Spain, July 2004.

S. Andersen and B. M. Slator. Requiem for a theory: the 'story grammar' story. *Journal of Experimental & Theoretical Artificial Intelligence*, 2(3):253–275, July 1990.

R. Anderson, R. Reynolds, D. Schallert, and E. Goetz. Frameworks for comprehending discourse. *American Educational Research Journal*, 14(4):367–381, 1977.

N. Asher and A. Lascarides. *Logics of Conversation*. Cambridge University Press, Cabridge, UK, 2003.

B.-C. Bae and R. M. Young. A use of flashback and foreshadowing for surprise arousl in narrative using a plan-based approach,. In *Proceedings of the International Conference on Interactive Digital Storytelling (ICIDS 08)*, pages 156–167, Erfurt, Germany, November 2008.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, volume 1, pages 86–90, Montreal, Quebec, August 1998.

C. Bartneck. Integrating the occ model of emotions in embodied characters. In *Proceedings of the Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges (in conjunction with HF2002 and OZCHI2002)*, Melbourne, Australia, November 2002.

R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17, Madrid, Spain, July 1997.

R. Barzilay and M. Lapata. Collective content selection for concept-to-text generation. In *Proceedings of the Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 331–338, Vancouver, British Columbia, October 2005.

R. Barzilay and M. Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, March 2008.

S. Bergsma, E. Pitler, and D. Lin. Creating robust supervised classifiers via web-scale n-gram data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 865–874, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

H. Bird, S. Franklin, and D. Howard. Age of acquisition and imageability ratings for a large set of words, including verbs and function words. *Behavior Research Method, Instruments, & Computers*, 33(1):73–79, 2001.

J. B. Black and R. Wilensky. An evaluation of story grammars. *Cognitive Science: A Multidisciplinary Journal*, 3(3):213–229, July 1979.

J. Bos, S. Clark, M. Steedman, J. Curran, and J. Hockenmaier. Wide-coverage semantic representations from a ccg parser. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 1240–1246, Geneva, Switzerland, August 2004.

P. Bramsen, P. Deshpande, Y. K. Lee, and R. Barzilay. Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 189–198, Sydney, Australia, July 2006.

E. Briscoe and J. Carroll. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 1499–1504, Las Palmas, Gran Canaria, May 2002.

S. Brody and M. Lapata. Good neighbors make good senses: Exploiting distributional similarity for unsupervised WSD. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 65–72, Manchester, UK, August 2008.

S. Brody and M. Lapata. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the Association of Computational Linguistics (EACL 2009)*, pages 103–111, Athens, Greece, April 2009.

A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistic (NAACL) Workshop on WordNet and Other Lexical Resources*, pages 29–34, Pittsburgh, Pennsylvania, June 2001.

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

C. B. Callaway and J. C. Lester. Pronominalization in generated discourse and dialogue. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 88–95, Philadelphia, Pennsylvania, July 2002b.

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

N. Chambers and D. Jurafsky. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Singapore, August 2009.

E. Charniak and M. Elsner. Em works for pronoun anaphora resolution. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)*, pages 148–156, Athens, Greece, March-April 2009.

J. Chen, S. Bangalore, O. Rambow, and M. A. Walker. Towards automatic generation of natural language generation systems. In *Proceedings of the 19th International conference on Computational Linguistics*, Taipei, Taiwan, August 2002.

L. Chen, K. Bechkoum, and G. Clapworthy. Equipping a lifelike animated agent with a mind. In *Proceedings of the 3rd International Workshop on Intelligent Virtual Agents*, pages 72–85, Madrid, Spain, September 2001.

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

J. Clarke and M. Lapata. Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1–11, Prague, Czech Republic, June 2007.

T. R. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, second edition, 2001.

A. Correira. Computing story trees. *American Journal of Computational Linguistics*, 6(3-4):135–149, July-December 1980.

R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233—263, April-June 1995.

P. A. Duboue and K. R. McKeown. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the 2nd International Conference on Natural Language Generation (INLG-2002)*, pages 89–96, Ramapo Mountains, New York, July 2002.

T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. Planning under incomplete knowledge. In *Proceedings of the 1st International Conference on Computational Logic*, pages 807–821, London, UK, July 2000.

C. Fairclough and P. Cunningham. A multiplayer case based story engine. In *Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*, pages 41–46, London, UK, November 2003.

S. Fass. Virtual storyteller: An approach to computational storytelling. Master's thesis, Dept. of Computer Science, University of Twente, 2002.

M. Fayzullin, V. S. Subrahmanian, M. Albanese, C. Cesarano, and A. Picariello. Story creation from heterogeneous data sources. *Multimedia Tools and Applications*, 33 (3):351–377, June 2007.

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

E. Figa and P. Tarau. Story traces and projections: Exploring the patterns of storytelling. In *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2003)*, Darmstadt, Germany, March 2003.

C. J. Fillmore, C. R. Johnson, and M. R. L. Petruck. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250, September 2003.

P. Foltz, W. Kintsch, and T. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–307, 1998.

A. Gatt and E. Reiter. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 90–93, Athens, Greece, March 2009.

A. Gatt, I. van der Sluis, and K. vanDeemter. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the 11th European Workshop on Natural Language Generation*, pages 49–56, Saarbrücken, Germany, June 2007.

N. Ge, J. Hale, and E. Charniak. A statistical approach to anaphora resolution. In *Proceedings of the 6th Workshop on Very Large Corpora*, pages 161–170, Montreal, Quebec, August 1998.

P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás. Story plot generation based on CBR. *Knowledge-Based Systems*, 18(4-5):235–242, August 2004.

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

P. Grice. Logic and conversation. *Syntax and semantics: Speech acts*, 3:41–58, June 1975.

R. Grishman, C. Macleod, and A. Meyers. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 268–272, Kyoto, Japan, August 1994.

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

M. A. K. Halliday and R. Hasan. *Cohesion in English*. Longman, London, UK, 1976.

H. Halpin and J. D. Moore. Event extraction in a plot advice agent. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 857–864, Sydney, July 2006.

H. Halpin, J. D. Moore, and J. Robertson. Towards automated story analysis using participatory design. In *Proceedings of the 1st ACM Workshop on Story Representation, Mechanism and Context, SRMC 2004*, pages 75–83, New York, New York, October 2004.

B. Harrington and S. Clark. Asknet: Creating and evaluating large scale integrated semantic networks. *International Journal of Semantic Computing (IJSC)*, 2(3):309–310, September 2008.

R. Hervas and M. Finlayson. The prevalence of descriptive referring expressions in news and narrative. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 49–54, Uppsala, Sweden, July 2010.

G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *WordNet: An electronic lexical database*. MIT Press, 1998.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 133–142, Edmonton, Alberta, July 2002.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2 edition, 2009.

N. Karamanis and H. M. Manurung. Stochastic text structuring using the principle of continuity. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'02)*, pages 81–88, New York, New York, July 2002.

N. Karamanis, C. Mellish, M. Poesio, and J. Oberlander. Evaluating centering for information ordering using corpora. *Computational Linguistics*, 35:29–46, March 2009.

F. Keller, S. Gunasekharan, N. Mayo, and M. Corley. Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods*, 41(1):1–12, February 2009.

R. Kibble and R. Power. Optimizing referential coherence in text generation. *Computational Linguistics*, 30(4):401–416, December 2004.

S. Klein, J. F. Aeschlimann, D. F. Balsiger, S. L. Converse, C. Court, M. Foster, R. Lao, J. D. Oakely, and J. D. Smith. Automatic novel writing. In *Text Processing/Textverarbeitung*, pages 338–412, Berlin and New York, 1979.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 181–184, Detroit, Michigan, May 1995.

K. Knight and V. Hatzivassiloglou. Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, pages 252–260, Cambridge, Massachusetts, June 1995.

A. Korhonen, Y. Krymolowski, and E. J. Briscoe. A large subcategorization lexicon for natural language processing applications. In *Proceedings of the the 5th international conference on Language Resources and Evaluation (LREC 06)*, Genova, Italy, May 2006.

E. Krahmer, S. van Erk, and A. Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, March 2003.

H. Kucera and N. Francis. *Computational Analysis of Present-day American English*. Brown University Press, Providence, Rhode Island, 1967.

G. P. Lakoff. Structural complexity in fairy tales. *The Study of Man*, 2:128–190, 1972.

I. Langkilde and K. Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 704–710, Montreal, Quebec, August 1998.

M. Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 545–552, Sapporo, Japan, July 2003.

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

M. Lebowitz. Creating a story-telling universe. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 63–65, Karlsruhe, West Germany, August 1983.

M. Lebowitz. Story-telling as planning and learning. *Poetics*, 14:483–502, 1985.

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

H. Liu and G. Davenport. ConceptNet: A practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226, 2004.

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

B. Lönneker. Narratological knowledge for natural language generation. In *Proceedings of the 10th European Workshop on Natural Language Generation*, pages 91–100, Aberdeen, Scotland, August 2005.

B. Lonneker, J. Meister, P. Gervas, F. Peinado, and F. Mateas. Story generators: Models and approaches for the generation of literary artefacts. In *the 17th Joint International Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing*, pages 126–133, Victoria, Canada, June 2005.

A. B. Loyall and J. Bates. Personality-rich believable agents that use language. In *Proceedings of the 5th international conference on Autonomous agents (AGENTS '97)*, pages 106–113, Marina del Rey, California, February 1997.

J. M. Machado. *Early childhood experiences in language arts: emerging literacy*. Thomson/Delmar Learning, Clifton Park, New York, 7 edition, 2003.

F. Mairesse and M. Walker. Trainable generation of big-five personality styles through data-driven parameter estimation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 165–173, Columbus, Ohio, June 2008.

W. C. Mann and S. A. Thompson. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, Information Sciences Institute, June 1987 1987.

H. Manurung. *An evolutionary algorithm approach to poetry generation*. PhD thesis, University of Edinburgh, 2003.

R. A. Mar. The neuropsychology of narrative: story comprehension, story production and their interralation. *Neurophychologia*, 42(10):1414–1434, 2004.

D. Marcu. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational Linguistics*, 26(3):395–448, September 2000.

K. F. McCoy and M. Strube. Generating anaphoric expressions: Pronoun or definite description? In *Proceedings of the ACL workshop on Discourse and Reference Structure*, pages 72–81, Baltimore, Maryland, June 1999.

N. McIntyre and M. Lapata. 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*, pages 217–225, Singapore, August 2009.

N. McIntyre and M. Lapata. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572, Uppsala, Sweden, July 2010.

K. R. McKeown, S. Pan, J. Shaw, D. A. Jordan, and B. A. Allen. Language generation for multimedia healthcare briefings. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 277–282, Washington, D.C., March 1997.

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

I. Mel'čuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, 1988.

C. Mellish, A. Knott, J. Oberlander, and M. O'Donnell. Experiments using stochastic search for text planning. In *Proceedings of the 9th International Conference on Natural Language Generation*, pages 98–107, Niagara-on-the-Lake, Ontario, August 1998.

B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995.

E. Miltsakaki and K. Kukich. The role of centering theory's rough-shift in the teaching and evaluation of writing skills. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 408–415, Hong Kong, October 2000.

J. Mitchell and M. Lapata. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 430–439, Singapore, August 2009.

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

J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48, March 1991.

J. H. Murray. *Hamlet on the holodeck: the future of narrative in cyberspace*. Free Press, New York, New York, 1997.

M. Newman. The structure and function of complex networks. *Society for Industrial and Applied Mathematics Review (SIAM Rev)*, 45(2):167–256, 2003.

K. Oinonen, M. Theune, A. Nijholt, and J. Uijlings. Designing a story database for use in automatic story generation. In *Proceedings of the 5th International Conference on Entertainment Computing – ICEC 2006*, pages 298–301, Cambridge, UK, September 2006.

A. Ortony, G. L. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge, UK, 1988.

F. Peinado and P. Gervás. Evaluation of automatic generation of basic stories. *New Generation Computing, Computational Paradigms and Computational Intelligence*, 24(3):289–302, May 2006.

F. Peinado, P. Gervás, and B. Díaz-Agudo. A description logic ontology for fairy tale generation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation: Workshop on Language Resources for Linguistic Creativity*, pages 56–61, Lisbon, Portugal, May 2004.

L. Pemberton. A modular approach to story generation. In *Proceedings of the 4th conference on European chapter of the Association for Computational Linguistics*, pages 217–224, Manchester, UK, April 1989.

R. Pérez y Pérez and M. Sharples. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 13(2):119–139, April 2001.

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

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

P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, July–December 1999.

M. O. Riedl. Vignette-based story planning: Creativity through exploration and retrieval. In *Proceedings of the 5th International Joint Workshop on Computational Creativity*, pages 41–50, Madrid, Spain, September 2008.

M. O. Riedl and C. Leon. Toward vignette-based story generation for drama management systems. In *Proceedings of the 2nd International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN), Workshop on Integrating Technologies for Interactive Story*, pages 23–28, Cancun, Mexico, January 2008.

M. O. Riedl and R. M. Young. An intent-driven planner for multi-agent story generation. In *Proceedings of the 3rd International Joint Conference on Autonomous*

Agents and Multiagent Systems (AAMAS '04), pages 186–193, New York, New York, July 2004.

M. O. Riedl and R. M. Young. Story planning as exploratory creativity: Techniques for expanding the narrative search space. *New Generation Computing, Computational Paradigms and Computational Intelligence*, 24(3):303–323, May 2006a.

M. O. Riedl and R. M. Young. From linear story generation to branching story graphs. *IEEE Computer Graphics and Applications*, 26(3):23–31, 2006b.

J. Robertson and J. Good. Using a collaborative virtual role-play environment to foster characterisation in stories. *Journal of Interactive Learning Research*, 14(1):5–29, 2003a.

J. Robertson and J. Good. Ghostwriter: A narrative virtual environment for children. In *Proceedings of the 2003 Conference on Interaction Design and Children (IDC2003)*, pages 85–91, Preston, England, July 2003b.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

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

A. Siddharthan and K. McKeown. Improving multilingual summarization: Using redundancy in the input to correct mt errors. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 33–40, Vancouver, British Columbia, October 2005.

P. Singh. The public acquesition of commonsense knowledge. In *Proceedings of AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*, Palo Alto, California, March 2002.

P. Singh and B. Barry. Collecting commonsense experiences. In *Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP '03)*, pages 154–161, Sanibel Island, Florida, October 2003.

P. Singh, B. Barry, and H. Liu. Teaching machines about everyday life. *BT Technology Journal*, 22(4):227–240, 2004.

C. Solis, J. T. Siy, E. Tabirao, and E. Ong. Planning author and character goals for story generation. In *Proceedings of the NAACL HLT 2009 Workshop on Computational Approaches to Linguistic Creativity (CALC '09)*, pages 63–70, Boulder, Colorado, June 2009.

R. Soricut and D. Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 149–156, Edmonton, Alberta, May-June 2003.

A. Stent, R. Prasad, and M. Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 79–86, Barcelona, Spain, July 2004.

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

B. Tearse, M. Mateas, and N. Wardrip-Fruin. Minstrel remixed: a rational reconstruction. In *Proceedings of the 3rd Intelligent Narrative Technologies Workshop (INT3 '10)*, pages 1–7, Monterey, California, June 2010.

M. Theune, S. Faas, D. Heylen, and A. Nijholt. The virtual storyteller: Story creation by intelligent agents. In *Proceedings of the 1st International Conference on Technologies for the 1st Interactive Digital Storytelling and Entertainment (TIDSE 2003)*, pages 204–215, Darmstadt, Germany, 2003.

M. Theune, S. Rensen, H. op den Akker, D. Heylen, and A. Nijholt. Emotional characters for automatic plot creation. In *Proceeding of the 2nd International Conference on Interactive Digital Storytelling and Entertainment (TIDSE 2004)*, pages 95–100, Darmstadt, Germany, June 2004.

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

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

S. R. Turner. *The creative process: A computer model of storytelling and creativity*. Erlbaum, Hillsdale, New Jersey, 1994.

J. Viethen, R. Dale, E. Krahmer, M. Theune, and P. Touset. Controlling redundancy in referring expression. In *Proceedings of the 6th International Language Resources and Evaluation (LREC'08)*, volume 950-957, Marrakech, Morocco, May 2008.

M. Walker, A. Joshi, and E. Prince. Centering in naturally occurring discourse: An overview. In *Centering Theory in Discourse*, pages 1–28. Oxford University Press, Oxford, UK, 1997.

M. A. Walker, O. Rambow, and M. Rogati. Spot: a trainable sentence planner. In *the 2nd meeting of the North American Chapter of the Association for Computational Linguistics*, pages 17–24, Pittsburgh, Pennsylvania, June 2001.

M. A. Walker, O. Rambow, and M. Rogati. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3-4):409–433, July–October 2002.

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

R. M. Young and M. Riedl. Towards an architecture for intelligent control of narrative in interactive virtual worlds. In *Proceedings of the 8th international conference on Intelligent user interfaces (IUI '03)*, pages 310–312, Miami, Florida, January 2003.