Noname manuscript No.
(will be inserted by the editor)

The Role of Evaluation-Driven Rejection in the Successful Exploration of a Conceptual Space of Stories

Carlos León · Pablo Gervás

the date of receipt and acceptance should be inserted later

Abstract Evaluation processes are a basic component of creativity. They guide not only the pure judgement about a new artefact but also the generation itself, as creators constantly evaluate their own work. This paper proposes a model for automatic story generation based on the evaluation of stories. A model of how quality in stories is evaluated is presented, and two possible implementations of the generation guided by this evaluation are shown: exhaustive space exploration and constrained exploration. A theoretical model and its implementation are explained and validation of the evaluation function through comparison with human criteria are described.

Keywords Story generation · Evaluation of Stories · Reader Model · Conceptual Space Exploration

1 Introduction

Evaluation of created artifacts is one of the basic components of creativity. The evaluation itself is perhaps the origin of creativity, as human opinions establishes what *quality* is in a broad range of creative disciplines, specially artistic ones. As such, there has been general consideration of it as a fundamental part of most models of creativity, from different points of view [2, 26, 21].

Implementation of systems which focus on creativity has always been aware of the importance of evaluation [15, 16, 13]. However, none of these systems really perform evaluation in such a way that the system itself explicitly decides if the object is good or not. This happens according to their own definition of "good"

C. León
Departamento de Ingeniería del Software e Inteligencia Artficial
Universidad Complutense de Madrid
E-mail: cleon@fdi.ucm.es

P. Gervás
Instituto de Tecnología del Conocimiento
Universidad Complutense de Madrid
E-mail: pgervas@sip.ucm.es

story, which does not take creativity or high-value into account in all cases. In general, evaluation is usually addressed either as a human corroboration of the quality of the system or implicitly during the generation itself.

It can be claimed that evaluation must not necessarily happen post hoc [9, 20], and implicit evaluation is carried out in many generation systems. However, *explicitly* addressing evaluation itself could help to create systems in which the measurement of creativity is explicitly controlled. If a computational evaluation function is available in some domain, classic Artificial Intelligence approaches can be used to address the generation of artefacts evaluated by such a function.

The creation of such an evaluation function requires a known domain for developing a manageable and implementable model. This paper proposes a computational model for story evaluation in which an evaluation function receives stories and outputs a real value as the rating for that story. The domain of story generation has been chosen because there is a significant volume of existing literature on automatic generation of stories [5,13,16,15,17].

In generation, storytelling systems have focused on the pure generation of stories, with little emphasis on the evaluation of the stories by the system itself ([13, 10,4,19], for instance). They all rely on the idea that every story that the system can generate is by construction a good story according to human criteria or, at least, a *correct* story.

The introduction of self-assessment of the stories used as a control mechanism over a simple construction algorithm intended to over-generate opens up a new approach to story generation which presents three important advantages: it allows a more formal characterisation of the creative process, it generates a much larger set of candidate stories, and, in doing so, it explores possibilities that would never have been reached by knowledge-based construction algorithms.

Section 2 reviews formal accounts of creativity that have guided the development. Section 3 defines the conceptual space on which the experiment is based, and describes the evaluation function developed and its implementation. Section 4 presents the empirical validation of the evaluation function by human judges. Section 5 describes a simple exploration algorithm that relies on the evaluation function to discriminate between high and low valued stories, according to humans. Section 6 refines the search algorithm by applying a pruning function evolved as a partial approximation to the evaluation function. Section 7 discusses the presented solution, and section 8 outlines general conclusions of the experiment.

2 Review of Formal Aspects of Computational Creativity and Evaluation

Boden's account of creativity [2,3] has been widely studied and it is usually considered to set the base for current research on Computational Creativity. The philosophical definition of *conceptual space* by Boden led researchers in the field to the formal identification of the properties of creative systems.

From a more psychological perspective, Johnson-Laird defines computational creativity to be possible based on three different approaches: neo-Darwinian (combining and choosing elements), neo-Lamarckian (constrained selection of viable elements), and mixed, combining the two previous algorithms [8,9]. Johnson-Laird

makes an example of these types of creativity making reference to jazz improvisation in bass lines (improvisation) and chord sequences (composed by a more slow and developed process). The relation of the proposed model with this conception of creativity is discussed in Section 7.

2.1 Formalization of Creative Systems

Boden's ideas have been the cause of discussion in literature, and several different interpretations of her work have been published. Only those relevant to the purpose of this paper are summarised here.

Wiggins [26, 25] extends Boden's model and formalizes it by adding explicit and exact meaning of the parts that involve creativity. Wiggins formalizes the idea of conceptual space in such a way that it can be defined through the relations between several sets containing rules and artifacts.

Conceptual spaces ($\{\mathscr{C}_0 \dots \mathscr{C}_n\}$) must be strict subsets of a universe $\mathscr{U}$. Wiggins defines three rule sets operating on conceptual spaces: $\mathscr{R}$, rules defining the conceptual space; $\mathscr{T}$, rules for traversing the conceptual space; and $\mathscr{E}$, a function that evaluates the objects in the conceptual space.

It is important to note the differences between these three elements. The rules in $\mathscr{R}$ constitute a declarative definition of the set of all possible elements that might be considered a solution. As an example we could consider the definition of all integer numbers. In contrast, the rules in $\mathscr{T}$ are intended as means to arriving, in a particular moment in time, at a particular point in the conceptual space. One can imagine them more like operations on integer numbers, which lead directly to a particular result. In terms of creativity, they encode the processes by which each particular creator moves along a conceptual space. As arithmetic operations would, they are not intended to exhaustively enumerate all possible numbers, but rather to lead to specific numbers that are solutions to particular problems. The interesting insight that Wiggins' formalism makes very clear is that it is possible to define a set of rules $\mathscr{T}$, ostensibly over a conceptual space defined by a set of rules $\mathscr{R}$, that leads to results that are not in the space defined by $\mathscr{R}$.

This insight by Wiggins illustrates the difference between Computational Creativity and Good Old Fashioned Artificial Intelligence (GOFAI). Whereas GOFAI operates over statically defined problems defined by a set of rules $\mathscr{R}$ and an evaluation function $\mathscr{E}$ (but no $\mathscr{T}$), Computational Creativity becomes dynamically defined in terms of a loop where $\mathscr{T}$ (a set of tentative jumps over a conceptual space of only barely hinted at extension, and only enumerable in an abstract, unachievable way) progressively forces the definition of that very conceptual space (as given by $\mathscr{R}$) to expand. The force that drives this loop is an evaluation function $\mathscr{E}$ that is capable of scoring the results of $\mathscr{T}$ even when they fall outside the set defined by $\mathscr{R}$.

Another powerful insight that can be clarified by Wiggins' formalism involves the fact that the evaluation function can itself become affected by the iterations of the loop, progressively evolving to rate higher elements produced by $\mathscr{T}$ that were not originally in the set defined by $\mathscr{R}$. Although computational systems modifying its rules already exist [11], little is known about this evolution. Jennings [7] presented a very plausible explanation for the emergence of $\mathscr{E}$ as a result of interaction

among a society of producers and consumers of creative artifacts, each operating with his own evolving versions of $\mathscr{T}$ and $\mathscr{R}$.

Revising Boden's work with a particular slant on the task of written composition, Sharples [21,22] addresses the definition of the conceptual space as the set of elements (written text, in his case) that can be obtained inside a certain universe (which would match Wiggins' $\mathscr{U}$ set) by the application of several restrictions that external and internal circumstances put on the writer: her knowledge, the length of the writing, and so on. These restrictions or *schemas* lead the writer to artifacts inside the conceptual space. Sharples also studies the psychological aspects of writing as a task, identifying two stages: an unrestricted generation process (*engagement*) and a revision and correction of the generated material (*reflection*). Related with Wiggins work, the engagement stage could match the traversal in the $\mathscr{T}$ set, and the reflection would be heavily influenced by the writer's own $\mathscr{E}$ function. Riedl also studies a possible formalization of Wiggins framework regarding story generation [18].

2.2 Evaluation vs. Generation in Story Telling Systems

Storytelling systems follow a long tradition of generation systems, in which evaluation has traditionally been addressed using human judgement. TaleSpin creates stories by running a simulation of characters trying to achieve its objectives [13]. In TaleSpin, there is an explicit evaluation function for the evaluation of stories: the story is appropriate if the main character succeeds in its objective. In Fabulist [17], any generated story is also accepted with no additional evaluation. Minstrel [23] generates stories by using schemas of previous stories that can be adapted so that the new story can use them. Minstrel does not really focus on the evaluation of the stories it generates, but on the system itself, comparing it with models of cognition, psychology and computation, although it defines a process to evaluate the results using human feedback. Universe [10] implicitly accepts that a correct story (in the domain of TV serials) creates and develops conflicts between the characters, so, again, no explicit evaluation is performed. Brutus performs story generation using a knowledge intensive rule-set that adds information about what is a character, a story or a "good" narration. Brutus [4] hard-codes *quality* into its internal logic rules, and, following a concept shared with many storytellers, assumes that the final objective of a storytelling system is to create a story that is evaluated with a high value for humans. In this way, every story generated by Brutus is always "correct". Mexica [16] creates stories by incrementally adding events that increase the emotional links in the story. It follows Sharples' model of engagement and reflection as fundamental process for switching from pure generation (engagement) to a form of evaluation of the partial story in which the system corrects some parts of the story (reflection). Mexica shows an evaluation of the stories it produces based on direct human opinions about the plots. Mexica uses two main approaches for measuring creativity: *novelty* (comparing the generated story with previous ones) and *emotional tension* (measuring the evolution of basic emotions in the story). These evaluations are used to conclude that Mexica's result is good enough inside its domain. Peinado and Gervás [15] perform specific evaluation by querying four values from human testers. Among other things, they conclude that human knowledge, while understanding stories, tends to complete

and give sense to partially non-coherent stories. This happens to the extent that story with a high degree of random facts with no causal relation with the rest of the story receive good ratings.

3 Definition of the Conceptual Space for Story Generation

The universe of all possible stories $\mathscr{U}_s$ should cover all possible artifacts that can be considered a story by a human reader. We want to restrict the exploration in our model to a particular subset of stories, so it is necessary to define a conceptual space $\mathscr{C}_s$ inside the universe $\mathscr{U}_s$. This conceptual space is restricted by a rule-set $\mathscr{R}_s$, in such a way that it only contains the computationally valid stories in which we are interested. In the proposed model, rules for constraining the set are:

- *Events are basic structures based on a verb or action, with a variable list of parameters.* Informally, any basic sentence containing a verb and some other information is an event. "John went out" and "Michael was reading a book" are events.
- *Messages are sets of events.* Any set of events (order is not important) is a message. Messages contain information units conveyed to the audience as a single element. For instance, sentences in written text or camera takes in movies would be messages. "John went out while Michael was reading a book" is a valid message in this model.
- *Stories are lists of messages.* Any ordered list of messages can be considered a story, no matter its quality. For instance, a story could be: "John went to the cinema. He watched the movie. He went back home".

Any story so defined is a valid story in the conceptual space $\mathscr{C}_s$, considering this as the set of stories that interest us. It is possible to create a new story in the set $\mathscr{C}_s$ by adding a new event to the story, so it is infinite and therefore it is not possible to fully explore it in finite time. While this does not preclude a computational approach, having a finite set allows exact study of the proportion of "good" stories over "bad" ones, according to an evaluation function, achieved by any given procedure, which will serve to identify the quality of the proposed solution, as explained later. To create a new finite subset of $\mathscr{C}_s$, A new $\mathscr{R}_s^{'}$ is defined by Equation 1:

$$\mathscr{R}_s^{'} = \mathscr{R}_s \cup \mathscr{R}_d \tag{1}$$

where $\mathscr{R}_s^{'}$ is a new rule set constraining a new computationally processable conceptual space of stories inside a domain d, $\mathscr{C}_d$, and $\mathscr{R}_d$ is the set of restrictions regarding computational requirements, time and space limits and domain information. Parameters for $\mathscr{R}_d$ are:

- μ: the maximum number of events in a message.
- φ: the maximum number of messages in a story.
- Δ: the set of terminals representing characters.
- Π: the set of terminals representing places.
- ω: the set of terminals representing objects.
- Φ: the set of terminals representing actions.

Argument name	Accepted types	Meaning
subject	terminal in the domain	subject of the event
direct	terminal or another event	direct object of the event
indirect	terminal	indirect object of the event
place	terminal	location
start	integer	event's start time
end	integer	event's end time
before	event	following event
after	event	preceding event

Table 1: Set of arguments available for actions, used to construct events.

where μ, φ, Δ, Π, ω and Φ are domain dependent. In the current prototype, the values for these sets are given by Equation 2:

$$\begin{aligned}
&\mu = 1 \\
&\varphi = 10 \\
&\Delta = \{john, man, godfather\} \\
&\Pi = \{bus_stop, warehouse\} \\
&\omega = \{gun, money\} \\
&\Phi = \left\{ \begin{array}{c} go, in, take, give, say, desire, suppose, realize, late, current_time, \\ angry, afraid, surprised, kill, find, dead, die, \\ pay, goodbye, boss, sad, happy, escape, friend, hate, have \end{array} \right\}
\end{aligned} \tag{2}$$

The set of all possible events can be defined in terms of a grammar. Such a grammar encodes how the different types of terminal can be put together to form valid events.

The grammar of events is defined in terms of domain dependent template based rules.[1] For instance, events for the action *go* could be generated using one of these patterns:

$$\begin{array}{c} go(character?, place?) \\ go(character?, place?, start\_time?) \\ go(character?, place?, start\_time?, end\_time?) \\ \cdots \\ go(character?, place?, relative\_start\_time?, relative\_end\_time?) \end{array} \tag{3}$$

The set of arguments available for the actions in ϕ is listed in Table 1.

3.1 Expanding and Contracting the Conceptual Space

According to the brief description given in Section 2 of how Computational Creativity can be characterised according to Wiggins' formal model, a creative system should not be considered in terms of its generation rules. Instead, one should concentrate on studying the dynamics of the various rule sets that govern it. In formal terms, this can be represented as the way sets $\mathscr{R}$, $\mathscr{T}$ and $\mathscr{E}$ change over

[1] These templates define relationships between particular actions and the set of arguments that are valid for them, and thereby encode the semantic constraints that people associate with particular terms.

time, and how they affect one another in the process. If this insight is applied to the storytelling domain, two very different approaches to the task of searching for a successful story generation program arise.

On one hand, classic storytelling systems focus quite strictly on developing successful models of the traversal function $\mathscr{T}$ (a rule set or program that will generate good candidates for best-story-right-now). No attempt is usually made to define the conceptual space in which the search takes place. However, the set of knowledge elements on which each program is based, together with the construction process employed, constitutes an implicit definition of $\mathscr{R}$. This is not absolutely required for performing story generation in general, specially outside Computational Creativity. There are very few efforts at defining explicitly an evaluation function along the lines of $\mathscr{E}$. It is also rare for reports on this kind of work to include descriptions of the refinement process that has lead to the development of particular solutions. However, one can surmise from the reports that an initial implementation of a $\mathscr{T}$ is built, which results in a solution space that is found poor (according to an implicit evaluation function provided by the researcher's own intuition). The implementation of $\mathscr{T}$ is progressively extended until a certain threshold of acceptability is reached.

On the other hand, the approach presented in this paper attempts to provide explicit formal representations of all the three sets of rules described in Wiggins' formalism (sets $\mathscr{R}$, $\mathscr{T}$ and $\mathscr{E}$), albeit for a simplified toy problem. We consider a very broad definition of the set of rules $\mathscr{T}$ for traversal of the conceptual space, one that basically covers the whole universe as defined by $\mathscr{R}$. This definition of $\mathscr{T}$ is to be held constant, and the identification of an interesting solution space is explored exclusively by progressively modifying the rules $\mathscr{E}$ that define the evaluation function which is used to filter the set of candidates identified by $\mathscr{T}$, excluding low scorers. The progressive refinement of the solution space from a given instantion of $\mathscr{E}$ allows for two basic operations: *expanding the solution space* and *contracting the solution space*. When expanding the solution space, a restrictive $\mathscr{E}$ function is made less restrictive, thus making a larger set of stories accessible as solution to the system. When contracting the solution space, a very permissive $\mathscr{E}$ function (one that allows the system to generate many "bad" stories) is made more restrictive.

3.2 Evaluation Function

The conceptual space with domain restrictions, $\mathscr{C}_d$, contains both high-rated, low-rated and meaningless stories according to human criteria. A function capable of selecting the high-valued stories from among the rest, at least to some extent, is required. Equation 4 describes the *story evaluation function*, E, an instantiation of the $\mathscr{E}$ evaluation function ranging over the domain of stories in the $\mathscr{C}_d$ set and returning a real value in the range $[-1, +1]$, -1 representing extremely poor quality and $+1$ representing a very good story.

$$E : \mathscr{C}_d \longrightarrow \mathbb{R}_{[-1,+1]} \tag{4}$$

The function iterates over the sequence of messages in a story in order, processing their constituent events. The value for this function is computed from values assigned to a set of significant variables. These values are of three types, corresponding to three different aspects of a story that need to be taken into account:

accumulation of contributions, *appearance of patterns* and *inference*, as explained next.

A number of variables that take values based on accumulation of contributions from individual events depending on the meaning of the event:

- *Interest* models the intention of the reader to continue reading the story.
- *Danger* represents how much danger the reader perceives in the story. When a character is about to die, the *danger* variable is raised.
- *Love* measures the amount of love in the story that the reader perceives. All kinds of love are covered by this variable: romantic love, friendship, and so on. For instance, when a character kisses another character the value of the *love* variable is increased.
- *Tension* captures the sense that an important event is to come in the story.
- *Humanity* is raised when human behaviour is clearly present in the story and characters' reactions are human-alike.
- *Action* represents the amount of change and movement that the story contains. Events involving some kind of action (moving, talking...) raise the *action* variable, whereas descriptive events (position, feelings) do not.
- *Empathy* models the development of empathy (positive or negative) towards characters. For instance, if some character kills another character, the empathy towards the murderer is lowered.
- *Emotion* represents the perception of emotive events: a heroic fact, fear and other events related with human emotions.

Some variables measure the appearance of particular patterns or relationships between the events of a story:

- *Causality* measures the number of causality links. If the cause for an event is found, the *causality* is raised.
- *Funny* measures how funny the story is, based on the occurrence of specific templates.
- *Chronology* measures, according to some basic rules, the correctness of the time order of facts in the story.

To model the way people react to stories, the evaluation function must model the ability people have for "interpreting" stories by adding hypotheses, causes and explanations for what they are told event if those are not explicitly present in the story (some recent work studies this aspects of storytelling [14]). To capture the effect of these operations on the overall rating, some variables operate over the number of facts that have been inferred or hypothesized during interpretation (which is computed by ad-hoc, domain dependent rules):

- *Compression* is defined as the ratio between the number of events that the reader infers and the number of events that story explicitly includes.
- *Hypotheses* measures the amount of knowledge that the reader hypothesizes when she reads the story. The *hypotheses* variable measures how many hypotheses have been made.

The final rating for the E is a linear combination of these variables. Although several ways of combining these values are possible, the unweighted mean value is used as the overall rating. While being a rather simple approach, the empirical tests

show good results using this approach. Other combinations could yield different values that are better fitted to human evaluation following the comparison in Section 4.

This set of variables is by no means exhaustive. One can think of several other plausibly valid variables, like surprise. The objective is not to create a full model, but to study the use of evaluation in story generation and creativity.

3.3 Implementation of the Evaluation Function

The evaluation function has been implemented as a rule based system. Domain specific rules have been encoded in an independent module in such a way that the general evaluation engine can be kept general enough to allow changes of domain at a later stage.

The E function is a knowledge intensive rule-based function that receives stories and iteratively processes them to compute a rating value. Thus, the evaluation function sequentially processes every event, just as a reader would do with written text. Although several psychological models and identification of variables regarding its influence on perception of narrative are available [6, 24, 12], none has been used. Only ad-hoc rules have been created for this prototype, so no psychological plausibility is claimed.

The evaluation function relies on a *context* Γ which stores the partial information state accumulated by a hypothetical reader as the processing evolves. This state includes a partial assignment for evaluation variables.

A rule has preconditions (that must be satisfied by the current context for the rule to be applicable) and postconditions that define the changes that should be applied to the current context to obtain the context after processing the event under consideration.

The *process* function searches in the rule base for rules whose preconditions match Γ_i and e_i. The effects of these rules create a new context which is returned by the *process* function, in this way updating the state of the evaluation. Equation 5 shows this relation:

$$\Gamma_{i+1} = process(\Gamma_i, e_i) \tag{5}$$

where Γ_{i+1} and Γ_i are the next and current contexts respectively, e_i , is the event being processed and *process* is the function that chooses which rules to apply and applies them.

For creating rules, the authors' intuition has been applied, with special focus on effectiveness of the rules for the working domain. The main objective for rules in the current prototype has been to demonstrate that such an evaluation function, at least for simple narrative, is possible.

The current prototype has 73 rules. Each rule is only applicable to one type of event. In the rule set there are rules for the *go* event, for the *take* event, and so on. This means that the "verb" in the event is the base when creating rules in the current prototype. Some example rules (translated to natural language) are shown in Table 2.

Not every rule is applied for every story. Only those rules whose preconditions are satisfied by the context are used, and the order of application is not important

Context	Event	Variable changes
x has to pay money to y **and** x did not pay **and** y is dangerous **and** y is in p	x goes to p	**raise** *danger* **and raise** *humanity* **and raise** *action*
x and y are not friends	x asks y for help	**raise** *humanity* **and raise** *tension*
z is the boss of x **and** z hates y	x kills y	**raise** *humanity* **and raise** *danger*

Table 2: Example of evaluation rules.

because the definition of rules only takes into account the story so far, not the partial results from other rules at previous stages stage.

The main flaw of this design is that the creation of the rules by hand is costly and the rule-set can not be easily updated without an extra effort to keep consistence on the knowledge base. This is a typical problem of rule-based systems, and it affects storytelling systems like BRUTUS [4]. Future work for this research contemplates the study of a possible automation of this approach.

4 Validation of the Evaluation Function using Human Judgment

It is necessary to validate the current model, at least to demonstrate that the task of modeling story evaluation is worth exploring. For this task, 10 stories generated by the exhaustive conceptual space exploration approach (as explained in Section 5) were issued to human evaluators (the set of evaluators is detailed later) and they were asked to order them by quality. Seven stories out of 10 were picked from those which the evaluation system rated as "good" (1, 3, 4, 5, 6, 8 and 9) and three from the set rated as "bad" (2, 7 and 10). The selection has not been totally random. Instead, the focus has been put on getting a sample of different stories with a broad range of values values from the evaluation function.

Eleven evaluators are male and eight are female, all Spanish. Their ages ranges between 24 and 59 years old and none of them are native English speakers, although all of them consider to have a high level of reading comprehension in English. Fourteen have graduate or post-graduate academic studies. None have any specialization in narrative.

Human judgements have been compared to the ordering that the evaluation function puts on the stories. The evaluation function creates this *quality order* by assigning a value to each story (as explained in Section 3.2) and then ordering stories accordingly. Stories used for the validation are shown in Figure 1.

Human evaluators are asked for a rating in the integer range $[1, 5]$ for every evaluation variable presented in Section 3.2. That interval has been chosen in order to use positive integer values instead of real numbers, which has been considered to be simpler for evaluators.

1. John was in the bus stop. A man was in the bus stop. John realized that it was late. He was surprised. John asked the time to the man, and he said that it was two o'clock. John supposed that he was going to die. Some time before, John had agreed with a godfather that he would pay him some money before 2 o'clock. John wanted to ask for help to the man in the bus stop. The man in the bus stop, then, said that it was too late, and he killed John.
2. John was in the bus stop. John went to the warehouse. John gave some money to a man. The godfather was the boss of the man. The man gave the money to the godfather. The godfather said to the man that John had to pay him that money. The guy said goodbye to John. John said goodbye to the man.
3. The godfather hated a man. The godfather was the boss of the man. The man was a friend of John. The godfather was the boss of John. John was the friend of the man. The godfather told John to kill the man. John killed the man. John was sad.
4. The godfather was sad. The man killed John some time before. The godfather desired John to be alive. The godfather told the man that he hated him. The man loved the godfather. The man was sad. The man killed himself.
5. The man desired that John was in the bus stop. The man was in the bus stop. The godfather told the man to kill John some time before. The man was afraid. The man wanted to escape. The man supposed that the godfather would get angry. The man escaped.
6. The man was angry. John was angry. The godfather told the man that he would pay him some money some time before. The godfather told John that he would pay him some money some time before. John supposed that the man would kill the godfather. John found the godfather. The godfather was dead. The man supposed that John had killed the godfather.
7. John took the gun. John was friend of the godfather. The godfather was friend of the man. The man was friend of John. The man had some money. The man gave some money to John. John gave some money to the godfather. The godfather gave some money to the man. John killed the man.
8. The godfather was surprised. The man had killed John before. The man told the godfather that it was late. The godfather told the man that it was 2 o'clock. The godfather took the money. The godfather gave the money to the man. The man was happy.
9. John was in the bus stop. The godfather was in the bus stop. The man was in the bus stop. John realized that the godfather took the gun. The man realized that the godfather took the gun. The godfather killed John. The godfather killed the man. The godfather killed himself.
10. John was angry. The godfather realized that the man was in the bus stop. The godfather told John that he supposed that it was late. The man took the gun. The man went to the warehouse. The man killed John. The man was surprised. The godfather told the man that he had killed John. The godfather was happy.

Fig. 1: Stories used for validation. They have been generated using the algorithm explained in Section 5. They have been translated to natural language using simple templates and text in some sentences has been corrected by hand.

These human evaluation values have been normalized to match the range $[-1, +1]$. Additionally, the opinion about the overall value has been gathered, in the same range. Nineteen people have been queried. Table 3 shows the mean gathered values.

Table 4 shows the values computed by the implementation of the evaluation function. The overall rating is the mean value of all the variables.

Figure 2 shows the comparison between the overall value computed by the implementation of the evaluation function for the test stories against the overall value gathered from humans. A very nice fitting can be seen between rating in human evaluation and the implementation. The mean quadratic error between the two sets of values is 6.21%. On the other hand, a perfect fitting would be difficult to interpret as strong validation of the evaluation function: there are so many aspects in story evaluation, and there are so many possible interpretations, that there is

	1	2	3	4	5	6	7	8	9	10
Interest	0.6	-0.32	0.24	0.51	0.29	0.24	0.07	0.07	0.69	-0.2
Causality	0.56	0.32	0.69	0.87	0.51	0.24	-0.07	0.2	0.24	-0.1
Compression	0.68	0.36	0.73	0.64	0.29	0.24	-0.02	0.29	0.38	-0.2
Danger	0.88	-0.16	0.69	0.6	0.47	0.64	0.78	0.64	0.91	0.78
Love	-0.32	-0.56	-0.02	0.51	0.02	-0.24	-0.33	-0.16	-0.11	-0.24
Tension	0.44	-0.16	0.29	0.38	0.42	0.42	0.24	0.02	0.33	0.2
Humanity	0.44	-0.28	0.2	0.64	0.51	0.38	-0.07	0.29	0.47	0.24
Action	0.4	-0.12	0.29	0.6	0.29	0.42	0.47	0.24	0.69	0.51
Hypotheses	0.32	0	-0.29	0.2	0.56	0.2	-0.16	0.16	0.33	0.45
Empathy	0.2	-0.6	0.11	0.29	0.24	-0.2	-0.24	-0.29	0.16	-0.07
Funny	-0.12	-0.44	-0.42	-0.29	-0.42	-0.42	-0.29	-0.42	-0.7	-0.47
Emotion	0.24	-0.56	-0.02	0.16	0.11	-0.24	-0.11	-0.11	0.24	-0.11
Chronology	0.84	0.48	0.6	0.78	0.6	0.38	0.29	0.69	0.87	0.02
Overall rating	0.64	-0.28	0.16	0.38	0.11	0.24	-0.16	0.2	0.56	-0.16

Table 3: Mean values for human evaluation of the set of stories.

	1	2	3	4	5	6	7	8	9	10
Interest	0,8	-0.2	0.2	0.6	0.2	0.2	-0.2	0.2	0.6	-0.6
Causality	1	-0.2	1	1	0.8	0.6	-0.2	0.2	0.4	-0.3
Compression	1	-0.6	1	1	0.8	0.7	-0.4	0.6	0.6	-0.5
Danger	1	-0.2	1	0.6	0.6	0.6	-0.4	0.6	1	0
Love	-0.2	-0.6	0.6	0.2	0.2	0	-0.2	-0.2	0.2	-0.5
Tension	0.6	-0.2	-0.2	0.6	0.4	0	-0.6	-0.2	0.9	-0.3
Humanity	0.9	-0.6	0.6	0.6	0.6	0.6	-0.2	0.2	0.4	-0.4
Action	0.2	-0.4	0.2	0.6	0.1	0.2	-0.2	0.2	0.6	-0.5
Hypotheses	1	0.6	-0.2	0.2	0.8	0.8	-0.2	0.6	0.2	-0.8
Empathy	0.8	-0.6	0.6	0.6	0.4	-0.2	-0.2	0.2	0.6	-0.7
Funny	0	-0.6	-0.4	-0.6	-0.5	-0.4	-0.6	-0.6	-0.2	-0.6
Emotion	0.2	-0.6	-0.1	0.2	0.1	0.1	-0.2	0.3	0.4	-0.7
Chronology	1	-0.2	1	1	1	1	-0.4	1	0.9	-0.7
Overall rating	0.64	-0.34	0.41	0.51	0.42	0.31	-0.31	0.24	0.51	-0.51

Table 4: Values computed by evaluation function for the set of stories.

not a *correct solution*. Therefore, a "nice" fitting of the evaluation function output values is, in general, useful enough.

This result suggests that the implementation and the selection of the variables is promising for simple stories as those we present.

There is a big deviation for story 10. The automatic evaluation system rates that story as a very bad one, whereas human do not set such a low value. This is because story 10 is meaningless for the current implementation of the evaluation function and humans tend to create meaning in a more powerful way. This case shows that a more detailed study of how humans assign meaning to stories must be made in order to add coverage for more complex stories in the evaluation function.

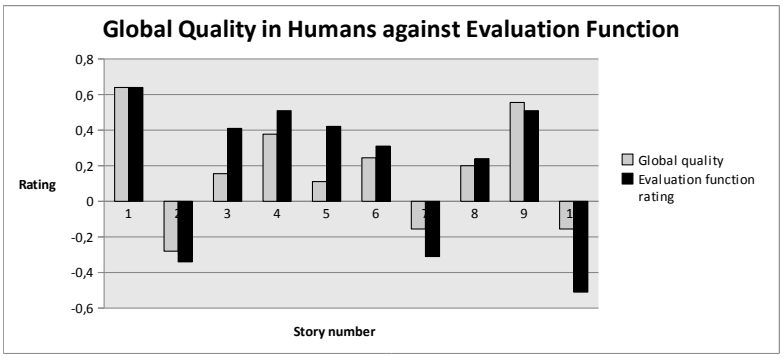

Fig. 2: Mean global quality variable against evaluation function.

5 Exhaustive Exploration of the Space of Stories for Generation

A simple *generate and test* approach, using PROLOG in the implementation, has been used to perform an exhaustive exploration of the $\mathscr{C}_d$ set of stories. The generation iteratively constructs the $\mathscr{C}_d$ set, and every created story is evaluated with the implementation of the E function.

The implementation consists on a simple generative approach based on an incremental strategy: a story with a single event is generated and evaluated, then a new story with that event and an additional one is generated, and so on. This is done for every combination of events and entities and considering the maximum length of stories ($\varphi \cdot \mu$, defined in Equation 2).

The order in which events are generated depends on the order in which event generation rules are ordered in the PROLOG program. Events are instances of basic actions (the set of possible events in the current prototype is described in Equation 2). Terminals in events are taken from the sets Δ, Π and ω.

The generation of messages is carried out incrementally by building messages for all available events.

Stories can be created based on the way messages are generated. First, all stories with one message are sequentially created. This means that the system will generate one story per different possible message. Then, the process is repeated for all possible stories of length 2, and so on until the number of messages reaches φ.

Two sets are created during generation: the set of *good* stories and the set of *discarded* stories. Stories are classified into one or the other based on the results of the evaluation function E described in section 3.3. A user given threshold τ is used to include stories in one set or another. Those stories whose rating falls above τ are good, and the rest are considered bad and they are included in the set of discarded stories.

The implemented prototype has 3 terminals for different characters, 2 different locations, 2 objects and 26 different types of events. Therefore, it can generate 473,979 different events. This number is obviously dependent on the number and the type of parameters of the event. For a maximum depth of 10 events per story and 1 event per message, according to Equation 6, the conceptual space $\mathscr{C}_d$ has a size of:

$$|\mathscr{C}_d| = \sum_{i=1}^{\delta} \mu^i = \sum_{i=1}^{10} 473,979^i = 5.7226 \cdot 10^{56} \text{ different stories} \quad (6)$$

5.1 Results of the Exhaustive Exploration Approach

A threshold τ of 0.01 has been used. The execution was run in a single core Intel Centrino 2.16 GHz computer with 3GB of RAM memory using SWI Prolog version 5.7.15 [1] on an Windows 7 machine.

After 4 hours and 25 minutes (99% of processing time for the generation) the execution was manually stopped. In total, 15,932,143 stories had been created and evaluated. Only 4,234 stories received a rating over the threshold (0.01), which means a rate of only 2 good stories in every 10,000 generated stories. Not every story rated as good by the implementation was high-valued according to humans and many discarded stories were probably high-valued. As checking several million stories by hand is quite a difficult task, only a random sample has been picked from the set of good stories to check that the stories are, at least, meaningful. "Bad" stories in the discarded set are also checked, and most elements of the "bad" set were found to be meaningless. A subset of this sample and some stories in the discarded set were chosen as the test set for the evaluation function explained in Section 4.

From the set of good stories, the mean overall quality value was 0.14. The best story received a rating of 0.64:

> John was in the bus stop. A man was in the bus stop. John realized that it was late. He was surprised. John asked the time to the man, and he said that it was two o'clock. John supposed that he was going to die. Some time before, John had agreed with a godfather that he would pay him some money before 2 o'clock. John wanted to ask for help to the man in the bus stop. The man in the bus stop, then, said that it was too late, and he killed John.

Only 319 stories received a rating over 0.25 (7.53% of stories). It is possible to state that (approximately) this should be the set of stories that are not only meaningful but also high-valued to some extent, although checking all these stories and evaluating them using human criteria would be needed.

6 Improving Conceptual Space Exploration by Constraining

The previous section showed that the amount of stories that the system can generate is so high that practical generation is unmanageable. In this context, it makes sense to constrain the bounds of the conceptual space so that the number of generated stories becomes smaller. The previous section showed that 99,97% of stories were discarded. Considering that the required time for generating any story is approximately equal, a large percentage of total computing time was spent on incorrect stories. Also, humans do not achieve creativity by trying all possible alternatives and then choosing the most appropriate one.

These two aspects of the exhaustive generation approach have led to the modification of the basic generation system to include the possibility of a *pruning function*, $\mathscr{P}$. The second version of the prototype includes a domain dependent

function that returns a boolean value: if it returns true, the current branch in the generation is explored. Otherwise, the generation stops at that point for the current branch, making the algorithm try a different option.

Although the current prototype permits any implementation of this pruning function, we want to explore the possibility of basing this pruning on the evaluation function that has been defined in Section 3.2. The definition of this pruning can be formally represented by Equation 7:

$$\mathscr{P}(s) = \mathscr{E}(s) > \tau \tag{7}$$

where τ is any real parameter in the range $[-1, 1]$, as defined in Section 5.

6.1 Adapting the Evaluation Function for Use as a Pruning Function

To adapt the evaluation function so that it can be used as a pruning function, an iterative modification based on the exhaustive generation approach results has been carried out. The overall point is to adapt the evaluation function in such a way that it detects unfinished stories, and treats them so that an adapted procedure can be applied to them.

If a partial story is evaluated as if it was a complete story, the overall rating it is likely to be quite low. If it was discarded before completion, the system would be discarding a potentially high-valued story. The basic E function must be adapted so that it considers partial stories in a different way.

This is done by providing a set of additional rules. New rules have the same format as those described in Section 3.3, but they are designed so that they avoid low rating of partial stories when extensions of the story may achieve high ratings. If the story rates as an unfinished story above a user given threshold, the partial story is considered to be promising and it is extended. Otherwise, the story is considered not promising and it is discarded.

The definition of the rules is based on experimental results. The process of creating good rules for constraining the search in the conceptual space takes four steps:

1. Using the current E function (the function which *does not* take into account partial stories) a sample of stories L is generated. The size of this sample is kept small so it can be checked manually.
2. Using the E_p function (the function which *does* take into account partial stories) the generation is repeated, getting the $L^{'}$ set (in less time).
3. Checking the generated logs from the execution in step 2 in comparison with those in step 1, those non-promising stories (according to E_p) that yielded good stories (according to E) are identified as the θ set.
4. Each rule in E_p is adapted so that it accepts the stories in θ as promising. This process involves several evaluations of every element in θ, trying new rules and testing them so that the evaluation fits the authors' criteria.

Steps 2–4 are repeated until $L = L^{'}$. When this happens, a different sample set is chosen for generation.

6.2 Results of the Constrained Conceptual Space Exploration Approach

To compare both approaches (exhaustive exploration and constrained exploration), the constrained version was run to generate exactly the same number of good stories than in the previous experiment, 4,234. Using the same machine, the generation took 53 minutes. The obtained set of stories, however, was not the same, so the application of the pruning function does not only affect time, but also the output set itself in terms of Wiggins' formalism. This suggests that the same $\mathscr{T}$ combined with a different $\mathscr{E}$ leads to a different set of points within the conceptual space.

Using constraints in this way, the mean value for stories rose to 0.22, and 814 stories received an evaluation greater or equal to 0.25. That is, 19.22% were "good" stories. Again, this should be proved by checking the real quality in stories. However, the maximum value was 0.60, which is slightly lower. It corresponds to the story presented below. It is quite similar to the best story in the exhaustive exploration presented in Section 5.1:

> John realized that it was late. John was in the bus stop. He was surprised. John asked the time to the man, and he said that it was two o'clock. John supposed that he was going to die. Some time before, John had agreed with a godfather that he would pay him some money. John wanted to ask the man in the bus stop for help. The man in the bus stop killed John.

These results show that constraining the conceptual space search saves time and discards non-promising stories.

7 Discussion

It is clear that approaches to storytelling that stake out a very large conceptual space and then proceed to filter it exhaustively in search for good solution is inefficient from a computational point of view. This had already been identified by Johnson-Laird [8]. In his analysis, Johnson-Laird discusses neo-Lamarckian as opposed to neo-Darwinian approaches, in terms that match very closely our description of the two alternatives to refining a computational creative system. The choice of focusing the refinement on the traversal function would correspond to a neo-Lamarckian approach and the choice of focusing on the evaluation function would correspond to a neo-Darwinian approach. A purely neo-Darwinian solution in this sense would not be of practical applicability in contexts such as interactive storytelling or story generation assistants, where response time can be crucial. This, to some extent, justifies efforts made to obtain more efficient search algorithms by identifying an initial traversal function capable of coming up with a set of good stories (however small) and then progressively refining the generation process (as represented by the traversal function) in the hope of adding more good stories as possible solutions. There are currently a number of research efforts following this different approach with moderate success in terms of practical applications.

However, from an academic point of view, the neo-Darwinian approach presents significant advantages. The fact that the whole context of operation is described (including a model of the evaluation function and the partition it provides of the conceptual space into good artifacts and discarded artifacts) allows a more detailed characterisation of the processes involved. Because the actual algorithms

employed for traversing the conceptual space are very simple, solutions built along these lines can generate massive amounts of data. Whereas other story generators produce a small number of high-valued stories from any given set of starting data, exhaustive approaches are capable of generating millions of stories. The use of a properly validated evaluation function allows the system to filter this huge data set to provide usable solutions. The main advantage of the exhaustive approach lies in the fact that it will explore all possible solutions, independently of whether the author of the system has conceived particular combinations as possible sources for high-valued stories, according to humans. In this sense, whereas conservative story generators will only produce stories that their authors might have predicted, an exhaustive story generator will produce stories that surprise its authors.

The quality of the results generated by the exhaustive approach is directly related to the quality of the evaluation function E as a model of how a human reader reacts to stories. As the evaluation function is refined, the quality of the resulting stories will improve. This approach shifts the focus from modelling the construction processes to modelling the evaluation methods. This shift entails a transition from relying on knowledge-based solutions to construct artifacts (at which computers are notably worse than humans) to exploiting brute force approaches based on very simple computations (which computers excel at) guided by an evaluation function. The onus is then on a successful modelling of the evaluation function. If this goal is achieved however partially, the authors believe that it would open the door for a surprising explosion of what might be perceived as computer creativity.

However, the model of the evaluation function is itself a knowledge-based solution. One major concern about knowledge intensive systems is their scalability. Rules are heavily coupled with the definition of the domain. Each time a new type of event is added, several rules have to be adapted to correctly process it, because this event could be modifying the whole meaning of the context. To reduce the impact of this coupling, strong effort has been put on designing the system so that both the basic generation engine (Sections 5) and the theoretical description of the evaluation function (Section 3.2) are reasonably independent from the definition of rules for implementing the evaluation (Section 3.3 and 6.1). The design contemplates division between domain knowledge and evaluation knowledge (which uses the former), so domain knowledge is not polluted with the model of the reader. The rule interface explicitly states input and output values, and the reader model attributes are general enough to be managed by rules. This said, scalability is improved by the design, but not assured for ever. Once the model grows enough, rule handling will probably become very hard and either a different approach or a model redesign will be needed.

8 Conclusions and Future Work

Two approaches to story generation, both relying on a model of story evaluation, have been explained. How generation of high-valued stories can be achieved, according to human criteria, by two different approaches based on this evaluation function, is also studied. An example execution of both generative processes and the adaptation of the system in order to get a system that can be used in real environments is also shown.

The evaluation function has been developed based only on the first author's intuitions. Although it is possible to claim that given the limited generative abilities of the system, there is no need of specific expertise to evaluate the quality of the evaluation and therefore the generation, applying psychological models of reading and narrative models may provide an important improvement on the system. The current implementation has been successful in demonstrating the plausibility of the main hypothesis (it is possible to create a partial model of the reader for the evaluation of stories), but more precise information about how the mind processes narrative is crucial for the system. Considering different parameterizations for rules (different modifications when updating evaluation variables) is also important, that is, computing different ratings in event evaluation. The implementation shown in this article has been tuned heuristically in order to classify good stories, but many other different parameterizations are of course possible. Different adaptations could even be the key to model different readers, which will be addressed in future implementations.

Acknowledgements This research is funded by the Ministerio de Investigación, Ciencia e Innovación (MILES: TIN2009-14659-C03, GALANTE: TIN2006-14433-C02-01), Universidad Complutense de Madrid and Dirección General de Universidades e Investigación de la Comunidad de Madrid (IVERNAO: CCG08-UCM/TIC-4300) and by the Creation and Consolidation of Research Groups Program by Banco Santader Central Hispano-Universidad Complutense.

Thanks to anonymous reviewers for invaluable comments which have improved this paper greatly and have given us many ideas for the further development of this work.

References

1. SWI-Prolog. http://www.swi-prolog.org/.
2. Margaret Boden. Computational models of creativity. *Handbook of Creativity*, pages 351–373, 1999.
3. Margaret Boden. *Creative Mind: Myths and Mechanisms*. Routledge, New York, NY, 10001, 2003.
4. Selmer Bringsjord and David Ferrucci. *Artificial Intelligence and Literary Creativity: Inside the mind of Brutus, a StoryTelling Machine*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1999.
5. Pablo Gervás. Computational approaches to storytelling and creativity. *AI Magazine*, 30(3):49–63, Fall 2009.
6. A. C. Graesser, M. Singer, and T. Trabasso. Constructing Inferences During Narrative Text comprehension. *Psychological Review*, (101):371–395, 1994.
7. Kyle Jennings. Developing creativity. Artificial Barriers in Artificial Intelligence. In *Proceedings of International Joint Workshop on Computational Creativity*, 2008.
8. P. N. Johnson-Laird. Mental models in cognitive science. *System Design for Human Interaction*, pages 17–39, 1987.
9. P. N. Johnson-Laird. How jazz musicians improvise. *Music Perception*, 19(3):415–442, March 2002.
10. M. Lebowitz. Storytelling and generalization. In *In Annual Conference of the Cognitive Science Society*, pages 100–109, Berkeley, California, 1987.
11. Douglas B. Lenat. Eurisko: A program that learns new heuristics and domain concepts. *Artificial Intelligence*, 21(1-2):61–98, 1983.
12. M. Mateas and A. Stern. Structuring content in the Façade interactive drama architecture. In *Proceedings of AIIDE*, pages 93–98, 2005.
13. J. Meehan. *The Metanovel: Writing Stories by Computer*. PhD thesis, 1976.
14. James Niehaus and R. Michael Young. A Computational Model of Inferencing in Narrative. In *Intelligent Narrative Technologies II*. AAAI Press, 2009.

15. F. Peinado and P. Gervás. Evaluation of automatic generation of basic stories. *New Generation Computing, Computational Paradigms and Computational Intelligence. Special issue: Computational Creativity*, 24(3):289–302, 2006.
16. R. Pérez y Pérez. *MEXICA: A Computer Model of Creativity in Writing.* PhD thesis, The University of Sussex, 1999.
17. M. Riedl. *Narrative Planning: Balancing Plot and Character.* PhD thesis, Department of Computer Science, North Carolina State University, 2004.
18. M. Riedl and R. Young. Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space. *New Generation Computing, Computational Paradigms and Computational Intelligence. Special Issue: Computational Creativity*, 24(3):303–323, 2006.
19. Riedl, Mark Owen and Young, R. Michael. An Intent-Driven Planner for Multi-Agent Story Generation. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 186–193, Washington, DC, USA, 2004. IEEE Computer Society.
20. K. Sawyer, V. John-Steiner, S. Moran, R. Sterberg, D. Feldman, J. Nakamura, and M. Csikszentmihalyi. *Creativity and Development.* Oxford University Press, 2003.
21. M. Sharples. An account of writing as creative design. *The Science of Writing*, 1996.
22. M Sharples. *How We Write.* Routledge, 1999.
23. Scott Turner. *MINSTREL: A Computer Model of Creativity and Storytelling.* PhD thesis, University of California at Los Angeles, Los Angeles, CA, USA, 1993.
24. P Weyhrauch. *Guiding interactive drama.* PhD thesis, Computer Science, Carnegie Mellon Univ., Pittsburgh, PA., 1997.
25. G. Wiggins. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7), 2006.
26. G. Wiggins. Searching for Computational Creativity. *New Generation Computing, Computational Paradigms and Computational Intelligence. Special Issue: Computational Creativity*, 24(3):209–222, 2006.