From socio-emotional scenarios to expressive virtual narrators

Roman Miletitch
Laboratoire d’informatique de Paris 6 (LIP6)
4 place Jussieu 75005 Paris, France
Email: roman.miletitch@gmail.com

Nicolas Sabouret
Université Pierre et Marie Curie
Email: Nicolas.Saboure@lip6.fr

Magalie Ochs
CNRS - LTCI
Email: magalie.ochs@gmail.com

Abstract—Telling a story requires linking a series of significant events using statements to maintain tension, create suspense and allow time for the development of emotions. When automating this process, one major challenge is the generation of these filling sentences and ensuring that they are sufficiently consistent with the story. To this end we propose in this paper to use a knowledge representation model of a local coherent world. We present a scheme for automatic storytelling based on an ontological representation of concepts and natural language generation algorithms that dynamically build relations between concepts. Our algorithms scan the scenario to build a sentence skeleton that will be enriched using pseudo-random queries in the ontology. This allows us to enrich the story while keeping the storyline coherent. We end by discussing our evaluation and presenting our preliminary results.

I. INTRODUCTION

In order to tell a story, the story teller should not only relate the events, but also make the readers/listeners feel emotions along the narration. Enriching a story, catching the reader’s attention and stimulating the senses are ways to create a moving story, making it memorable [1].

One has to keep in mind that telling a story is not limited to oral communication, but also involves the creation of an atmosphere through senses, in which all told events are emphasized one after the other [3]. Beyond these sensual elements, the motivation to pursue reading is the interest that the story brings and the tension it creates through the distribution of this interest.

To this end, one way to catch readers’ attention is to stimulate emotions [5], and this is where empathy comes into play. We think that the storyteller shouldn’t maintain a neutral and cool frame but instead should exaggerate his or her emotions in order to better share them. Telling what emotions one actor feels is not enough, the narrator should express them, by choosing specific and appropriate words, and through gesture and facial expression.

In section 2, we discuss existant works in automated storytelling, and tackle some of the topics mentioned in this introduction. Then, in section 3, we present our model and the algorithms. Finally, in section 4, we will show our evaluation and results.

II. EXISTING WORKS

The virtual Story Teller [14] is a multi-agent framework that dynamically creates a story. Its agents are guided by objectives that the narrator gives, and are controlled by an agent called the ‘director’. The objectives enable a scenario to emerge from a rough sketch or outline. This framework shows us that the indetermination in the creation process of a story is not a flaw, and can bring interesting results, which is something we will hopefully be able to demonstrate in our model. Translating the scenario into a story is the job of two agents: the narrator and the presenter. This separation is found in numerous systems and leads us to choose two different classes of algorithms: one for the construction of these sentences on a semantic level, and another for their superficial realisation in natural language.

Cavazza deals with the integration of natural language generation inside an emotional framework in [7] that is inspired by the work of Flaubert. This research also benefits from the fact that Flaubert used what would nowadays be called an ontology for his writing. We note here again the interest of using emotions in a narration, which will be the base of our model in addition to using an ontology.

Jean Louis Dessalles [9] suggests the establishment of a model of interest based on communication events, which can be applied to storytelling. He defines the interest of an event as the difference between its complexity of occurrence, and its complexity of being told. For instance, winning the lottery is something easy to tell, but that is considerably rare in occurrence, at least on an individual level, hence the interest of telling it. One aspect that adds complexity is the improbability of the event, as seen in the lottery example. Being improbable, the event arouses curiosity and some kind of suspense because one wonders how it could happen. Finally, he shows that the interest of one event is proportional to the contrasting emotional intensity that it triggers [8], no matter what kind of emotion it is. We base ourselves on this model in order to get our algorithm to spot when and which emotions are to be emphasized in the story.

OSSE Model The source of our work is a software which aims at creating a dynamic model of the social and emotional state of characters based on a scenario : OSSE [13]. This model focuses on personality factors that trigger emotions,
and on the influence of these emotions on the behaviour of the characters, and on their social relationships. OSSE’s vocabulary is divided in 4 sets: \( V_{\text{OSSE}} = O \cup V \cup R \cup P \), where \( O \) is the object and game character set, \( V \) is the action set (verbs), \( R \) is the set of all possible roles for the characters to play, and \( P \) the character set. The roles are linked by social relationships: \( \text{RelSoc} : R \times R \rightarrow \{ \text{Agreeability}, \text{Dominance}, \text{Solidarity}, \text{Familiarity} \} \).

Characters are defined as: 
\[
\{ \text{Name}_p, \text{Personality}_p, \text{Emotions}_p, \text{Roles}_p, \text{Attitudes}_p \}.
\]

This vocabulary allows us to define events \( e \in \text{Evt} \) as 
\[
e = \{ \text{Agent}_e \in P, \text{Verb}_e \in V, \text{Patient}_e \in O \cup P, \text{date}_e \in N, \text{WhoWatching}_e \subseteq P \}.
\]

The \( \text{Agent}_e \) defines who is acting, \( \text{Verb}_e \) defines what the action is, and \( \text{Patient}_e \) defines who undergoes the action. The \( \text{date}_e \) lets us organise the events’ chronology. Finally \( \text{WhoWatching} \subseteq P \) corresponds to a list of characters who observe the action.

A scenario is an ordered set of events. When each event is played out, the social state and social relationships of the characters watching and undergoing that actions are updated. The OSSE model takes all those parameters as an input, and outputs the evolution of social relationships value between characters, and the value of their emotions.

### III. Our model

Our starting point is the event formalism of the scenario’s description defined in the OSSE model. We’ll be referring to them as **triplets**, since the most important information is the Agent, the Verb and the Patient. This representation is simple enough to be used by game designers with little or no background knowledge in programming language or artificial intelligence. But this simplicity leads to the question of natural language generation for the expressive virtual narrator. Hence our objective is, from a sequence of events and a limited set of vocabulary, to enrich a story with coherent sentences that would reinforce its interest and its emotional impact on the people listening to or reading the story without fundamentally changing it.

We base our work on two hypotheses:

- A simple and local modelling of the world, through concepts and relations in an ontology, can create enough sense to strengthen the interest, sensual and emotional effect of a story
- The pseudo random generation of possibly non-relevant sentences or parts of sentences does not interfere with the narration. Instead, it can actually help to enrich it because the viewer/reader/listener will create the missing links by himself or herself [21].

#### A. Global architecture

From the creation of events to the telling of the story, we will use two modules. The first one is OSSE, which will take care of socio-emotional modelling, and our own module, which is composed of an ontology and scanning algorithms that work to make sense emerge from it [19], in order to create an actual sentence.

1) **Ontology:** An ontology is the specification of the conceptualisation of a knowledge field [11]. It is itself a knowledge representation modelling of a set of concepts in this field, and a set of relations between these concepts. Our ontology is inspired by a subject close to that of Cavazza [7] and by a style close to the conceptual graphs of Sowa [18]. The original ontology defines a new set of vocabulary, linked with OSSE’s one, based on concepts and relations: \( V_{\text{Onto}} = \text{Con} \cup \text{Rel} \). Each OSSE’s concept in \( V_{\text{OSSE}} \) is attached to one element in \( V_{\text{Onto}} \). The concepts are organised in a hierarchical way, and even if this organisation is subject to change by the game designer, there is a skeleton that can’t be altered, designed specifically to act as a guide. In addition to these concepts, our ontology will be described by relations \( r \), defined by 
\[
r : \text{Con} \times \text{Con} \rightarrow \text{Rel},
\]
which allow two concepts to share a special link between them, defined by the same kind of relation. The use of such a representation that also enables inheritance, makes it possible to define high level concepts on a broader scale [15], and makes their sub-concepts inherit their parent’s qualifiers, which makes the creation process easier for the game designer.

Using an ontology, it is possible to step out of the limited frame of the scenario and to enrich the story by randomly selecting associated concepts (i.e. concepts in relation with these that appear directly in the scenario). Thus, the story is more varied, less predictable, even for the author himself, which is hardly possible with scripted approaches. This is why we try to work with concepts rather than with a string on a semantic level. This allows us to better play with meaning and sense, and intensify the emotional side of the story [17].

2) **Generation algorithms:** The ontology makes it possible not only to describe the world, but also to structure it, so as to draw meaning from it. However, this meaning is only visible when used in the generation process, where information is turned into knowledge by our scanning algorithms. To sketch these algorithms, we were inspired by emotional impact theories and interest’s modelling from Jean Louis Dessalles [9]. There are eight algorithms, and each is a way to maintain control over the dramatic development of the story. Two are significantly different from the others: they are the introduction and conclusion algorithms, which create an initial atmosphere and an epilogue respectively in such a way that the story is not too abrupt. Five others are cast at each date of events. They generate links between the ontology and triplets from each event at a given date:

- **Event Realisation** translates the sequence of events from \( \text{date}_e \), it is the story structure created by the game designer
- **Suspense** creates a sentence linked to the event, in order to build tension
- **Social Relationship** tries to reinforce the emotional aspect of the story by generating side-sentences inspired by the relationships between characters
- **Stand Alone** creates stand alone sentences, without real impact, as a way to release the tension
- **Narrator POV (point of view)** gives the opportunity to
the virtual narrator to express emotions and react to the events depicted at date_

- **Filling** deals with finding qualifiers (such as adjectives, adverbs, personality traits or social roles) to describe the items, in order to add complexity to the story, and to bring a more varied experience to the reader.

These algorithms (detailed in section C) describe the ways to alter the tension and emotional impact of the story. It is possible to adjust these algorithms in order to change aspects of the story, especially with the narrator parameter. Indeed, the story will not be told the same way by different narrators. For instance, if you are the thief who has been arrested, your point of view of the story will be different from that of the policeman making the arrest. All six enriching algorithms must not be invoked for each event in the scenario: this would result in a very unnatural, overwhelming story. Thus we have associated each algorithm with a selection probability which regulates their use and creates a more varied tension pattern.

The randomness of their appearance and of the meaning generated can create some redundancy, but it is not usually felt as repetitive, and has its importance in natural language [20].

Even if these algorithms base their working on informations from the sentence, they put chance as a parametris, which lead the result to be less predictable, and not as precise as one would hope. But we made the hypothesis, that these inconsistencies don’t kill the story when there isn’t too many of them. This was confirmed by later evaluation. Moreover, they prevent the story from seeming too flat, like a computer generated story would be expected to feel like. The main work in setting the different thresholds of the various algorithms was to find an equilibrium between that roughness and the consistency of the story. Lastly, every relations created or taken from the ontology is put in memory, so that we don’t create contradictory relations. This helps preserving the consistency of the story.

3) **Realisation Algorithms:** When the links between the sequence of events and the ontology are made, realisation of these links are formed into a sentence. We coded superficial realisations algorithms in order to transcribe the meaning generated by the scanning algorithms from the ontology into a sentence, and to add emotional and behavioural tags that simultaneously guide the sentence’s interpretation by a virtual storyteller. The aim of both of these algorithm classes is to use the item organisation for as long as possible, since once it’s translated into a string, you can no longer generate semantic information and links in an easy way. As for the introduction and conclusion algorithms, we use sentences with gaps to be filled depending on the full set of events. They introduce the major characters and their relationship (defined as the characters that appear most throughout the course of the story), and one of the principal places.

4) **Sentence’s grammatical organisation:** As mentioned above, we decided to divide the sentence into various items: $item \in I = \{concept \in Con, gram \in \{Subject, Verb, Directobject\}, qualifiers\}$ where

qualifiers is a set of descriptions of the concept which defines the word (mainly adjectives and adverbs for now), and gram its place in the sentence. A sentence is defined as a set of items: $sentence = \{item_0, ... , item_n\}$, and a paragraph as a set of sentences: $paragraph = \{sentence_1, ... , sentence_n\}$.

In order to create a paragraph at a given date, we first create sentences from the events, then we precede it by the simple sentence of suspense, the simple sentence of social relationship and the stand alone sentence. Then we add qualifiers and try to organise the sentences and mix them. Finally we make the superficial realisation, and we add the narrator’s sentence, which forms a string.

B. **Ontology**

To begin with, all scenario concepts are organised in a hierarchy. This initial ontology is enriched by the scenario author using libraries or his or her own imagination with:

- **missing concepts relevant for the hierarchy structure** (ex: the scenario uses the concepts safe and door, but never mentions key or lockpick). Following an RDF-like approach, we make no distinction between instantiation (instance of) and inheritance (subclass of) in our hierarchy. Leaves are instances and other nodes correspond to classes. For instance, let us imagine that we have a concept key inheriting Small object and the concept door’s key inheriting key. The concept key is a parent and so a class, and door’s key is a leaf, so it makes an instance. When our algorithms casts key, they use it as “a key”, but with door’s key, they use “the door’s key” as a direct reference.

- **relations between these concepts** (e.g.: a Cop can arrest a Thief). These relations are potential relations; they are not yet instantiated. For instance, the arrest relation between Cop and Thief means that the cop can arrest the thief, not that it has actually happened.

In addition, we impose that concepts and relations are associated to the “verb”, “adverb”, “noun” or “adjective” typology for NL generation. This results in an ontology structure following the schema on figure 1. For instance, in our original implementation we decided to separate big objects from small objects, which makes our ontology closer to common sense than usual ontologies. This choice of hierarchy is linked to the relations, and hence, to the algorithms that are implemented.

Our aim wasn’t to construct a heavy ontology, but a small one in order to show that such a methodology was not only pertinent, but also scalable. For instance, the ontology we used for the later tests had little more than one hundred concepts. The variety of sentences produced by the algorithms depends a lot on the ontology’s size, the kind of the algorithm, and the triplets. With our small ontology, each algorithm could trigger around 50 different sentences in average.

**Base-structure of the Ontology:** The NL generation algorithms we programmed depend on the structure of the ontology which is not only defined by its hierarchy but also by the relations in play. In our case, these relations are binary, and directional. A relation is defined by its kind, and the
two concepts it links (limited to a certain kind of concept depending on the type of relation). There are numerous types: positioning which let the designer place characters and objects on the stage; grammatical, defining which subject and direct object can fit each verb; qualifiers (different types if it’s a verb, a person or an object we’re referring to); and more semantic kinds, like \textit{Obj-Vrb} which links a verb and an object that is meaningful to it, like \textit{to eat} and \textit{fork}, or \textit{to read} and \textit{book}. Our goal here was not to create an exhaustive list of relations, but a simple and relevant architecture, and to give way to the author’s imagination so that new kinds of relations can be created in the future.

In order to prevent the game designer from creating meaningless, or worse, absurd relations, every relation has a set of acceptable concepts. For instance, contain relations can only link \textit{places} with \textit{object} and \textit{being}. As described above, our relations can be either potential or instanced. The relations that are created before the execution of the story are by definition potential. They describe a state that is not yet expressed. If we have the relation “the room contains a table”, it is potential until an algorithm fixes it by generating a sentence from it. The relations that are created in the process by our algorithms are directly used in the story line, and in this way, they become instanced relations. Potential relations are seen as usable material in order to generate meaning, and instanced relations as already expressed meaning, which should not be contradicted. This is why relations created during the story development process are stored as instanced relations. This way, we can keep them in memory and check them when we create new relations as to maintain story consistency.

\textbf{Diversity:} Our model is not scenario-specific. When an author defines a new scenario, it has to attach all concepts to our hierarchical structure for the NLG algorithms to work. This work can be time consuming, but it is also possible to define libraries of concepts and relations attached to different scenario categories, and to import them to the ontology. This way, the author would always use some standard library with all typical concepts, and add more specific libraries in order to create an atmosphere. That would also be a way to set the parameters of the narration. If you want your sci-fi story to have more of a film-noir ambiance, then it would be necessary to switch from one to another library.

\section{Scanning algorithms}

As mentioned before, the introduction and conclusion algorithms are different from the other NLG algorithms, and their role is to build an atmosphere, so that the reader already knows the characters in the story to some extent at least at the beginning, and is granted an epilogue for the tension to be released at the end. Five other algorithms deal with this tension throughout the story. The first one is Event realisation, the simplest, which only transcribes the list of events in a sequence of items, so that future algorithms will be able to process it more easily. This part of the paragraph matches the climax, where the tension is at its peak before a release, and is scripted by the author. The Suspense algorithm goes straight from the events sequence, and tries to generate a sentence that precedes the events, trying to build an expectation for the reader, creating in the process a tension in the script. Its working is explained in the algorithm 1.

\begin{algorithm}
\caption{Algorithm Suspense}
\begin{algorithmic}
\Require \texttt{paragraph}
\State 1: choose at random \texttt{stc = sentence in paragraph}
\State 2: choose at random \texttt{cod'} \in O \cup P \texttt{ so that}
\hspace{1em} \texttt{cod' \rightarrow Obj_Vrb \rightarrow stc.Verb or cod' = stc.COD}
\State 3: choose at random \texttt{Verb'} \in V \texttt{ so that}
\hspace{1em} \texttt{cod' \rightarrow Obj_Vrb \rightarrow Verb'}
\State 4: create sentence : \texttt{Sente = \{stc.Subject, Verb', cod'\}}
\State 5: \texttt{paragraph.addSentence(Sente)}
\end{algorithmic}
\end{algorithm}

Let’s take for instance a triplet from a paragraph: \{Tuco, to open, safe\}. We must instantiate a new \textit{directobject}, which is either the current patient, or a concept linked to the verb through \textit{Obj_Vrb} which in our case leads us to \textit{safe} or \texttt{\{key, lockpick\}}. Let us take \textit{key}; the next step is to get all the verbs that are linked to it through the same relation, which is here: \{\textit{to use, to watch, to play with}\}, the last of which we select. The last step is keeping the same agent, which gives us the triplet: \{Tuco, to play, key\}, which after realisation is read as: “Tuco plays with the key. Tuco opens the safe.”. We see here that the suspense has been created by a trivial action which has a heavy link with another action that will follow. They differ qualitatively, but have a semantic link, which makes it close to human narration.

Next is the Social Relationship algorithm 2, which refers to the characters on the stage, generating other sentences depending on their value of agreeability, dominance, solidarity and familiarity. We did not want to just describe a situation (“Tuco is familiar with Maria”) but to create a situation where the characters expressed their feelings without interfering too much with the scenario (“Tuco winks at Maria”). We rely on two thresholds (min and max) to select wether the social attitude should be expressed or not. To reflect the fact that some social attitudes are more often expressed than others,
we have different values for min and max, depending on the considered social trait.

Algorithm 2 Social Relationship algorithm

Require: paragraph
1: list all characters \( C = \{c \in P, c \in \text{paragraph}\} \)
2: choose at random \((c_1, c_2) \in C\)
3: get SocRel from \((c_1, c_2)\)
4: choose at random SocRel.char in \{Agreeability, Dominance, Solidarity, Familiarity\}
5: if SocRel.char \(> \max(\text{SocRel})\) then
6: addSentence(SocRel.char is high)
7: else if SocRel.char \(< \min(\text{SocRel})\) then
8: addSentence(SocRel.char is low)
9: end if

For instance, let’s imagine a paragraph made of: \{Tuco, to open, safe\}, \{Maria, to look, Tuco\} and \{Bank Man, to think, money\}. The character list is \{Tuco, Maria, BankMan\}. Supposing we choose Maria and Tuco, we would then have to check their social relationship. We choose a random social relationship from our list of four: let us say ’dominance’. The value of their dominance social relationship is given by OSSE, which purpose is indeed to dynamically calculate those social and emotional values. Let’s suppose it’s 0.7 and over the dominance high threshold, favouring Maria. We then create a new triplet, by choosing a verb in the list associated to the threshold, specific to this information. In our case, it will be: \{Maria, to despise, Tuco\}. Each relation has specific min and max thresholds, and a list of associated verbs for each threshold. In this case, for dominance, the max threshold was 0.6 and the verb list was \{to despise, to abhor, to look down, to snub, to patronize\}.

Next is Stand Alone, with no relation to the events present, past or future. This neutral sentence is a way to calm the rhythm, which is often too dense in information because each OSSE's events is supposed to be emotionally intense. Of course, these are sentences one should not abuse in frequency, since they are void by definition (“Tuco thinks for a while”, “Clint watches Maria”). Finally, the last of the five generating algorithms is Narrator POV, which gives importance to the narrator, and allows him or her to show his or her emotions [12]. By giving an identity to the narrator, we try to give a different edge to the story. This is on the basis that the interest in a story depends a lot on who is telling it and the involvement of the narrator [2], which is why we try to generate sentences that will grant the narrator a degree of importance, and incidentally, to emphasize particular emotional events in a subjective way. We intend it as a release, as the narrator shares his or her moods, state of mind and feelings with the readers. The narrator is a copy of a character from the scene. At each event where the POV algorithm is triggered, we choose one of the character who acts in this event. Then we ask the OSSE engine how this character is currently feeling, and how the narrator feels about him. We then outputs a scripted sentence depending on those informations.

The last algorithm, Filling, is different. Its aim is not to generate sentences, but to fill them, to link the triplet items with relevant qualifiers. For that it scans all the pre-generated set of items and seeks any opportunities in the ontology, in its relations, in OSSE’s proprieties to add qualifiers such as adjectives, positioning relations, personality traits, social roles etc. These qualifiers are meant to give a more realistic touch to the story. They give a virtual history to the qualified object, and the reader will try to seek a meaning to this in the course of events imagined in the story[21]. The qualifiers are added randomly to characters, objects or verbs, depending on the algorithm 3.

Algorithm 3 Filling algorithm

Require: paragraph
1: \(\forall s \in \text{paragraph}\)
2: choose at random \(k \in [0, 100]\)
3: if \(k > 35\) then
4: Fill(sentence.getPersonne)
5: execute Algorithm Filling again
6: else if \(k > 50\) then
7: Fill(sentence.getPersonne)
8: else if \(k > 66\) then
9: Fill(sentence.getObjet)
10: else if \(k > 83\) then
11: Fill(sentence.Verb)
12: end if

Since we always grant more importance to the characters in the story, we made the probability higher for them to be granted a qualifier. Even more, since adding a qualifier to a character was felt as less likely to feel odd in the case of multiple qualifiers, we included an odd of executing the algorithm again. For instance, dealing with \{Tuco, to open, safe\}, we could randomly get \(rusty\) for safe, \(outlaw\) for Tuco and \(quickly\) for the verb, which would end as: “Tuco the outlaw quickly opens the rusty safe.”. This was also a way to avoid having two adverbs for the same verb, which gave an odd feeling to the sentence.

In addition to this set of algorithms is an organisation algorithm, whose job is to trigger the alternative eight algorithms. Besides generating sentences, each algorithm adds an emotional tag, so that the narration is more expressive when it is be used by a virtual agent. The Stand alone and Social relationship sentences are supposed to be neutral, the suspense being either hope or fear depending on the average valence of the set of scripted events for the narrator. For these scripted events, the emotional tag is the emotion felt by the narrator as if he or she was an observer.

IV. Results

A. Example

We have used multiple scenarios to test our hypothesis. The main one is a Western, and focuses on Clint the sheriff and Tuco the outlaw. It begins with a bank robbery by Tuco,
and ends with a duel between Tuco and Clint. We present here an example of generated sentences, making explicit each algorithm’s work before showing the final result. Each algorithm is not always called each time for each date, but we chose an example where it was the case, in order to demonstrate them as a whole.

In our example, as input we have two triplets: {Tuco, to shoot, Clint} and {Clint, to shoot, Tuco}. Here is the sentences obtained after each subalgorithm:

- **Stand alone**; Clint thinks about himself.
- **Social relationship**: Tuco frowns at Clint.
- **Suspense**: Clint watches his gun.
- **Events**: Clint shoots Tuco. Tuco shoots Clint.
- **Narrator POV**: I wish I could have done something to help!
- **Filling and Organisation**: Clint peacefully thinks about himself near the saloon. Tuco frowns at Clint the sheriff. Clint watches his old gun. Then violent Clint and mean Tuco shoot each other. I wish I could have done something to help!

### B. Scenarios

We have just seen a Western scenario. Another one is called CopThief, and is depicted in another article [13]. This scenario was made especially to get an impression of the possibilities of the evolution of emotions and social relationships between characters.

As we saw, Western is our principal scenario. Alongside Tuco and Clint, we have Maria the deputy sheriff, an undertaker, a priest, and bank counter clerk. All these characters are void at first, but we use all the possibilities of OSSE and of our ontology to deepen their personae. For instance, the clerk is strange, Tuco is violent and Clint impulsive. We can also give them roles: sheriff, deputy sheriff, outlaw, local, strangers... We can parameter their preferences (for instance Tuco likes the pistol concept, whereas usually people are afraid of them), and their starting social relationships (Tuco and Clint don’t like each other at all, whereas Clint and Maria have a strong solidarity feeling). We made a specific library for the Western atmosphere, which we add to the standard library. In that library we included concepts such as to shoot, saloon, sheriff, to duel, gun.

In order to attain a clearer view of what the result looks like, here is a more detailed example:

**Before**: Tuco threatens Bank Man. Tuco could attack Bank Man. Bank Man gives the money. Tuco gets it. Tuco leaves the bank. Cherokee wounds Maria. Old Cherokee watches distressed Maria. Afterwards, annoying Cherokee wounds Maria the deputy sheriff. He shouldn’t have done that, don’t you agree?

**After**: Old Bank Man frowns at Tuco. Tuco plays with the gun and thinks about Bank Man. And after that, Tuco threatens and could slowly attack old Bank Man. I’m sure he would like to be as small as a mouse! Strange Bank Man frowns at Tuco who loves helping. Tuco thinks about the money and Bank Man plays with it. Next, strange Bank Man gives the money and Tuco gets it. Look how proud he is! Afterwards, Tuco the stranger leaves the bank. Show-off Maria the deputy sheriff dreads annoying Cherokee the stranger.

### V. Evaluation

In order to test our algorithms, we created an online interface. After a quick introduction and explanation of the test, participants are shown the two scenarios (Western and CopThief) in random order. For the Western scenario, we show them two stories in random order, one without algorithm, and the other with one algorithm among: Suspense alone, Social Relationships alone, Narrator POV alone, Filling alone and all algorithms. For the CopThief scenario, we show them two stories in random order, one without algorithm, and one with all algorithms.

After reading the two versions of each scenario, the subject is being asked what he thought of the two stories. They are evaluated based on a number of different criteria: Surprising, Amusing, Boring, Repetitive, Captivating, Disturbing. In addition to these, both stories are compared, and the subject is being asked which one she/he preferred, and whether he had the feeling of reading twice the same story. These criteria ranged from 1 to 7, one being labelled “Not at all” and seven “Completely”. For the preference criterion, the more it is close to 7, the more one preferre the story with specific algorithm.

Once the subject finishes judging both scenarios, we ask him a few questions to gather metadata. We ask him/her his gender, how old he is, his english level, how familiar he is with video games, how many books he reads in a month, and how often he goes to the movies. This data was meant to check if our panel was sufficiently fluent in english, and if the population was consistent.

### A. Results

The study involved 32 people reading our stories. The results are shown in the Western scenario’s table (Fig 2) and CopThief scenario’s one (Fig 3). Each row corresponds to a specific scenario applied to a story, and each column to a criterion. For the column “Same” and “Pref” the value corresponds to the comparison with the “none algorithm” version of the story. Each cell contains the average evaluation value, and in parentheses are comparisons between with and without algorithms.

In average the algorithms had an improving effect, would that be improving the surprising, amusing, captivating aspects, or to lower the feeling of boredom, repetitiveness and disturbance. Furthermore, we see that some algorithms have quite a specific effect, like the Narrator POV that have a strong impact on amusement. We see too that Suspense lowers boredom, which was indeed its principal aim, by stimulating the brain with tension. Filling is the algorithm that lowers the disturbance the most by adding variety together with Narrator POV, which implies that in order to lower this aspect, you have to make the sentences richer, and to give it a perspective, a context. Strangely, the Social Relationships added to the disturbance, possibly because in our architecture, the algorithm...
added a sentence describing the relation between two random characters, the point of which might not always have been clear to the reader. We also note that by mixing the algorithms, they lose their specific strength, but keep on adding on the amusing level. Last, the subjects found the two kinds of stories quite different, and tend to prefer the ones with generated meaning added.

Most commentaries about the disturbing aspect were that natural language generation was lacking in some aspect, for instance the fact that the nouns of the characters are always repeated. Another subject e-mailed us that she felt that “more sentences meant it was more of a story and less of an event” which is clearly the direction we wanted to take. Beside, there weren’t any commentaries on the consistency of the story. That and the fact that in average the altered stories were seen as less disturbing, let us suppose that consistency issues were mostly due to the natural language generation and not the roughness resulting from the scanning algorithms.

Our results are encouraging, and prove that randomly generated context from a representation of the world can enhance user experience. The next step will be to tweak the algorithms parameter in order to maximize their effect, and create new algorithms in order to lower the repetitive factor by adding variety.

VI. Conclusion

In this article we have suggested and presented means of enriching a story with real time generated sentences using an ontology, its relations, and scanning and realisation algorithms. They are meant to give a more realistic aspect, less jerky and hence more human-like on a semantic scale. The creation of suspense, and its way of varying tension throughout the story helps the narrator build a stimulating story. Modelling and expressing social relationships contributes towards deepening the emotional effect of the story, and anchors gamers/listeners/viewers/readers in the story as they become more affected by what happens to characters they learn more about them. Lastly, letting the narrator express his or her point of view on the events is a way of humanizing the narrator, trying to move closer to a more classical interaction. Our current evaluation will be a way to test if our hypothesis are true, and to correct our research direction.

The technologies mentioned here are meant to enrich the story, to make it more emotional, which is important for storytelling generally but also on a broader scale [16]. This is why our model, and our algorithms, can be used not only for narration, but also in other situations in which an agent has to communicate and only has an outline of his or her speech (scenario, argumentative speech, predicate based architecture...).

Perspectives: Our aim was not to develop a perfect natural language realisation, but more to create higher level algorithms that play with the tension and the emotional state of a story, and to generate enough realisations to test our hypothesis in an online evaluation. In addition, we wanted neither to create a bigger and better ontology nor to create an exhaustive list of possible relations, but to prove that with a simple knowledge representation and process, we can already produce promising results, which may bode well for a larger and more complex structure.

This project was made possible thanks to the LTCI and the LIP6, and the final aim is to use GRETA [10], an emotional virtual agent from the LTCI, as a narrative agent, using the OSSE model from the LIP6. The next step is to connect OSSE and GRETA so that she can speak the sentences, and use the emotional tags we added to the text to show facial expressions linked to the triggered emotions. This will be a new way to test our hypothesis on more gestual and expressive modality [6].

We also plan to add goals in the scenarios’ descriptions, which is coherent with the OCC [4] model used in OSSE, so that the ontology and the algorithms know what the objectives of each character are, and build more accurate sentences. This way they will be able to construct logical sentences before a goal’s realisation and increase the interest of a story by diluting the dense scenario skeleton.

Furthermore, a good way to make the ontology even more dynamic would be to let the event change the relations between concepts, creating a more reactive environment. This issue is made particularly obvious when a dead character
reappeared in our story after he was buried! We also plan
on creating a graphical user interface in order to make it
easier for game designers to use, paving the way for more
complete libraries. We believe that this could become a tool
for game designers to create more complex scenarios and to
model complex interactions between Non-Player Characters
and players, and we wish to create a community around that
software. Indeed, depending on the atmosphere, the game
designer would not have to rebuild his or her ontology from
scratch, but would need to acquire access to a specific library,
or at least to draw inspiration from a specific one (film noir,
sci-fi, Victorian era...) created by a community of researchers
and game designers.

REFERENCES

perspective. *Working notes of the Narrative Intelligence Symposium*,
1999.
characters. *Proceedings of the Workshop on Virtual Conversational
University of Texas Press, 2010.
evidence on the role of unexpectedness. *Proceedings of the 31st Annual
Greta: Towards an interactive conversational virtual companion. *Arti-
ficial Companions in Society: Perspectives on the Present and Future*,
2009.
for Knowledge Sharing in Formal Ontology in Conceptual Analysis and
expressions of emotions. *First International Conference on Affective
[13] N. Sabouret, M. Ochs and V. Corruble. Simulation of the dynamics of
non-player characters emotions and social relations in games. *Transactions on Computational Intelligence and AI in Games*, 2009.
[15] H. Shrobe, R. Davis and P. Szolovits. What is a knowledge representa-
*Affective interactions*, 2000.
[18] J. F. Sowa. Conceptual structures : information processing in mind and