De la description de scénarios socio-émotionnels au narrateur virtuel expressif

R. Miletitch† N. Sabouret† M. Ochs‡
roman.miletitch@gmail.com Nicolas.Sabouret@lip6.fr magalie.ochs@gmail.com

†Laboratoire d'informatique de Paris 6
Université Pierre et Marie Curie
4 place Jussieu 75005 Paris, France

‡CNRS LTCI
Telecom-ParisTech
37 rue Dareau 75014 Paris, France

Résumé :
Raconter une histoire nécessite de relier entre eux un ensemble d'événements significatifs à l'aide d'énoncés permettant de maintenir la tension, de créer du suspense et de laisser du temps au développement des émotions. Lors de l'automatisation de ce processus, l'une des difficultés majeures est la génération de ces phrases brodées et le fait de leur donner suffisamment de cohérence avec l'histoire. Pour ce faire, nous proposons dans cet article d'utiliser un modèle de représentation de connaissances locales et cohérentes du monde. Nous présenterons une architecture sous forme d'ontologie de ces concepts. Mis en relations entre eux, ils permettront de décrire ce monde, et des algorithmes parcourront l'ontologie pour en extirper du sens de manière pseudo aléatoire, permettant la création de phrase. Ces différents éléments seront mis en commun lors du traitement d'une histoire prédéfinie. Lors du parcours de son squelette imposé, les phrases issues de nos algorithmes (Suspense, Relation sociale, Phrase seule, Événement, Phrase du Narrateur), une fois mises en forme, viendront enrichir l'histoire.

Mots-clés : Narration, Ontologie, Agent Conversationel Animé, Emotions

Abstract:
Telling a story requires linking a series of significant events using statements to maintain tension, create suspense and to allow time for the development of emotions. When automating this process, one major challenge is the generation of these filling sentences and them being 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 will present an architecture in the form of an ontology of these concepts. By establishing relations between them, they will describe this world, and our algorithms will scan the ontology to drag sense out of it in a pseudo-random manner, allowing the creation of a sentence. These elements will be gathered during the process of a predefined storyline. During the scan of the scenario, an imposed skeleton, sentences from our algorithms (Suspense, Social Relations, one phrase, Event, Phrase Narrator), once edited, will enrich the story.

Keywords: Narration, Ontology, Animated Conversational Agent, Emotions

1 Introduction

Pour raconter une histoire, le narrateur ne doit pas seulement relater des évènements, mais aussi faire ressentir des émotions tout au long de la narration. Le fait de "broder" autour des évènements, d'accaparer l'attention de l'interlocuteur, de stimuler sa visualisation et ses sensations, tout ceci amène à créer une histoire émouvante, intéressante et donc mémorable[1].

Il est important de garde à l'idée que lorsque l'on raconte une histoire, il ne s'agit pas juste de communication vocale, mais d'une communication à plus haute échelle, d'une création d'une ambiance par les sens, ou tous les événements racontés sont mise en valeur les uns par rapport aux autres. Au-delà de ces sensations, ce qui motive le lecteur à poursuivre sa lecture, c'est l'intérêt que suscite l'histoire, et la tension qu'elle crée, à travers une répartition de cet intérêt[4].

En plus de ces sensations et de cet intérêt, une autre manière d'ancrer le public dans la scène du narrateur est de stimuler des émotions[2], et c'est là que le don d'empathie devient prépondérant. Nous pensons que le narrateur ne doit pas avoir un regard objectif sur la situation, mais doit plutôt exacerber ses émotions lors du conte pour en rendre la lecture intéressante. Signaler les émotions des personnages n'est pas suffisant, il est mieux pour le narrateur de les exprimer, de les manifester, à travers un choix de mot, à travers une gestualité ainsi que des expressions faciales.

Dans la section 2, nous allons présenter les travaux existant en "StoryTelling", ainsi qu'approfondir certaines thématiques abordées dans cette introduction. Ensuite dans la section 3, nous vous présenterons notre modèle, et les algorithmes mis en oeuvre. Enfin dans la section 4, nous exposerons les résultats obtenus.

2 Travaux existants

The virtual Story Teller [8] est un framework multi-agents qui permet la création dynamique

d'une histoire. Ses agents, guidés par leurs objectifs et contrôlées par un agent réalisateur construisent un scénario. Ce framework nous montre que l'aspect indéterminé de la création d'une histoire joue en sa faveur, effet que l'on retrouvera dans notre modèle. Deux agents s'occupent de traduire le scénario en histoire : le narrateur, et le présentateur. Cette séparation se retrouve dans nombre de systèmes, et c'est ce qui nous a fait opter pour deux classes d'algorithmes : une pour la construction de ces phrases, et l'autre pour leurs réalisations superficielles.

Cavazza traite dans un de ses articles [3] de l'intégration d'un générateur de langue naturelle au sein d'un planificateur émotionnel en s'inspirant de l'univers de Flaubert. Cette recherche tire aussi profit du fait que Flaubert avait l'habitude de créer ce que nous appellerions aujourd'hui une ontologie pour ses livres. Nous retrouvons ici l'intérêt pour les émotions qui serviront de base à notre modèle ainsi que l'utilisation d'une ontologie.

Jean Louis Dessales propose une modélisation de l'intérêt [4] issu de la communication d'événements, qui s'applique tout particulièrement au StoryTelling. Il définit l'intérêt d'un évènement par la différence entre sa complexité d'occurrence, et sa complexité à être raconté. Au-delà de cela, il caractérise l'intérêt par l'aspect improbable d'un évènement. Il suscite alors la curiosité, une certaine forme de suspense quant à sa raison d'être. C'est ce suspense qui sera la source d'un de nos algorithmes de création d'intérêt. Enfin, il indique que l'intérêt d'un événement est proportionnel à l'intensité émotionnelle contrastante qu'il suscite[5], peu importe le type d'émotion. Nos algorithmes se basent sur cette théorie pour savoir quelles émotions sont à développer dans notre scénario, et à quels moments.

Modèle OSSE. Notre travail a eu comme source un logiciel qui se fonde sur une modélisation dynamique de l'état socioémotionnel : OSSE [7]. Cette modélisation se concentre sur les facteurs de personnalité qui favorisent le déclenchement d'émotions, ainsi que l'influence que ces émotions ont sur le comportement et sur l'évolution des relations sociales entre les personnages mis en jeu. Le vocabulaire de OSSE est composé de 4 ensembles : $Voc_{OSSE} = O \cup V \cup R \cup P$, où O est l'ensemble des objet ou game characters, V l'ensemble des actions (verbes), R l'ensemble des rôles possibles pour les personnages, et P l'ensemble des personnages. Ces rôles sont liés par des relations sociales : $RelSoc : R \times R \rightarrow \{Agreeability, Dominance, Solidarity, Familiarity\}$. Les personnages sont définis comme un set : $\{Name_p, Personality_p, Emotions_p, Roles_p, Attitudes_p\}$. Ce vocabulaire nous permet de définir des évènements $e \in Evt$ par $e = \{Agent_e \in P, Verbe_e \in A, Patient_e \in O \cup P, date_e \in N, WhoWatching_e \subset P\}$. L'$Agent_e$ définit qui agit dans l'action, $Verbe_e$ définit le verbe, $Patient_e$ celui qui subit l'action. La $date_e$ permet d'organiser la liste des évènements. $WhoWatching$ correspond à la liste des personnages de P qui observent l'action. Un scénario est un ensemble d'évènements. Lors de l'avènement d'un triplet, l'état émotionnel ainsi que les relations sociales des observateurs de cet événement sont modifiés.

3 Modèle proposé

Comme dans le modèle OSSE, nous nous attachons à conserver une description du scénario à l'aide de triplets. Cette représentation, qui présente l'avantage d'être utilisable par des "game designers " non spécialistes de l'IA, pose le problème de la génération automatique de textes en langue naturelle par le narrateur virtuel expressif. Notre objectif est donc, en partant de cette suite d'événements, et à partir d'un vocabulaire restreint, d'enrichir une histoire de phrases cohérentes renforçant l'intérêt, l'impact émotionnel et la crédibilité de l'histoire chez les spectateurs sans pour autant la modifier fondamentalement.

Nous nous appuyons sur deux hypothèses :
– une modélisation simple et locale du monde mise en relation avec elle-même est suffisante pour renforcer l'intérêt, les sensations et les émotions
– la génération pseudo aléatoire d'éléments peut-être non pertinents ne gêne pas la narration. Au contraire, cela l'enrichit, car le lecteur/spectateur construira les liens absents.

3.1 Architecture globale

De la création d'événement au conte raconté, nous utiliserons deux modules. OSSE qui s'occupe de la modélisation socioémotionnelle et notre module composé d'une ontologie et d'algorithmes de parcours qui permettent d'en faire émerger du sens [11], et de créer la phrase.

Ontologie. Une ontologie est la spécification d'une conceptualisation d'un domaine de connaissance [6]. Elle constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Notre ontologie s'inspire de Cavazza [3], ainsi que des Graphes conceptuels de Sowa [10]. L'ontologie initiale définit un nouveau vocabulaire, en parallèle de celui d'OSSE, composée de concept et de relation entre ses concepts : $Voc_{Onto} = Con \cup Rel$, où chaque concept d'OSSE a son équivalent. Les concepts sont organisés de manière hiérarchique, et même si leur organisation est modifiable par le game designer, il en existe une de base, imposée, afin de guider le game designer. En plus des concepts, notre ontologie sera décrite par des relations r définies par $r : Con \times Con \rightarrow Rel$ permettant la mise en relation de deux concepts, par un lien d'un certain type de catégorie. L'utilisation d'une représentation de connaissance avec héritage permet de définir les propriétés et les relations entre concepts à un haut niveau, les sous-concepts héritant automatiquement des qualificatifs de leurs parents.

Le fait d'utiliser une ontologie plutôt que de scripter notre histoire nous permet d'avoir une mise en scène variée. Tout au long de notre démarche de création de langue, nous essayerons de garder le plus longtemps et le plus souvent possible cette idée de concepts et de lien entre concepts. Il est préférable d'avoir une liste de concept ordonnée plutôt qu'une chaine de caractère ; une fois générée, elle ne nous serait en effet alors impossible de lui rajouter du sens, de jouer sur sa syntaxe [9]. Elle est un résultat fini, qui empêche tout rajout ultérieur. Voilà pourquoi nous gardons une organisation de phrase à partir d'item : $item = \{concept + typeGramatical + qualificatifs\}$.

Algorithme de génération. L'ontologie permet de décrire le monde, de le structurer. L'ordre est un moyen d'accès au sens, mais c'est le traitement de cette information qui permet de le faire émerger. Il s'agit là du travail des algorithmes de parcours que nous avons créé en nous inspirant des théories sur l'impact émotionnel et la modélisation de l'intérêt de Jean Louis Dessales [4]. Ces algorithmes sont au nombre de huit et chacun permet d'avoir un contrôle accru sur la tension des spectateurs tout au long de l'histoire. Deux sont particuliers : il s'agit des algorithmes d'introduction et de conclusion, afin de créer une ambiance au début, ainsi qu'un épilogue, qui permet un relâchement de la tension. Cinq autres sont appelés à chaque date, et à partir de l'ensemble des triplets des événements de cette date, génère des liaisons entre ces triplets et l'ontologie :

- *Réalisation évènement* traduit l'ensemble des évènements d'une $date_t$, correspond à la structure de l'histoire créée par le game designer
- *Suspense* crée une phrase en amont d'accroche afin de créer une tension
- *Relation sociale* s'inspire des relations entre personnages renforçant le coté émotionnel de l'histoire
- *Stand Alone* crée des phrases seules, sans impact fort, afin de jouer sur la tension
- *Phrase narrateur* laisse l'occasion au narrateur de s'exprimer par commentaire sur les évènements de la $date_t$, renforçant l'attention du spectateur, ainsi que le coté émotionel
- *Filling* s'occupe de trouver des qualificatifs à rajouter aux items, pour varier l'expérience du spectateur

Ces algorithmes sont à la base de la tension, et de l'émotion générées dans notre histoire. Il est possible de modifier le paramètre narrateur, ce qui permet de modifier les histoires. En effet, l'histoire ne sera pas racontée de la même manière selon si l'on est le voleur qui se fait arrêté ou le policier qui effectue l'arrestation, par exemple. Enfin, un dernier algorithme, se chargeant de l'organisation, gère l'occurrence de ces algorithmes. Permettre à chacun de ces algorithmes de s'exprimer en permanence surchargerait l'histoire. L'algorithme d'organisation associe à chaque algorithme de génération une probabilité d'occurrence, et les appelle en fonction de cette dernière.

Algorithme de réalisation. Une fois les liaisons faites entre la liste des événements et les éléments de l'ontologie, il reste à réaliser une phrase de ces relations. Ce sont les algorithmes de réalisation superficielle qui transcrivent le sens extirpé par les algorithmes de parcours depuis l'ontologie en une phrase compréhensible par l'être humain et qui ajoutent des tags d'émotion et de comportement qui guideront l'interprétation du texte pour des agents conversationnels. L'objectif de ces deux catégories d'algorithme est d'interdire le plus possible des bouts de phrases écrits par le codeur et de favoriser une génération automatique de phrase de par l'agencement des mots et leurs liaisons. Dans les cas dont l'occurrence est unique, nous nous permettons d'en revenir à l'utilisation de phrase à trous (par exemple pour l'introduction), qui est alors remplie selon les éléments mis en jeux, les

concepts, ou encore les émotions du narrateur afin de lui donner un comportement plus proche de ce qui pourrait être attendu d'un narrateur humain.

Organisation grammaticale de la phrase. Nous avons décidé de séparer la phrase en item : $item \in I = \{concept \in Con, type_{gram} \in \{Sujet, Verbe, COD\}, qualificatifs\}$ où $qualificatifs$ correspond à une liste de description de $concept$ définissant le mot, et $type_{gram}$ sa position dans la phrase. Une phrase est donc définie comme une liste de mots : $sentence = \{item_0, .., item_n\}$, et un paragraphe comme une liste de phrases : $paragraph = \{sentence_1, .., sentence_{n'}\}$.

Nous partons des phrases simples des événements, qui forment déjà un paragraphe, auquel nous ajoutons la phrase simple de suspense, celle des relations sociales, celle Stand Alone, et lors de la réalisation de ces phrases, nous ajoutons la phrase du narrateur. À la suite de quoi, nous modifions leur mise en forme et nous ajoutons les qualificatifs.

3.2 Ontologie

Afin de permettre une manipulation plus aisée, notre ontologie se base sur une hiérarchie (FIG 1). Elle se remplit à l'aide de bibliothèque ou grâce à l'imagination de l'auteur qui permet ainsi de développer une hiérarchie plus complexe. Il ne s'agit pas juste de la remplir de connaissances, mais aussi de l'architecturer. Par exemple, nous décidons dans notre implémentation de séparer objets en gros objets, petits objets, et objets liquides. Ce choix est lié au choix de nos algorithmes et aux relations mises en jeu. Une fois cette architecture créée, l'auteur peut créer des liquides, définir des relations qui leur sont propres, mais aussi d'autres classes. De plus, les classes ou instances créées peuvent hériter de plusieurs concepts Parent. Il est possible d'avoir un gros objet récipient. Toute cette infrastructure permet à l'auteur de créer du sens, en reliant les classes entre elles par des relations. Par exemple, si nous permettons aux instances de Être de parler, alors tous nos personnages auront la possibilité de parler, et les algorithmes de génération en seront automatiquement informés.

Enfin, il existe une dernière distinction entre les concepts de l'ontologie. On considère par défaut que si le concept est une feuille, il s'agit d'une

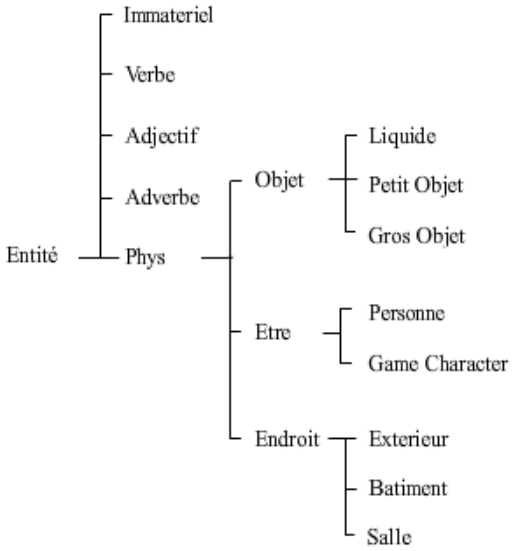

FIGURE 1 – Architecture de base de l'ontologie

instance, et que s'il s'agit d'un parent (hors Absurdité comme fils), alors on préférera l'imaginer comme une classe. Imaginons le concept *clef* héritant de *petit objet* et le concept *clef de la porte* héritant de *clef*. Le concept *clef* est alors parent, et le concept *clef de la porte* est alors une feuille. Lorsque nos algorithmes appelleront *clef*, ils auront tendance à écrire "une clef", mais lorsqu'ils appelleront *clef de la porte*, ils écriront "la clef de la porte" en y faisant référence directement puisqu'il s'agit d'une instance.

Les différentes relations. Pour que les algorithmes puissent faire émerger un sens de ces concepts organisés, il faut les mettre en rapport, et pour cela nous avons besoin des relations évoquées précédemment. Dans notre cas, ces relations sont binaires, et directionnelles. Une relation est définie par son type, et les concepts qu'elle relie (limitée à certains concepts, selon le type de la relation). Il existe de nombreux types : les types de placement qui permettent au designer de situer sa scène, et à cette description d'émerger dans l'histoire ; les types grammaticaux, définissant quel sujet et COD peuvent aller avec quels verbes ; les types qualificatifs (adjectif, adverbe) ; et les types plus sémantiques, comme Obj_Vrb liant un verbe avec un objet qui lui est souvent associé, comme $manger$ et $fourchette$.

Notre but ici n'est pas de créer une liste exhaustive de type de relations, mais plus une architecture simple et pertinente, et de laisser libre cours à la création de différents types de relations dans le futur. De plus, les relations que nous avons

créées ici sont intimement liées au choix des algorithmes de génération de sens. Lors de la création de nouveau type de relations, il faudra les inclure dans la génération de sens de précédents algorithmes voir aussi de créer de nouveaux algorithmes.

En représentant notre monde par ses concepts, et en le décrivant par des relations, nous avons accès à un sens général. Ce sens permet de créer une masse de relation entre les concepts, décrivant plusieurs possibilités de notre monde, par exemple : "Le chapeau bleu est sur la commode" ou encore "Tuco a un grand chapeau". Nos algorithmes créeront à la volée ce sens par association, et/ou choisiront parmi les associations préexistantes afin de créer un sens cohérent avec le scénario.

3.3 Algorithmes de Parcours

Deux des algorithmes de parcours sont particuliers : il s'agit des algorithmes d'introduction et de conclusion, afin de créer une ambiance au début, une attente, ainsi qu'un épilogue, qui permet un relâchement de la tension. Cinq autres s'occupent de gérer la tension au sein de l'histoire. Le premier de ces cinq : Réalisation événement, le plus simple, ne s'occupe que de récupérer la liste des événements, et de la traduire en une grammaire compréhensible par les algorithmes suivants. Il correspond au climax, point culminant du scénario, où la tension doit être au plus forte, avant de diminuer. L'algorithme Suspense part directement de la liste des événements, et essaye de construire une autre phrase qui va précéder les événements afin de construire une attente chez le spectateur, permettant de faire émerger une tension à travers l'attente de l'évènement. Son fonctionnement est explicité dans l'algorithme 1.

Algorithm 1 Algorithme Suspense

Require: $paragraph$
1: choose at random $stc = sentence$ in $paragraph$
2: choose at random $cod' \in O \cup P$ so that $cod' \rightarrow Obj\_Vrb \rightarrow stc.Verbe$ or $cod' = stc.COD$
3: choose at random $Verbe' \in V$ so that $cod' \rightarrow Obj\_Vrb \rightarrow Verbe'$
4: create sentence : $Sente = \{stc.Sujet, Verbe', cod'\}$
5: paragraph.addSentence($Sente$)

Prenons un exemple, avec comme triplet issu de notre paragraphe : {Tuco, to open, safe}. Nous devons instancier un nouveau COD, qui correspond soit au patient, soit à un concept lié au verbe par une relation Obj_Vrb ce qui dans notre cas donne : $safe$ ou $\{key, lockpick\}$. Imaginons que l'on prenne key, nous en retirons une liste de verbe potentiel (grâce à la même relation) qui est *{to use, to watch, to play with}*, dont nous tirons *to play with*. En conservant le même agent, nous obtenons le triplet {Tuco, to play, key}, ce qui donne comme réalisation de phrase : "Tuco plays with the key. Tuco open the safe.". On voit ici que le suspense est créé par une action triviale sur un objet qui est lourd en sens, et qui sera en effet utilisé lors de la phrase suivante (si ce n'est au niveau de l'écrit, du moins au niveau sémantique).

L'algorithme 2 Relation Sociale quant à lui part des personnages présents dans la scène, favorisant les acteurs, et selon leurs valeurs d'appréciation, de dominance, de familiarité et de solidarité génère une phrase. Plus que de décrire ces relations, nous avons voulu ici faire agir les personnages selon leur relation de manière simple et n'influençant pas le scénario (sourire, froncer les sourcils...).

Algorithm 2 Algorithme Relation Sociale

Require: $paragraph$
1: list all characters : $C = \{c \in P, c \in paragraph\}$
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** $SocRel.char > max$ **then**
6: addSentence($SocRel.char$ is high)
7: **else if** $SocRel.char < min$ **then**
8: addSentence($SocRel.char$ is low)
9: **end if**

Par exemple, imaginons un paragraphe formé des triplets {Tuco, to open, safe}, {Maria, to look, Tuco} et {Bank Man, to think, money}. La liste des personnages est $\{Tuco, Maria, BankMan\}$. Supposons que le hasard nous donne Maria et Tuco, nous nous intéressons donc à leur relation. Nous tirons au hasard une caractéristique parmi les quatre énoncées plus haut : $\{Agreeability, Dominance, Solidarity, Familiarity\}$. Une fois dominance choisie, il ne nous reste plus qu'à comparer sa valeur à un seuil minimal et maximal, qui différent selon la caractéristique de la relation sociale

sur laquelle on se fonde. Dans notre cas, la dominance sera de 0.7 sur 1 dépassant le seuil de 0.6, en faveur de Maria. Nous créerons donc un triplet préassemblé, spécifique à ce seuil et cette caractéristique. Dans notre cas, cela sera le triplet : {Maria, to despise, Tuco}.

À ces algorithmes s'ajoute Stand-Alone, phrase seule, sans relation avec les événements précédents ou futurs, permettant de calmer le rythme, souvent trop rapide, car chaque événement OSSE est voulu comme ayant un fort impact émotionnel. Enfin, le dernier des cinq, Phrase Narrateur, permet à la fois au narrateur de transparaitre, et de manifester ses émotions. Nous avons vu précédemment que l'importance du conteur dans le conte ne doit pas être négligée, voilà pourquoi nous avons voulu inclure cet algorithme. Dans ce cas, le narrateur émet un commentaire subjectif sur l'évènement qui vient de se passer, ce qui permet aussi de relâcher la tension après le climax.

Enfin, le dernier des huit algorithmes est un cas à part, il s'agit du Filling (Remplissage). Il parcourt les phrases créées précédemment, et cherche toutes opportunités dans l'ontologie, dans ses relations, et dans les propriétés d'OSSE pour ajouter des qualificatifs, tels que les adjectifs, des relations de positionnement, des traits de personnalité... Ces qualificatifs permettent de plonger l'histoire dans le réel. Ils donnent un historique aux objets qualifiés, et le spectateur ira a la recherche de cette histoire inventée à la volée. Les qualificatifs se rajoutent de manière aléatoire aux personnages, objets ou verbe, selon l'algorithme 3.

Algorithm 3 Algorithme Filling

Require: $paragraph$
1: $\forall sentence \in paragraph$
2: choose at random $k \in [0, 1000]$
3: **if** $k > 0.35$ **then**
4: Fill(sentence.getPersonne)
5: execute Algorithme Filling again
6: **else if** $k > 0.50$ **then**
7: Fill(sentence.getPersonne)
8: **else if** $k > 0.66$ **then**
9: Fill(sentence.getObjet)
10: **else if** $k > 0.83$ **then**
11: Fill(sentence.Verbe)
12: **end if**

Par exemple, en partant du triplet {Tuco, to open, safe}, nous pourrions obtenir comme qualificatif $rusty$ sur $safe$, $outlaw$ sur Tuco et $quickly$ sur le verbe. Ce qui donnerait : "Tuco the outlaw quickly opens the rusty safe.".

À cette suite d'algorithmes s'ajoute l'algorithme d'organisation qui gère l'appel des huit algorithmes que nous venons de décrire.

En plus de créer ces phrases, chaque algorithme ajoute un tag émotionnel, afin de permettre une narration plus chargée en émotion. Les phrases Stand Alone et relation sociales sont considérées comme neutres et le suspense est lié à l'émotion de peur ou d'espérance selon l'impact positif ou négatif des évènements sur le narrateur. Pour ces derniers, le tag est défini par l'émotion perçue par le narrateur. Enfin, pour la phrase de narrateur, le tag correspond à une émotion dépendant de l'affection du narrateur pour le personnage en question, et de l'émotion que ce dernier ressent.

4 Résultats

4.1 Exemple

Nous avons utilisé plusieurs scénarios pour tester nos hypothèses. Le principal portait sur la thématique du Western, et se focalisait sur Clint le shérif, et Tuco le hors-la-loi. Le scénario débute par l'attaque de la banque par Tuco, et se finit par un duel. Nous allons ici montrer un exemple de phrase générée, en explicitant la part de chaque algorithme avant de montrer le résultat final. Chaque algorithme n'est pas automatiquement appelé lors d'une suite d'événements, mais nous avons choisi un exemple où c'est le cas afin de tous les montrer.

En entrée, nous avons les deux triplets suivants : {Tuco, to shoot, Clint} et {Clint, to shoot, Tuco}. Voici l'exemple disséqué de réalisation que notre programme a généré.

- Stand Alone : Clint thinks about himself.
- Relation Sociale : Tuco frowns to Clint.
- Suspense : Clint watches his gun.
- Evènement : Clint shoots Tuco. Tuco shoots Clint.
- Phrase narrateur : I wish I could have done something to help !
- Filling et Organisation : Clint peacefuly thinks about himself near the saloon. Tuco frowns to 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 !

4.2 Scenarios

Les deux scénarios que nous avons choisi de créer pour tester nos algorithmes sont *Western* et *Policier*. Le premier a été déjà évoqué ; le second, traite d'un interrogatoire en cinq évènements entre un policier et un voleur. Il a été présenté en détail dans un autre article [7]. Ces évènements ont été créés à titre d'exemple d'évolution de relation sociale entre deux personnages. Western est notre scénario principal. En plus de Tuco et Clint, nous avons Maria, la shérif adjointe, un indien, un croque-mort, un prêtre et un guichetier de banque. Tous ces personnages possèdent une personnalité propre, définie par des mots clefs dans OSSE. Par exemple, le banquier est *dull* (ennuyeux), Tuco est *violent* et Clint *gentle* (doux). Comme nous l'avons vu dans le modèle OSSE, ils possèdent aussi des rôles : shérif, shérif adjoint, hors-la-loi, local ou stranger... Enfin, nous définissons leurs préférences (par exemple : Tuco a une vision moins négative du pistolet que les autres par exemple) ainsi que leurs relations sociales de départ (par exemple : Tuco et Clint ne s'aiment pas, Clint et Maria ont un fort sentiment de solidarité). Pour l'ontologie, une bibliothèque de concept spécifique à l'ambiance Western est rajoutée aux concepts standards définissant une bibliothèque standard minimale. Parmi ces concepts on retrouve *to shoot*, *saloon*, *sheriff*, *to duel*, *gun*.

Pour mieux voir l'effet de nos algorithmes, voici leur résultat sur un texte plus conséquent :
Avant : Tuco threatens Bank Man. Tuco could attack Bank Man. Bank Man gives the money. Tuco gets it. Tuco leaves the bank. Cherokee wounds Maria.
Après : Old Bank Man frowns to 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 to 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. Old Cherokee watchs distressed Maria. Afterwards, annoying Cherokee wounds Maria the deputy sheriff. He shouldn't have done that, don't you agree ?

Le texte ainsi généré donne un aperçu des résultats que l'on peut attendre. Il nous reste à affiner la probabilité d'occurrence de chacun des algorithmes ainsi que la variété des relations avant notre prochaine évaluation.

Planification de l'évaluation. Nous sommes en train d'organiser une évaluation pour tester l'influence de nos algorithmes sur les histoires racontées. Nous voulons comparer *sans algorithme* à *Suspense* seul, *Relation Sociale* seul, *Phrase du Narrateur* seul, *Filling* seul, ou *avec tous les algorithmes*. Cela nous permettra de voir l'impact global ainsi que celui de chacun des algorithmes majeurs. Les critères de jugements sont : *Surprising*, *Amusing*, *Boring*, *Repetitive*, *Captivating*, *Disturbing*. En plus de ces six critères, nous aurons des critères qui permettront de comparer les histoires entre elles : *Same* comparera la similarité, et *Preference* indiquera la préférence d'une histoire sur l'autre.

5 Conclusion

Nous venons de voir comment, à partir d'une ontologie, de ses relations, et d'algorithmes de parcours et de réalisation nous pouvons grossir le squelette d'une histoire, afin de lui donner une apparence que nous espérons plus réaliste, moins saccadée, et donc plus humaine. La création de suspense, sans pour autant créer une réelle intensité (qui est de la responsabilité du scénariste) lors de l'évolution de l'histoire, permet de jouer sur la tension le long de la narration. Le fait de manifester les relations sociales et leur évolution renforce l'immersion des spectateurs, ils se sentent plus concernés par ces personnages et par les péripéties qu'ils rencontrent. Enfin, les phrases du narrateur (pour ne parler que des principaux algorithmes) permettent d'humaniser le narrateur, et donnent une tout autre dimension à l'histoire. Notre future évaluation nous permettra de mettre à l'épreuve nos idées, et de nous guider dans notre recherche.

Les technologies employées ici permettent de rendre l'histoire plus riche et émotionnelle, sans être trop redondante, caractéristiques qui sont importante non seulement lors de sa narration, mais de manière beaucoup plus générale. Voilà pourquoi la représentation employée ici, ainsi que les algorithmes mis en oeuvre sont généralisables à tous les autres cas où un agent doit communiquer et où il ne possède qu'un squelette (établi ou généré automatiquement à chaque instanciation) de planification du discours (scénario, méthode d'argumentation, modèle basé sur des prédicats ...).

Perspectives. Notre objectif principal n'était pas de créer la meilleure ontologie possible ni de créer une liste exhaustive de relations, mais de montrer que notre méthode marchait et qu'une réalisation simple laissait voir les possibilités et les potentialités de cette architecture. Nous nous proposons ici de montrer des améliorations possibles.

Même si l'ontologie en tant que représentation des potentialités doit rester fixe, son interprétation doit être dynamique, et c'est en cela qu'il faudrait que les événements puissent créer à la volée des relations entre les concepts. Imaginons qu'un personnage se déplace, il faudrait que la liaison "Personnage dans Ville" soit détruite, et que l'on crée "Personnage dans Campagne".

Enfin, un aspect qui n'est pas négligeable est la création d'une interface graphique pour l'ontologie qui simplifierait sa gestion, ce qui fait qu'elle pourrait grossir sans pour autant devenir illisible, permettant l'utilisation de bibliothèque de concepts et relations. Selon l'ambiance de l'histoire, selon les personnages mis en scène, selon l'ambiance voulue, le scénariste pourrait en effet faire appel à une bibliothèque particulière (film noir, science-fiction, roman historique...). L'intérêt de cette notion de librairie serait de créer une communauté autour de ce système, afin de rentabiliser dans le partage les efforts de chacun à la complexification de l'ontologie.

Références

[1] C. Peterson A. McCabe. What makes a good story ? *Journal of Psycholinguistic Research*, 13, 1984.

[2] J. Bates. The role of emotion in believable agents. *Communications of the ACM*, 37 :122–125, 1994.

[3] J.-L. Lugrin D. Pizz, F. Charles and M. Cavazza. Interactive storytelling with literary feelings. *ACII2007, Lisbon, Portugal*, 2007.

[4] J-L. Dessalles. Vers une modélisation de l'intérêt. *Actes des journées francophones 'Modèles formels de l'interaction'*, 2005.

[5] J-L. Dessalles. Le rôle de l'impact émotionnel dans la communication des événements. *Actes des journées francophones 'Modèles formels de l'interaction'*, pages 113–125, 2007.

[6] T. R. Gruber. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993.

[7] 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.

[8] A. Nijholt M. Theune, S. Faas and D. Heylen. The virtual storyteller. *ACM SIGGROUP Bulletin*, 2002.

[9] F. De Rosis and F. Grasso. Affective natural language generation. *Affective interactions*, 2000.

[10] J.F. Sowa. Conceptual structures : information processing in mind and machine. 1984.

[11] M. Stone. Ontology and description in computational semantics. 2004.