An AI-based Process for Generating Games from Flat Stories

Rosella Gennari and Sara Tonelli and Pierpaolo Vittorini

Abstract TERENCE is an FP7 ICT European project that is developing an adaptive learning system for supporting poor comprehenders and their educators. Its learning material are books of stories and games. The so-called smart games serve to stimulate the story comprehension. This paper focuses on the analysis of flat stories with a specific annotation language and the generation of smart games from the analysed texts, all done mixing natural language processing and temporal constraint-reasoning technologies. It first describes the annotation language, focusing on the key elements for the generation process and stressing which features depends on the requirements of the TERENCE learners. Then it shows how the language is used for annotating stories by the natural language processing module of TERENCE, and how the semantics is provided and is used by the constraint module of TERENCE. Finally, the paper illustrates the generation process of games from the annotation, with example. We conclude commenting on the approach to the automated analysis and extraction of information from stories for specific users and domains, briefly outlining the benefits of the semi-automatic generation process in terms of production costs.

Rosella Gennari
CS Faculty – Free University of Bozen-Bolzano – P.za Domenicani, 3 – 39100 Bolzano, IT e-mail: gennari@inf.unibz.it

Sara Tonelli
HLT – FBK-irst - Via Sommarive 18 – 38100 Bolzano, IT e-mail: stonelli@fbk.eu

Pierpaolo Vittorini
MISP – University of L'Aquila – P.le S. Tommasi, 1 – 67100 Coppito, L'Aquila, IT. e-mail: pierpaolo.vittorini@univaq.it

1 Introduction

1.1 The Problem

Developing the capabilities of children to comprehend written texts is key to their development as young adults. From the age of 7–8 until the age of 11, children develop as independent readers. Nowadays, more and more children in that age range turn out to be poor (text) comprehenders: they demonstrate difficulties in deep text comprehension, despite well developed low-level cognitive skills like vocabulary knowledge, e.g., see [27]. In particular, several experiments show that inference-making questions concerning events of a story and their relations are pedagogically effective in fostering deep comprehension of stories, e.g., see [17]. Few systems promote general reading interventions, but they have high-school or university textbooks as learning material, instead of stories, and are developed for old children or adults, and not specifically for poor comprehenders. TERENCE is a Collaborative project funded by the European Commission and developing the first intelligent *adaptive learning system* (ALS) [25] for poor comprehenders and their educators, from primary schools. The system proposes stories, organised into difficulty categories and collected into books, and smart games for reasoning about events of a story and their relations.

1.2 Rationale and Outline of This Paper

The TERENCE smart games are serious games [13] and, like the entire ALS, are designed within a therapy plan for learners with specific reading problems. See Sec. 2 of this paper for an overview of the games. One of the goals of TERENCE is to generate textual components for smart games semi-automatically, starting from flat stories for primary-school children, in Italian and in English, so as to improve development performances, given that we deal with 256 different stories, each having c. 12 smart games.

The starting point for such a generation process is the definition of a language that allows for the annotation of stories' key features for the smart games for the TERENCE users. The annotations are then used (1) for automatically generating textual components of smart games (2) and for rating these into difficulty levels. In particular, the paper shows how we combine *natural language processing* (NLP) techniques and tools, that recognise events and (causal-)temporal relations among them in stories, with *temporal constraint* (TC) algorithms and tools, that reason about events and temporal relations for generating textual components of games.

Firstly, the paper outlines the requirements for the language, resulting from contextual inquiries with c. 70 experts of poor compreheneders and educators. See Sect. 3. Given such a language, the automated analysis process can take place on stories by mixing NLP and constraint-based algorithms and tools, see Sec. 4. The

resulting annotated stories are then fed to the generation modules of TERENCE that, again integrating NLP and reasoning techniques, automatically generates games as explained in Sec. 5. This paper concludes with an analysis of the approach to the generation of games and proposes new working directions.

2 Background on the TERENCE Smart Games

According to [1], a game should specify (at least) the *instructions*, the *states* of the game, with the initial and terminal states, and the legal *actions* of the players. For specifying the data for the TERENCE smart games, we analysed the requirements for smart games resulting from contextual inquiries with c. 70 experts of poor comprehenders and educators as well as from field studies with c. 500 poor comprehenders. The results allowed us to classify and design smart games as follows. Following experts of therapy plans, the TERENCE smart games were classified into 3 main difficulty macro-levels, that is,

1. at the entry macro-level, *character* games, that is, either who the agent of a story event is (WHO), or what a character does in the story (WHAT);
2. at the intermediate macro-level, *time* games, for reasoning about temporal relations between events of the story, purely sequential (BEFORE-AFTER) or not (all the others);
3. at the last macro-level, *causality* games, namely, concerning causal-temporal relations between events of the story, that is, the cause of a given event (CAUSE), the effect (EFFECT), or the cause-effect relations between two events (CAUSE-EFFECT).

All games were specialised into *levels* arranged in a so-called linear layout, see the bottom green part in Fig. 1, and designed as puzzle casual games [1]. In particular, the data for all levels were structured via the TERENCE game framework. See [3, 7] for a description of the framework and the design process.

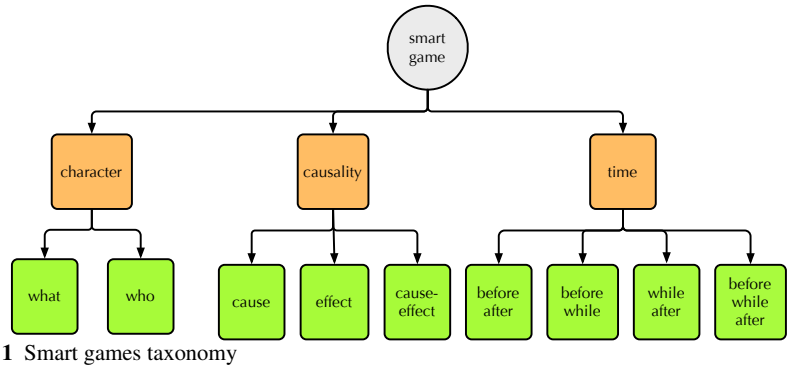

Fig. 1 Smart games taxonomy

Relevant fields of the framework for this paper are the following ones: (1) the *question* of the game, (2) a *central event* from the story, (3) the *choices* for learner's actions, so that the availability of choices depend on the state the game is in, and (4) which choices form a correct or wrong *solution*. The fields are rendered with illustrations and textual components that vary according to the level of the game and the specific game instance, as explained below.

In case of WHO games, the question is related to a single central event that depends on the game instance, if the central event is "Aidan runs fast" then the question is "Who runs fast?". The choices are 3 characters and only one is the correct solution, namely, the agent of the central event named "Aidan". See the left top screenshot in Fig. 2.

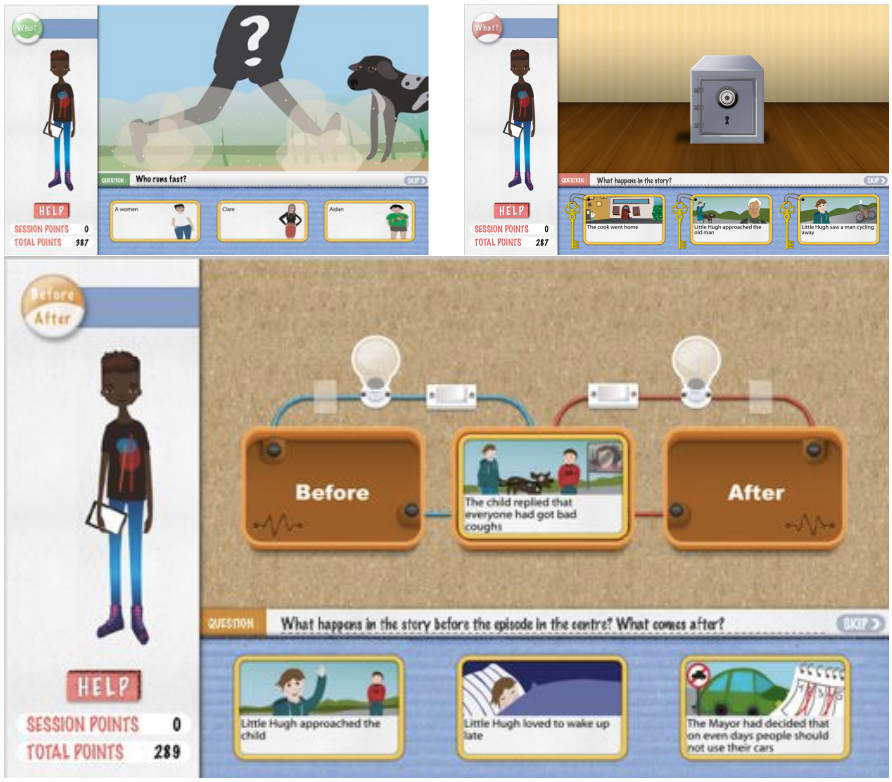

Fig. 2 Screenshots of WHO, WHAT and BEFORE-AFTER game instances.

In all the other levels, the question is related to the instructions for the game only, and is the same for all game instances. Choices are textual and visual descriptions of events from the story, and are placed at the bottom of the interface. In WHAT games, the central event is also the correct solution, and is placed at the bottom with the

other choices that are wrong solutions. See the right top screenshot in Fig. 2. In time and causality games, the central event is at the centre of the interface. The choices are at the bottom and are story events that the learner has to correctly correlate with the central event. See the bottom screenshot in Fig. 2, a BEFORE-AFTER time game instance.

3 The TERENCE Annotation Language

The events of a story, their temporal and causal-temporal relations, and the characters that participate in events are at the heart of the TERENCE smart games. Therefore TERENCE needs an annotation language that can specify them. TimeML [21, 23] is a good starting point for that: it covers events and qualitative temporal information, relevant for stories, and is the de-facto standard markup language for temporal information in the NLP community, which allows for the re-use of existing NLP and TC tools for qualitative temporal reasoning. See Ssect. 3.1 for an introduction to TimeML and Ssect. 3.2 for an analysis of the advantages and limitations of TimeML for TERENCE. The analysis of the limitations lies at the heart of the TERENCE annotation language, built on top of TimeML, and outlined in Ssect. 3.3.

3.1 The TimeML Annotation Format

TimeML [21, 23] is an international markup language for annotating events and temporal information in a text. It was designed in order to address the following four issues: *(i)* identify events and anchor them in time, *(ii)* order events, *(iii)* reason with contextually underspecified temporal expressions and *(iv)* reason about the persistence of events.

TimeML includes three major data structures, namely, a qualitative time entity (e.g., action verbs) called `EVENT`, a quantitative time entity (e.g., dates) called `TIMEX`, and relations among them called generically links. In particular, in TimeML events are characterised as follows in [26]:

> "Events" [is] a cover term for situations that happen or occur. Events can be punctual or last for a period of time. We also consider as events those predicates describing states or circumstances in which something obtains or holds true.

That implies that TimeML events can be expressed by tensed and untensed verbs, but also by nominalizations (e.g. *invasion*, *discussion*, *speech*), predicative clauses (e.g. *to be the President of something*), adjectives (e.g. *dormant*) or prepositional phrases (e.g. *on board*). TimeML is by now a standard in the NLP community and is used in the annotation of linguistic resources such as the TimeBank corpus [22], the Ita-TimeBank [5] and of news events in the 2010 TempEval competition [30].

As for `TIMEX` temporal expressions, they include specific dates (e.g. *June 11, 1989*), and durations (*three months*). TimeML includes also the concept of time

normalisation, where a normalised form is assigned to each expression based on consistency and interchange format in line with the ISO 8601 standard for representing dates, times, and durations.

Time entities (`EVENT` and `TIMEX`) can be connected through a qualitative temporal relation called `TLINK`, making explicit it the two entities are simultaneous, identical or in sequential order. Specifically, several types of relations have been introduced in TimeML modelled on *Allen interval algebra* of qualitative temporal relations [2]. Other types of links are also present in TimeML standard, for instance subordination links between specific verbs of reporting and opinion and their event arguments, or aspectual links between aspectual events and their argument events. However, since they are not included in the TERENCE framework, we do not discuss them in details.

3.2 Pros and Contras of TimeML for TERENCE

TimeML events and their relations are key elements for representing the semantics of children's stories and for generating the TERENCE smart games. Moreover, TimeML is the de-facto standard language on which the NLP communities have grown their experience and tools in the last decade. However, TimeML is sometimes underspecified and sometimes too detailed for the TERENCE purposes.

For instance, the semantics of `TLINKs` in terms of Allen relations is not uniquely specified but, in the working practice, is pretty context dependent. Key information like the participants in an event have already been proposed [24] but neither their attributes nor their relations with events have been clearly pinpointed. Since knowing which are the characters in a story and how they interact is crucial for the TERENCE smart games, such information is essential for analysing the TERENCE stories. Another main problem for TERENCE with the TimeML language is the lack of causal links when explicitly signaled by linguistic cues such as "because" and "since".

In several other cases, the level of detail foreseen in TimeML is not needed to process children's stories like in TERENCE. For instance, while in TimeML modal and hypothetical actions are annotated as events, the primary interest of TERENCE are the restricted events that actually take place in the story, namely, *factual* events.

3.3 The TERENCE Annotation Format (TAF)

In order to exploit the tools and body of knowledge created by the NLP communities for TimeML, we built the *TERENCE Annotation Format* (TAF) on top of TimeML. See [16]. Hereby we outline the main design choices of the language, relevant for generating diverse smart games, their wrong and correct solutions, as well as for classifying smart games into fine-grained difficulty categories.

The main requirements leading to the design of the language are recapped in Table 1. For the analyses leading to such requirements, see [27]. For instance, morphological information could be relevant for deaf poor comprehenders, e.g., they tend to have problems with verbs that are irregular or in passive forms. Referential expressions, especially if the referent and the reference are mentioned in distant sentences, can be difficult for poor comprehenders, who tend to reason on isolated chunks of text, e.g., the same or consecutive sentences. Well-placed signals like connectives tend to easy their understanding of relations between events. Thus it is important to trace whether, for instance, a temporal or causal-temporal relation between events is rendered by a connective, and if the events occur in the same sentence.

Concerning	**Main requirements**
events	factual event, in particular, if with an emotional appeal
	the root form (a.k.a., stem, lemma, infinitive form) of the core event, that is, its verb; the tense, mood and irregularity of this verb
	the main participants in an event, and the role (a.k.a., type)
	whether the participants in an event are characters of the story
	features of the character like its role, if animated or not
	the location of an event
	if a participant is via co-reference and the distance from the referent
relations	the type of relation, that is, temporal or causal
	if rendered with signals
	if an explicit temporal signal, the order
	if the correlated events occur in the same sentence/consecutive sentences and, if not, the distance between the events

Table 1: Requirements for the annotation language of TERENCE.

For explaining TAF, we use the *Extended Entity-Relationship* (EER) [28] diagram in Figure 3. Therein, classes are coloured in pink. Relations among classes are yellow. Attributes are in balloons. Each class has a unique key identifying attribute. Cardinality constraints between an entity and an attribute are equal to $(1,1)$ unless differently set.

The diagram in Figure 3 specifies the classes of `Event` and `Pos`, denoting an event and the lemma of the word evoking it, respectively. The attributes of `Pos` specify relevant information, in particular, if it is a verb and, e.g., its regularity.

In the diagram in Figure 3, we have `TIMEX` and `TLINKs`. Since children's stories, like in TERENCE, are not usually anchored to a specific date, time normalisation is often not possible, therefore temporal expressions are not normalised as opposed to TimeML standard. Therefore we focus on the analysis of a `TLINK`, which is relevant for time games (see Sect. 2). A `TLINK` denotes a temporal binary relation between events, and has the following attributes:

1. `reverse`, with Booleans "true" or "false", capturing if the order of occurrence of the two events in the story text is the temporal order ("false") or not ("true");

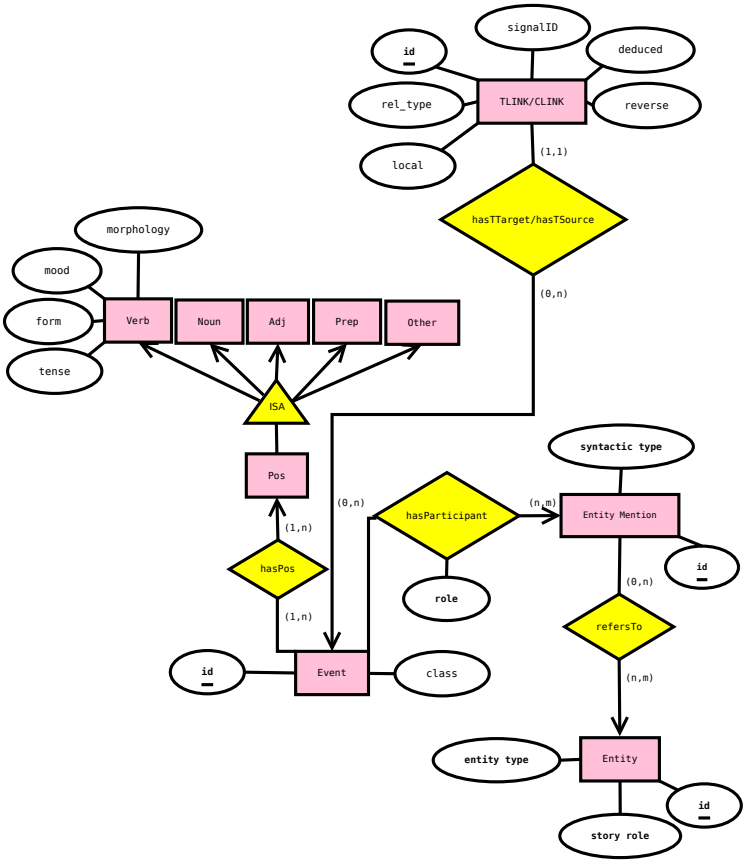

Fig. 3 EER of events and their relations

2. `signalID`, with string values the id of the annotated signal (e.g., “30” for the signal “and then”) if the two `Events` are related through an explicit temporal signal, or “none” otherwise;
3. `rel_type`, with string value the TimeML relations “before”, “after”, “overlaps”, “is_included”, “includes”, “identity”, “all”;
4. `local`, with Boolean values “true” if the two `Events` are in the same sentence or in consecutive sentences, or “false” otherwise.

Except for `rel_type`, such attributes are not present in TimeML. The `hasTTarget` and `hasTSource` relations serve to specify, respectively, the source event and the target event of the relation, which is done differently than in TimeML. Similar remarks apply for the `CLINK` class, its attributes, the `hasTTarget` and `hasTSource` relations. In particular, in TAF and not in TimeML we have causal links between events when explicitly signaled by linguistic cues such as “because”, “since”. The `CLINKs` are used for causality games (see Sect. 2).

The diagram in Figure 3 also introduces classes and relations for specifying the characters or other entities participating in events of a story, as well as their role within the story (see Sect. 2). In particular, `Entity` refers to the unique identifier of a participant, like a character in a story (e.g., the proper name "Ben" of a male character), whereas the `Entity Mention` stands for any expression that correlates via `refersTo` to a unique `Entity` (e.g., "the boy"); such a distinction between entities and their story mentions has been introduced to account for coreference in TAF, which is not covered in TimeML. The relation `hasParticipant` correlates each `Event` to all the `Entity Mentions` involved in such event. The relation has the composite attribute named `role`, which stands in for the role that a correlated entity mention has in the event. One of its atomic attributes is `semantic role`, and among its values we have "agent" for the agents of an event. While `Entity Mentions` are described through the syntactic type they have in the sentence in which they occur (e.g. "subject", "object"). `Entities` are characterised at the semantic level with the following attributes with string values:

1. `entity type`, with values "person", "location", "animal", "pseudo-person", "vegetable", "artifact", "other";
2. `story role`, with values "protagonist", "secondary", "minor".

Such information is relevant for the generation of more plausible wrong solutions in games and for organising them into fine-grained difficulty categories.

4 The Automated Analysis Process

The NLP module and the TC module perform the annotation process of stories of TERENCE with the TAF language described above. Firstly, the NLP module detects relevant information and annotates them with TAF. See Ssect. 4.1. Secondly, the TC reasoning module transform `EVENTs` and `TLINKs` into a qualitative temporal constraint problem, mapping `TLINKs` into Allen relations; in case the problem is detected inconsistent, the mapping is relaxed. If the problem is returned consistent, then the TC reasoning module deduces further relations, annotating their distance in the text. See Sect. 4.2. The resulting stories are stored as XML files in the annotated story repository.

4.1 The NLP Analysis of Flat Stories

The goal of this processing step is to take in input a flat story and output an XML-based version of the story annotated with the TAF language. We describe here the workflow for Italian stories, and a similar approach has been tested also for English. While the tools used for processing the stories vary from language to language,

the approach is language-independent, and also TAF has been formulated so as to minimise language-specific adaptations.

The story is first processed with existing NLP tools for Italian. Specifically, it is analysed with the TextPro suite [20] which performs part-of-speech tagging, lemmatization, morphological analysis, named entity recognition and annotation of temporal expressions. The story is then processed with the Malt parser for Italian [15], which outputs labeled dependency relations between tokens in the story (for instance, it identifies the subject and the object of a sentence and links them to the verb).

This intermediate analysis is then passed to the TAF annotation module that adds information on events, temporal links, causal links and participants to the preprocessed data in a cascaded fashion. The performed steps are the following ones.

Annotation of factual events. Factual `Events` are selected among tensed verbs that are not copulas and modals, and that are in the indicative mood. We discard verbs in the future tense, and expressing wish or conditionality.

Annotation of event participants. For each selected `Event`, we retrieve the syntactic dependents from the parsing analysis, which we label as `Entity Mentions`. For each mention, we link its head to the most frequent sense in MultiWordNet [19] and then get the corresponding type label from the Suggested Upper Merged Ontology [18], which we further mapped to the TAF `entity types`. For instance, "the cat" may be linked to the *cat#n#1* synset in MultiWordNet, which corresponds to the *Feline* class in SUMO. *Feline* is then mapped to the *Animal* `entity type`.

Annotation of participants' coreference. If two or more participants' mentions show a (partial) string match, they are annotated as co-referring via an `Entity` to which both are linked through a `refers_to` relation. This concerns for instance mentions such as "Ernesta Sparalesta" and "Ernesta". More sophisticated algorithms for coreference annotation are currently being evaluated within TERENCE, for instance supervised approaches to treat pronominal anaphora and zero-pronouns. But current results are still too low to be integrated in the final NLP pipeline.

Annotation of temporal links. Events that are in the same sentence or in contiguous sentences are local. For each pair of local events, a `TLINK` is created and its `rel_type` specified based on the tense and mood information associated with the events. Since events in children's stories tend to follow a chronological order, the default value for `rel_type` is "before", indicating that the event mentioned first (e_1) precedes the other (e_2). However if e_1 is at present tense and e_2 is expressed in a past tense, the relation is likely to be "after". Cases of "includes" and "is_included" are annotated when one of the two events is in a continuous form (e.g., "was playing") and the other is a punctual event (e.g., "noticed"). In this manner, a temporal chain is created for the story.

Annotation of causal links. Temporal links between events are also marked with causal links (`CLINKs`) if the events occur in two clauses connected by a causal marker such as "because", "for this reason", "so that".

The annotation module outputs an XML file with stand-off annotation which is then fed to the reasoning module.

4.2 The TC Enrichment of Annotated Stories

The TC reasoning module takes in input a story annotated by the NLP module from the annotated story repository, and processes it as explained hereby. First of all, the architecture of the reasoning module is made up of:

- a REST service, that contains two main operations: consistency checking and deduction;
- a Java library `tml2gqr`, which is used by the service for implementing the operations, but can be also embedded in other applications for performances' reasons;
- the GQR software [10], invoked by the library for performing the above operations.

The operations are outlined and executed as follows.

Consistency checking. The static method `consistency` of class `GQR` takes in input a Java `String` and returns a Java `boolean`. The input is the TAF document annotated by the NLP module. The TAF document is converted into a TC problem file, with disjunctive Allen relations [2]. The most critical task is to choose the 'right' semantics for `TLINKs`, that is, how to convert `TLINKs` into relations of a tractable subalgebra of the Allen interval algebra [2, 14] for the TERENCE stories. The chosen mappings were two and agreed upon with the NLP partner as those returning inter-consistency among manual annotators, within pilot studies for English and Italian children stories. The hard mapping is reported in Table 2. The relaxed mapping differs from the hard one in that "before" is mapped into "before or meets" and "after" into its inverse "before^{-1} or meets^{-1}". For both mappings the range is a subalgebra of the *continuous algebra* (CA), for which consistency checking and deduction take at worst cubic time in the number of events [11]. In fact, both consistency checking and deduction are done via the optimised path consistency algorithm of `GQR`, which runs in time cubic in the number of events. The output is then parsed and whether the document is consistent is returned as a Boolean. In case not, human interventions is invoked. Else, deduction is invoked.

Deduction. The static method `deduction` has in input the TAF document checked for consistency, with either the hard or relaxed mapping. The TAF document is converted into a TC file in the format of `GQR`, with the mapping for which the document is consistent. Then `GQR` is executed. The output of `GQR` is then parsed, and the deduced relations are added with `TLINKs` via the inverse mapping. In case a relation r deduced by the reasoner corresponds to no one in the range of adopted mapping for `TLINKs`, then r is approximated by the smallest (for inclusion) relation in the range of the mapping and that contains r, and a comment is added with the deduced relation for further processing—human or

automated. The input TAF document is updated, and the updated TAF document is returned as XML file in the annotated story repository.

The story in the annotated story repository is referred to as the *enriched* story, used as input for the generation process described below.

`rel_types` for `TLINKs`	**Allen relation(s)**
before	before
after	before^{-1}
overlaps	equal or during or during^{-1} or overlaps or overlaps^{-1} or starts or starts^{-1} or finishes^{-1} or finishes^{-1}
includes	during^{-1}
is_included	during
identity	equal
all	the disjunction of all Allen atomic relations

Table 2: Mapping of TLINKs into disjunctive Allen relations.

5 The Automated Generation Process

The reasoning module and the NLP also collaborate for generating textual components for smart games, to which we refer as *textual games*. More precisely, the TC reasoning module takes in input the enriched stories, described in the previous section, and ranks events according to their relevance for smart games. Textual game instances, for each level (see Sect. 2), are generated for the top ranked events. The process is described in [12, 7]. In particular, for each textual game instance, according to its level, the TC reasoning module sets its data structure, the available choices, wrong and correct solutions, as outlined in Ssect. 5.1. Then, for each textual game instance, for each event in it, the TC module invokes the NLP module that generates the necessary textual information for the event, as outlined in Ssect. 5.2. The TC module then places the generated text in the pertinent textual game instances. See Fig. 4 for the overall workflow.

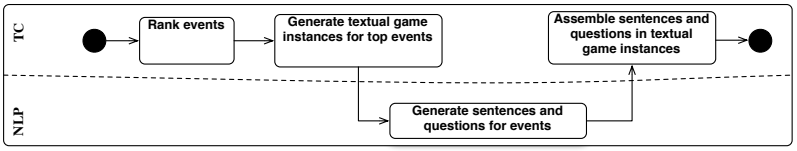

Fig. 4 The generation workflow

5.1 The Generation of Game Instances from Enriched Stories

In the following, we explain how choices, correct and wrong solutions are generated by giving examples of generated games for each macro-level. The generation process takes as input an enriched story and proceeds as follows.

Character games. In order to create a WHO game instance for an event, we exploit the existence or not of a chain that, starting from the event marked with `Event`, continues to the participant related via `hasParticipant` and marked with `Entity Mention`. This is then resolved by correlating it via `refersTo` to its `Entity`. Fig. 5 depicts the case of two entities, i.e., Ben and Kate, mentioned in the story as "The boy", and "The girl", respectively, and their participation in two events, i.e., "swim" and "run".

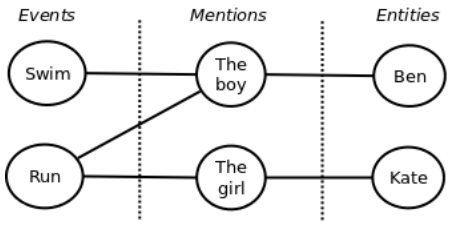

Fig. 5 TERENCE TimeML tags for a WHO smart game instance.

For instance, to generate the WHO game for the central event "swim", the module exploits the links that correlate

- "swim" to the respective mentions, then to the actual entities (in the example, Ben), for the correct choice,
- other entities that are not mentioned as participants in the "swim" event as wrong (e.g., Kate).

A similar process takes place with WHAT character games. Given an event in the story, this is taken as the correct choice, whereas the wrong choices are from either a different story, or from the same story but using a different entity as event participant. In the case depicted in Fig. 5, the correct choice would be "Ben swims", and a wrong choice would be "Kate swims".

Time games. In order to create a time game with a given central event, we exploit the `Event` and `TLINK` tags, as well as the `local` and `rel_type` attributes. Fig. 6 shows a portion of the temporal structure of a story, where the event "swim" occurs before "run", "jump" occurs at the same time as "run", and "rest" occurs after "run". Furthermore, two TLINKs are not local and were deduced by the TC module in the story annotation process, i.e., that "swim" occurs before "rest".

For instance, in order to create a BEFORE-AFTER-game instance difficult for a poor comprehender, we can:

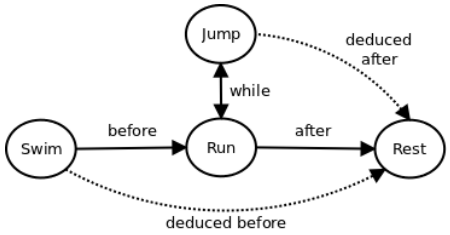

Fig. 6 TERENCE TimeML tags for time smart game instances.

- exploit the `TLINK` that has a `rel_type` equal either to "before" or "after",
- consider relations with `local` set to false (i.e., jump-rest, swim-rest).

Causality games. In order to create causality games with a given central event, we exploit `CLINKs` and the `reverse` attribute, mainly. Fig. 7 depicts causal relations in a story where a race was lost because the participant did not sleep enough, and that caused the participant to become sad.

Fig. 7 TERENCE TimeML tags for causality smart game instances.

For instance, to create a CAUSE-game, we can exploit the `CLINKs` stating that the cause for "The boy is sad" is that "He lost the race", and properly swap the cause and the effect, in case the `reverse` attribute is set to true ("The boy lost the race" because "He did not sleep").

5.2 The Generation of Textual Components for Game Instances

The NLP generation task, given in input the identifier of a story event, generates *i)* the question for WHO games for that event, and *ii)* the sentence for the event used as choice of the games of all the other levels.

The generation task is performed with a modified version of the sentence simplification system presented in [4]. Specifically, a module has been developed that, taking a TAF-annotated text in input, outputs a sequence of simplified statements for each identified event, and WHO-questions on the event subjects. The module removes from the dependency tree the event arguments that are not mandatory, i.e., that do not correspond to participants. For instance, in case of transitive verbs, it only retains subject and object. Other arguments labeled as RMOD (modifiers) are removed. Then, the NLP module puts the verb in the present indicative tense using a generator of morphological forms included in TextPro [20]. Finally, it generates questions on the subject of the simplified clause: it retrieves the corresponding en-

tity type saved in TAF, and uses it to generate a WHO question. We describe the four steps performed by the NLP module in the light of the following example:

(1)[Kate$_{Subj/Person}$] [was eating$_{event}$] [a cake$_{Obj}$] [with her friends$_{RMod}$]
(2)[Kate $_{Subj/Person}$] [was eating$_{event}$] [a cake$_{Obj}$]
(3)[Kate $_{Subj/Person}$] [eats$_{event}$] [a cake$_{Obj}$]
(4)[Who$_{Subj/Person}$] [eats$_{event}$] [a cake$_{Obj}$] ?

Sentence (1) is the output of the process described in Sect. 4.1, in which TAF annotations are added. Specifically, "Kate" is identified as the subject being a person, "was eating" is the event and the two following arguments are an object and a modifier. In Sentence (2), the modifier, not mandatory, has been deleted. In Sentence (3) the event is expressed in the present tense, while in (4) the simplified statement has been transformed into a WHO question, given that the subject is a person.

6 Conclusions

The paper shows how textual components of smart games for learners with specific text comprehension problems are generated via an automated process, mixing NLP and a qualitative constraint-based technologies and tools. In the remainder, we briefly assess such a generation approach. Firstly, the automated generation requires an annotation language defined on top of the learners' requirements, e.g., the language allows for specifying whether the events are not in the same sentence. That is per se a novelty element. Secondly, releases of the system and, in particular, its games are also evaluated iteratively, in four main evaluations; the second release for the TERENCE learners is available at [29]. One of such evaluations was concerned with the generated textual components of smart games, and it was done manually for all the TERENCE games by experts of education or poor comprehenders. According to this evaluation, the work of developing manual games was longer than for generating and revising them—on average 23 minutes vs 13 minutes. That indicates that the generation process is promising for cutting development costs. See [8] for such a revision process. Another evaluation was instead conducted as a field study with 168 learners for detecting usability issues and gaining further indications concerning the pedagogical effectiveness of smart games, see [6] and [9]. For instance, this evaluation indicated the need of optimising the generation of smart games with heuristics for more plausible wrong solutions, an on-going work done with NLP and constraint-based reasoning techniques. Another on-going work is the optimisation of the annotation process of stories, by interleaving NLP and temporal constraint reasoning enhanced with explanation capabilities, according to the specific application domain and intended users.

Acknowledgments.

The authors' work was supported by the TERENCE project. TERENCE is funded by the European Commission through the Seventh Framework Programme for RTD, Strategic Objective ICT-2009.4.2, ICT, Technology-enhanced learning. The contents of the paper reflects only the authors' view and the European Commission is not liable for it. Gennari work was also supported by the DARE project, financed by the Province of Bozen-Bolzano.

References

1. E. Adams. *Fundamentals of Game Design*. New Riders, 2010.
2. J. F. Allen. Maintaining Knowledge about Temporal Intervals. *ACM Comm*, 26:832–843, 1983.
3. M. Alrifai and R. Gennari. Deliverable 2.3: Game Design. Technical Report D2.3, TERENCE project, 2012.
4. G. Barlacchi and S. Tonelli. ERNESTA: A Sentence Simplification Tool for Children's Stories in Italian. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 7817 of *Lecture Notes in Computer Science*, pages 476–487. Springer Berlin Heidelberg, 2013.
5. T. Caselli, V. B. Lenzi, R. Sprugnoli, E. Pianta, and I. Prodanof. Annotating Events, Temporal Expressions and Relations in Italian: the It-TimeML Experience for the Ita-TimeBank. In *Proceedings of LAW V*, Portland, Oregon, USA, 2011.
6. M. Cecilia, T. D. Mascio, and A. Melonio. The 1st Release of the TERENCE Learner GUI: the User-based Usability Evaluation . In *Proc. of the 2nd evidence-based TEL workshop (ebTEL 2013)*, Advances in Intelligent and Soft Computing. Springer, 2013.
7. V. Cofini, F. de la Prieta, T. D. Mascio, R. Gennari, and P. Vittorini. Design Smart Games with Context, Generate them with a Click, and Revise them with a GUI. *Advances in Distributed Computing and Artificial Intelligence Journal*, 3, Dec. 2012 2012.
8. V. Cofini, R. Gennari, and P. Vittorini. The Manual Revision of the TERENCE Italian Smart Games. In P. e. a. Vittorini, editor, *Proc. of the 2nd evidence-based TEL workshop (ebTEL 2013)*. Springer, 2013.
9. F. de la Prieta, T. D. Mascio, R. Gennari, I. Marenzi, and P. Vittorini. User-centred and Evidence-based Design of Smart Games. *International Journal of Technology Enhanced Learning (IJTEL)*, Submitted for PDSG special issue in 2013.
10. Z. Gantner, M. Westphal, and S. Wölfl. GQR - A Fast Reasoner for Binary Qualitative Constraint Calculi. In *AAAI'08 Workshop on Spatial and Temporal Reasoning*, 2008.
11. R. Gennari. Temporal Constraint Programming: a Survey. *CWI Quarterly Report*, 1998.
12. R. Gennari. Generation Service and Repository of Textual Smart Games. Technical report, TERENCE project, 2012.
13. M. S. Jong, J. H. Lee, and J. Shang. *Reshaping Learning*, chapter Educational Use of Computer Games: Where We Are, And What's Next. Springer, 2013.
14. A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about Temporal Relations: The Tractable Subalgebras of Allen's Interval Algebra. *Journal of ACM*, 50(5):591–640, 2005.
15. A. Lavelli, J. Hall, J. Nilsson, and J. Nivre. MaltParser at the EVALITA 2009 Dependency Parsing Task. In *Proceedings of EVALITA Evaluation Campaign*, 2009.
16. S. Moens. Deliverable 3.1: State of the Art and Design of Novel Annotation Languages and Technologies. Technical Report D3.1, TERENCE project, 2012.
17. N. R. Panel. Teaching Children to Read: An evidence-based Assessment of the Scientific Research Literature on Reading and its Implications for Reading Instruction. Technical report, National Institute of Child Health and Human Development, 2000.

18. A. Pease, I. Niles, and J. Li. The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications. *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, 2002.
19. E. Pianta, L. Bentivogli, and C. Girardi. MultiWordNet: developing an aligned multilingual database. In *First International Conference on Global WordNet*, pages 292–302, Mysore, India, 2002.
20. E. Pianta, C. Girardi, and R. Zanoli. The TextPro tool suite. In *Proc. of the 6th Language Resources and Evaluation Conference (LREC)*, Marrakech, Morocco, 2008.
21. J. Pustejovsky, J. Castano, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, G. Katz, and D. Radev. TimeML: Robust specification of event and temporal expressions in text. In *IWCS-5 Fifth International Workshop on Computational Semantics*, 2003.
22. J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, et al. The TimeBank corpus. In *Corpus Linguistics*, volume 2003, page 40, 2003.
23. J. Pustejovsky, K. Lee, H. Bunt, and L. Romary. ISO-TimeML: An International Standard for Semantic Annotation. In *Proceedings of the Fifth International Workshop on Interoperable Semantic Annotation (ISA-5)*, 2010.
24. J. Pustejovsky, J. Littman, and R. Saurí. Arguments in TimeML: events and entities. *Annotating, Extracting and Reasoning about Time and Events*, pages 107–126, 2007.
25. J. Santos, L. Anido, M. Llamas, L. Álvarez, and F. Mikic. *Computational Science*, chapter Applying Computational Science Techniques to Support Adaptive Learning, pages 1079–1087. Lecture Notes in Computer Science 2658. 2003.
26. R. Saurí, J. Littman, R. Knippen, R. Gaizauskas, A. Setzer, and J. Pustejovsky. TimeML 1.2.1 Annotation Guidelines, October 2005. `http://timeml.org/site/publications/timeMLdocs/annguide_1.2.1.pdf`.
27. K. Slegers and R. Gennari. Deliverable 1.1: State of the Art of Methods for the User Analysis and Description of Context of Use. Technical Report D1.1, TERENCE project, 2011.
28. T. J. Teorey, D. Yang, and J. P. Fry. A Logical Design Methodology for Relational Databases using the Extended Entity-relationship Model. In *ACM Computing Surveys*. ACM, 1986.
29. TERENCE Consortium. The 2nd release of the TERENCE system for poor comprehenders.
30. M. Verhagen, R. Sauri, T. Caselli, and J. Pustejovsky. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, Uppsala, Sweden, July 2010. Association for Computational Linguistics.