Generated Narratives for Computer-aided Language Teaching

Michael LEVISON
School of Computing
Queen's University
Kingston, Ontario
Canada K7L 3N6
levison@cs.queensu.ca

Greg LESSARD
French Studies
Queen's University
Kingston, Ontario
Canada K7L 3N6
lessardg@post.queensu.ca

Abstract

VINCI is a Natural Language Generation environment designed for use in computer-aided second language instruction. It dynamically generates multiple parallel trees representing an initial text, questions on this text, and expected answers, and either orthographic or phonetic output. Analyses of a learner's answers to questions are used to diagnose comprehension and language skills and to adaptively control subsequent generation. The paper traces stages in the generation of short texts in English and French, and discusses issues of architecture, textual enrichment, and planning.

1 VINCI: Architecture, Implementation and Output

Ideally, a language teaching system should both "encourage the creative use of language in communicatively relevant settings" (Menzel and Schroeder, 1998) and also provide detailed and adaptive feedback (cf. (Michaud, 2002)). Many systems resolve the issue by means of complex parsing. In what follows, we describe VINCI, a multilingual generation environment which represents a complementary approach in that it assumes conversational control, dynamically producing more or less complex texts and asking information of users. VINCI is based on a collection of metalanguages which define the semantics, syntax, lexicon and morphology of a language. Text files defining the language are read by an interpreter (written in C) and output is presented either orthographically or phonetically.

When used in a teaching context, VINCI creates an utterance and presents it to a learner. It also creates a series of questions based on each utterance, together with some hidden answers. The learner is prompted to respond to each question, and his or her response is compared with the hidden one (or ones) and a detailed report is produced on the relation between the two. This report provides information on a learner's comprehension and language skills, as well as guidance for subsequent generation.

VINCI is capable of generation in many languages, and has been tested on such diverse languages as Spanish, Italian, Russian and Chinese. Our work to date has been carried out in both English and French, predominantly the latter.

For the generation of simple utterances, VINCI constructs a syntax tree using context-free rules and syntactic transformations. The nodes of the tree may be decorated with attributes, whose role is to maintain grammatical and perhaps semantic agreement by restricting the choice of lexical items and controlling morphology. Once a tree is formed, its leaves are assigned suitable lexical choices which are inflected by morphology rules. In the most basic situation, choices among syntactic alternatives and among possible lexical entries are made at random. In such a model, the semantic control exercised by attributes is minimal. They can, for example, ensure that the subject of a verb such as *eat* is animate and its object edible, but they cannot influence the overall meaning of the utterance produced. To achieve this, VINCI makes use of preselections.

1.1 Preselections

Preselections may be thought of as forming a metaphoric blackboard on which the person requiring an utterance writes some choices about its features. Lexical entries corresponding to these features are selected before the syntax tree is formed, and syntax rules have access both to the words themselves and to information about them obtained from the lexicon. We will illustrate this by showing the steps in the generation of a short fairy tale, although other sorts of texts are also possible.

In a typical fairy tale we have a collection of characters, including a pompous `twit` (a king or a rich merchant), a `victim` or heroine (the twit's daughter), a `villain` (a sorcerer or witch), a `hero` (a prince or a brave woodcutter), a `goodfairy` (the victim's fairy godmother), a `magicobj` (a sword or a silver goblet). These form the basis for a set of preselections such as:

```
twit :
   PN/"Midas";
victim:
   PN/_pre_ twit/@14:  daughter;
hero :
   PN[male, brave, handsome];
magicobj :
   N[physobj, magic]
```

These preselections presuppose a database of characters/objects which are simply entries in a lexicon:

```
"Midas"|PN|
   male, rich, weak, vain|...
      daughter:  "Marie";...
      type:  "king"; ...
      home:  "castle"; ...
"Marie"|PN|
   female, beautiful, kind|...
      type:  "princess"; ...
```

Preselections can be specified with more or less precision. In this example, only *Midas* can be chosen for the role of twit, but any member of the class PN (proper names) having the attributes male, brave and handsome can be selected as hero. We might well have written `twit: PN[rich]/@14: daughter` making the twit any PN who is rich and has a pointer to a daughter in lexical field 14.

These preselections are global, in that they persist across several utterances. If the hero is a prince in one sentence, he cannot be a woodcutter in the next. In contrast, the local preselections described below associate these characters with a semantic role in a particular sentence, for example, the agent or the patient. We can envisage a user typing/editing a global preselections file to select favorite characters for a story. Alternatively, there may be an interface which allows a user to choose characters from a set of menus. In the following tale, we assume that *Wanda* has been preselected as the goodfairy and *magic sword* as magicobj.

1.2 Semantic Expressions

A semantic expression is a representation of the content of an utterance in a form in which the grammatical constraints of any particular natural language have been abstracted away, leaving only some expression of meaning behind. A simple functional notation is used, described more fully in (Levison et al., 2001b). These expressions are transformed into VINCI preselections, triggering syntax rules which, in their turn, yield sentences in some language. The same sequence of expressions can transformed into paragraphs in different languages or different paragraphs in the same language.

The plot for the fairy tale can be specified as a sequence of these expressions:

`exists(twit)`
Once upon a time there was a twit.
`describe(twit)`
He was rich and vain.
`exists(victim)`
The twit had a daughter, the victim.
`describe(victim)`
She was beautiful and kind.
`admonish(twit, victim, action)`
The twit warned the victim about walking in the forest.
`disobey(victim)`
However, the victim disobeyed.
`action(victim)`
She went for a walk in the forest.
`exists(villain)`
In the forest there lived a villain.
`describe(villain)`
He was strong and evil.
`kidnap(villain, victim)`
The villain kidnapped the victim.
`exists(hero)`
In the same area, there lived a hero.
`seekhelp(twit, hero)`
The twit sought his help.
`seek(hero, goodfairy)`
The hero went to find the goodfairy...
`give(goodfairy, hero, magicobj)`
who gave him a magicobj.
`seek(hero, villain)`
The hero sought the villain...
`kill(hero, villain, magicobj)`
and killed him with the magicobj.
`rescue(hero, victim)`
The hero rescued the victim, ...
`marry(hero, victim)`
married her, ...

`livehappily(hero, victim)`
and lived happily ever after.

Obviously, the plot can be modified simply by varying the expressions. Indeed there might be alternative plots or sections, perhaps chosen by a user or produced by a text planner. Let us repeat that these expressions are language-independent. The names of the functions and parameters are, in fact, VINCI attributes, and although English words have been used here, any string of letters or digits could have been substituted. If a French grammar and lexicon is built using the same attributes, as in: `"donner"|V|vtdi, give, ...|...` then VINCI can construct French sentences from the same semantic expressions.

1.3 Local Preselections

Each of the expressions in the previous section is equivalent to and is transformed by VINCI into a set of local preselections which apply to the generation of a single sentence. For example, `give(goodfairy, hero, magicobj)` becomes:

```
vtdi; {this attribute selects a
      segment of syntax based
      on a verb with direct and
      indirect objects}
act :  V[give];
      {e.g.:  "give","offer"}
agent :  PN/_pre_ goodfairy;
beneficiary :  PN/_pre_ hero;
theme :  N/_pre_ magicobj
```

Some of these local preselections refer back to the global ones, associating the characters selected in the earlier preselections with the semantic roles they will play in the current sentence: agent, beneficiary and theme. So `goodfairy` (and hence *Wanda*) becomes the agent of the act of giving, `magicobj` (the *magic sword*) becomes its theme, and `hero` its beneficiary.

In effect, semantic expressions provide a more user-friendly form for the set of preselections. Conversion from the former to the latter is effected by semantic transformations; for example:

```
give :  vtdi;
act :  V[give];
agent :  PN/_pre_ #1;
beneficiary :  PN/_pre_ #2;
theme :  N/_pre_ #3
```

whose left-hand side matches `give(goodfairy, hero, magicobj)`, #1 being associated with `goodfairy`, #2 with `hero` and #3 with `magicobj`. In practice it shorter, if less elegant, to replace this semantic expression by `vtdi(give, goodfairy, hero, magicobj)`, since this single expression can be used for any verb taking both direct and indirect objects.

1.4 Syntax Rules

Syntax rules in Vinci take the abstract semantic representations produced by semantic expressions and preselections and clothe them in the syntax of the language chosen. Among other things, this allows the system to capture constraints on word order, argument structure, and agreement. This is accomplished by means of inheritance of attributes down the nodes of a syntax tree, and guarded syntax rules, in which attributes present on a parent node are used to determine the nature of child nodes. For example, given a parent node such as NP (noun phrase) containing the attribute 'p1' (first person), a guarded syntax rule (these are headed by the symbol <) will determine that the only possible child node is a pronoun. However, in the default case (these are headed by the symbol >), the child may take either the form of a pronoun or a full noun phrase.

Let us now return to the example of preselections developed above and resume with the action of syntax rules on the output of preselections and semantic expressions. The section of the context-free rules corresponding to vtdi describe the structure of a sentence with a vtdi verb in terms of the its agent, theme and beneficiary. Thus, assuming the local preselections given above:

```
ROOT =
< _pre_ vtdi:
    NPP[sing, agent, def]
    V[p3, sing, past]/_pre_ act
    NPP[sing, beneficiary, def]
    NP[sing, theme, indef] %

NPP = inherit Fn: Function, Nu: Number,
              De: Detkind;
    DET[De] N[Nu]/_pre_ Fn/@13:type %

NP  = inherit Fn: Function, Nu: Number,
              De: Detkind;
    DET[De] N[Nu]/_pre_ Fn %
```

The root of the utterance (the top of its syntax tree) has four child nodes, two of them

proper noun phrases (NPP), a third a common noun phrase (NP). To the first it passes a Number-value, sing, a Function-value, agent, and a Detkind-value. When this NPP is developed into its two children, it assigns these attribute values to Nu, Fn and De, passing De to DET (hence *a* or *the*) and Nu to the child noun, which will therefore be singular. This noun is also directed to obtain its lexical entry from the preselection labelled `Fn` (i.e. `agent`, which in turn refers to `goodfairy`, and hence to *Wanda*). Furthermore, rather than using *Wanda* itself, the chosen noun must be replaced the lexical entry indicated by tag type in field 13 (*fairy godmother*).

The other noun phrases obtain their nouns similarly from beneficiary and theme, though the last (NP) uses the preselected magicobj directly. The root's verb-child, which will be third person singular past tense, will obtain its lexical entry from the preselection labelled act. So, we get: *the fairy godmother gave the prince a magic sword.*

1.5 Two Generated Stories

Using a simple English lexicon and the grammar described above, VINCI generates fairy tales, of which the following is an intentionally short and simple example.

> *Once upon a time there was a king called Midas who lived in a castle. He was rich and vain. The king had a daughter, a princess named Marie, who was beautiful. The king warned Marie not to go out of the castle. The princess disobeyed the king. She left the castle.*
>
> *A sorcerer called Merlin lived in the woods. He was evil. The sorcerer kidnapped the princess.*
>
> *Nearby there lived a woodcutter who was named Axel. The king sought the help of the woodcutter. The woodcutter went to look for the fairy godmother. The fairy godmother passed Axel a magic sword. Axel searched for the sorcerer. The woodcutter killed the sorcerer with the magic sword. The woodcutter rescued the princess. The woodcutter and the princess got married and lived happily ever after.*

When linked to a French lexicon, morphology and syntax, VINCI generates comparable texts in French, as the following example shows:

> *Il était une fois un roi qui s'appelait Midas et qui vivait dans un beau château. Il était riche et vain. Le roi avait une fille: une princesse qui s'appelait Marie et qui était belle. Le roi interdit à Marie de quitter le château. La princesse désobéit au roi. Elle quitta le château.*
>
> *Dans la fôret il y avait un sorcier qui s'appelait Merloc. Il était méchant. Le sorcier enleva la princesse.*
>
> *Aux alentours vivait un prince qui s'appelait Coeur de Lion et qui était beau. Le roi demanda l'aide du prince. Le prince chercha la bonne fée. La bonne fée donna une épée magique au prince. Le prince chercha le sorcier. Coeur de Lion utilisa l'épée magique pour tuer le sorcier. Le prince libéra la princesse. Le prince épousa la princesse et ils eurent beaucoup d'enfants.*

Along with orthographic output, (Thomas, 2002) describes the generation of good quality prosodically controlled French oral output by linking VINCI with the MBROLA speech synthesizer (Dutoit, 2004). At this time, learner responses must still be entered orthographically.

1.6 Analysis of User Input

As well as constructing the story, VINCI may produce a series of questions to put before a learner; for example:

```
Question:  What was the name of
   the good fairy?
Expected (hidden) answer:  The
   good fairy was called Wanda.
Question:  Describe Maria.
Expected (hidden) answer:
   Maria was beautiful and
   kind.
```

In French, whose complex morphology gives scope for more varied errors, a typical question such as:

```
Question:  Où vivait le roi?
```

and a reponse from a particularly incompetent learner (one of the authors), gives rise to the following error report:

```
EXPECTED : le roi vivait dans
   un beau château
RESPONSE : la roi vivrait en
   une chapeau belle
C4 DELETE dans
S4 INSERT en
C6 ORDER C7 C6
C1 S1 la/le MORPH fém/masc
C2 S2 EXACT
C3 S3 vivrait/vivait MORPH
   cond/imparf
C5 S5 une/un MORPH fém/masc
C6 S7 belle/beau MORPH fém/masc
C7 S6 APPROX chapeau/château
```

If the learner had typed `habitait`, VINCI would have reported the change as LEX syn, *habiter* being tagged in the lexicon as a synonym for *vivre*. If he had omitted the circumflex accent on *château*, the error would have been marked as PHON, since the two forms would have been similar in sound. This is made possible by the fact that in VINCI, lexical entries may carry both orthographic and phonological information.

Output of the error analysis routines as shown above is not designed to be presented directly to a learner. However, since it is machine-generated, it is relatively easy to parse by a routine which uses it to present error analyses in a more user-friendly format. At the same time, results of each analysis may be stored and then used by a **driver program** to build a user model, and to adaptively control subsequent generation (Levison et al., 2001a). In this way, VINCI's architecture 'closes the loop' in the traditional pipeline approach to generation, in that the output of analysis and diagnosis drives the input of textual planning.

2 Enhancements and Future Work

VINCI's use of semantic input by means of functional expressions is designed to allow it to function either as an autonomous narrative generation system (cf. (Callaway and Lester, 2001a), (Bringsjord and Ferrucci, 2000) for examples) or as a story authoring environment (cf. (Umaschi and Cassell, 1997)) in which a language teacher may select or construct high-level utterance specifications, or alternatively, a learner may play with the order of a set of semantic specifications, or even add new characters with their own traits, examining in each case the texts produced. Two kinds of enhancements can be used to improve output.

2.1 Encyclopedic Enrichment

In examples shown above, descriptions are based on simple static attributes (beauty, morality, etc.). In fact, VINCI's compound attribute mechanism also allows for the expression of actions by characters. Thus, the attribute kill.monsters in the lexical entry for Prince Braveheart might cause `exists(hero)` to lead to: *Nearby there lived a prince called Braveheart, who was renowned for killing monsters.* This mechanism is also applicable to the expression of a character's thoughts and attitudes, and past background information, both narrative desiderata (Bringsjord and Ferrucci, 2000), (Robin, 1993) as well as the generation more or less complex versions of the same text (cf. (Chali, 1998)). Work is underway on mechanisms for the dynamic temporal tagging of attributes, as a story develops. For example, a learner given $50 and instructed to purchase groceries in a textual supermarket would have his or her remaining money reduced by each purchase he or she describes.

It should also be noted that the micro-world defined by means of Vinci may be fictional, as in the cases above, or based on real people and events. For example, we have performed experiments based on a database of French authors, their works, and their biographical details such as date of birth, death, etc.

2.2 Narrative Enrichment

Appropriate use of anaphoric pronous and aggregation of sentences both have a significant effect on perceptions of text quality (Callaway and Lester, 2001b). In a number of systems, both processes occur after sentences have been realized, at the level of revision, which often requires that utterances be reformulated. We propose to perform comparable operations at the level of semantic expressions. Suppose two functions: `exists(X)`, which generates *There was an X*, and `describe(X)` which generates *X was brave and handsome.* The fact that both expressions share a common argument allows for their replacement by another, say `exdesc(X)`, which aggregates the two functions to produce *There was a brave and handsome X.* Similarly, in the case of anaphoric relations, shared arguments allow for replacement of full names by pronouns. We are currently researching this. Finally, taking account of work by (Karamanis and Manurung, 2002) which shows that sharing of at least one argument characterizes a high

percentage of successive sentences in a text, it is possible to use the sequence of arguments to order a sequence of semantic expressions. Perhaps more interestingly, it may be that one of the criteria of a new paragraph is a break in the chain of shared arguments from one semantic expression to the next. The paragraph breaks in the English and French texts above, while human-constructed, respect this constraint.

References

S. Bringsjord and D.A. Ferrucci. 2000. *Artificial Intelligence and Literary Creativity: Inside the Mind of BRUTUS, a Storytelling Machine*. London, Lawrence Erlbaum.

C.B. Callaway and J.C. Lester. 2001a. Evaluating the effects of natural language generation techniques on reader satisfaction. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, pages 164–169. Edinburgh, UK.

C.B. Callaway and J.C. Lester. 2001b. Narrative prose generation. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 1241–1248. Seattle, WA.

Y. Chali. 1998. Text expansion using causal and temporal relations. In *Proceedings of the Third International Conference on Natural Language Processing and Industrial Applications*. Moncton, New Brunswick.

T. Dutoit. 2004. *The MBROLA Project*. http://tcts.fpms.ac.be/synthesis/mbrola.html.

N. Karamanis and H.M. Manurung. 2002. Stochastic text structuring using the principle of continuity. In *Proceedings of INLG-02*, pages 81–88. Columbia University.

M. Levison, G. Lessard, A-M. Danielson, and D. Merven. 2001a. From symptoms to diagnosis. In (K. Cameron, editor, *CALL – The Challenge of Change*, pages 53–59. Elm Bank, Exeter.

M. Levison, G. Lessard, B. Gottesman, and M. Stringer. 2001b. Semantic expressions: An experiment. *Working paper, found at: http://www.cs.queensu.ca/CompLing/semanticexpressions.html*.

W. Menzel and I. Schroeder. 1998. Constraint-based diagnosis for intelligent language tutoring systems. In *Proc. IT&KNOWS, XV. IFIP World Computer Congress*, pages 484–497. Vienna/Budapest.

L.N. Michaud. 2002. *Modeling User Interlanguage in a Second Language Tutoring System for Deaf Users of American Sign Language*. PhD Dissertation, Department of Computer and Information Sciences, University of Delaware.

J. Robin. 1993. A revision-based generation architecture for reporting facts in their historical context. In M. Zock H. Horecek, editor, *New Concepts in Natural Language Generation*, pages 238–268. Pinter, London.

C. Thomas. 2002. *A Prosodic Transcription Method for Natural Language Generation*. MSc Thesis, Queen's University, Kingston.

M. Umaschi and J. Cassell. 1997. Storytelling systems: Constructing the innerface of the interface. In *Cognitive Technologies Proceedings '97*, pages 98–108. IEEE.