Sistemas Tutores Inteligentes: El submódulo de Lenguaje Natural

Zulma Cataldi,
LIEMA - Laboratorio de Informática Educativa y Medios Audiovisuales. Facultad de Ingeniería. UBA.
Facultad Regional Buenos Aires. Universidad Tecnológica Nacional..
liema@fi.uba.ar

Fernando Salgueiro,
LIEMA - Laboratorio de Informática Educativa y Medios Audiovisuales. Facultad de Ingeniería. UBA.
fsalgueiro@fi.uba.ar

Fernando Javier Lage[1]
LIEMA - Laboratorio de Informática Educativa y Medios Audiovisuales. Facultad de Ingeniería. UBA.
Facultad Regional Buenos Aires. Universidad Tecnológica Nacional.
flage@fi.uba.ar

Abstract

In previous publications the low yield of the students in its partial and final evaluations had been detected. By that reason thought about supporting the actual classes with intelligent tutoring. The basis for the design of intelligent tutorial system beginning by the redesign of the modules that compose it were done. Now, one looks for to incorporate a natural language module that is independent of the rest of components taking like example the contents of basic programming.

Keywords: Intelligent Tutoring Systems, natural language.

Resumen

En publicaciones previas se había detectado el bajo rendimiento de los estudiantes en sus evaluaciones parciales y finales. Por ese motivo se pensó en apoyar las clases presenciales con tutorizado inteligente. En las primeras publicaciones se sentaron las bases para el diseño del sistema tutor inteligente comenzando por el rediseño de los módulos que componen el nuevo modelo. Ahora, se busca incorporar un módulo de lenguaje natural que sea independiente del resto tomando como ejemplo los contenidos de programación básica.

Palabras Clave: Sistemas Tutores Inteligentes, lenguaje natural.

[1] Esta comunicación ha sido desarrollada en el marco del Convenio FI-UBA y UTN-FRBA. Proyecto *Modelado del tutor basado en redes neuronales para un Sistema Tutor Inteligente* del Programa de Incentivos 2007-2008.

1. INTRODUCCIÓN

Se busca definir las interfaces para poder incorporar un submódulo de lenguaje natural a la estructura del módulo del tutor en un STI. Para ello, se utilizarán los lineamientos de ya presentado en trabajo previos [1] y los aportes de Di Eugenio [2], quien plantea que una de las claves para solucionar el abismo actual que existe entre los sistemas tutores inteligentes de esta generación y sus contrapartes humanas reside en la forma en que el primero interactúa con el humano por lo que realiza un análisis completo de la mejora que puede garantizar el uso de lenguaje natural en los STI. Este tema fue tratado por otros autores, como Evens *et al.* [3] quien aportó el resultado de su investigación al proyecto del tutor *CircSiM*. Freedman [4] y Graesser *et al.* [5] entre otros, realizaron notables aportes acerca de los beneficios que conlleva la incorporación de lenguaje natural en los STI. Este módulo está relacionado con el planificador de la lección y genera las salidas del sistema con lo cual se consigue tener un sistema tutor inteligente con capacidades de lenguaje natural. Pero, estas funciones están intrínsecamente enlazadas con el sistema del planificador y no se pueden ver cómo módulos realmente separados. En este caso se propone, el análisis del módulo de lenguaje natural como una *"caja negra",* a fin de definir las funcionalidades y sus interfaces con los demás módulos para hacerlo completamente independiente de éstos. Algunos tutores que en la actualidad utilizan el lenguaje natural como en forma integrada son: *Circle* y *AutoTutor System* [5] y se están desarrollando mejoras en tutores más antiguos para brindarles las capacidades del lenguaje natural [5][6][7].

2. FUNDAMENTACION

Se busca obtener un submódulo independiente de lenguaje natural con lo que los primeros prototipos tendrán capacidades limitadas pero en la medida que el desarrollo avance se podrán obtener módulos más potentes y completos con solo modificar el submódulo en cuestión.
La modularización permite comparar el desempeño de cada uno de los módulos implementados por separado, con lo que el módulo de lenguaje natural se puede cambiar para idiomas diferentes al realizar la sesión de enseñanza, logrando de este modo un tutor multilenguaje, sin tener que modificar el resto de los módulos, solo agregando la información en el módulo de conocimientos en los distintos idiomas. En el modelo más tradicional, muchas de las frases son de tipo "*hardcoded*"[2] es decir pre-armadas dentro del planeador, quien es el encargado de producir la salida del sistema, tal como en las primeras versiones del tutor *CircSim* [8].
Si se analiza el módulo de lenguaje natural como una caja blanca, se encontrará que el proceso más complejo que debe realizar es el denominado: "*alegación de sentencias*" de Reiter y Dale [9] quienes lo definen como el proceso de agrupación de sentencias a fin de definir oraciones. Esto se puede lograr obteniendo información acerca de la sintaxis del lenguaje humano de una base de datos interna, de la que se obtendrá el vocabulario para armar las oraciones, el conjunto de reglas que determinan las combinaciones que están permitidas en el lenguaje, las reglas de agregación, etc. Estos aspectos fueron estudiados por, Huang y Fiedler, Shaw [10] y Scott y Sieckenius de Souza [11] entre otros. Independientemente del algoritmo que se utilice para la construcción lingüística, la independencia del módulo y el uso de la interface permitirá que el STI pueda obtener la capacidad de lenguaje natural, desde la simple generación de frases aisladas, hasta el armado de párrafos relacionados entre si. Debido a ello, se definirá primero la forma de representación de la interface entre los submódulos del módulo tutor y el que corresponde al lenguaje natural. Esta consistirá en una serie de funciones básicas con sus parámetros, que permitirán realizar la interacción con el usuario final del sistema en ambos sentidos al submódulo de lenguaje natural, es decir, desde el sistema hacia el usuario (como el *output* o salida del sistema) ó desde el usuario que

[2] Vocablo que se usa Para de definir valores o parámetros que se encuentran fijos en el código fuente y no pueden ser cambiados una vez compilado el programa, restándole así flexibilidad.

responde a alguna de las preguntas del sistema (como uno de los múltiples *inputs* ó entradas del sistema). Para ello, se deberán definir previamente los objetivos del módulo de lenguaje natural; que son:

- *Realizar la salida de datos de manera similar se esperaría de un tutor humano*: La salida de los datos se puede realizar de varias maneras, siendo éstas completamente independientes de los datos que pertenecen a la interface, que es la encargada de mostrar los contenidos multimediales de las lecciones ó sesiones de tutelado y de presentar una pantalla homogénea para navegar a través del sistema. Por ejemplo, la salida puede ser en forma de texto coloquial en el idioma nativo del usuario, es decir podría ser español incluyendo los términos técnicos asociados al vocabulario correspondiente a un alumno de nivel inicial que va a realizar un primer curso o en forma de voz, aplicando las mismas consideraciones que para el texto. Estas dos opciones no son mutuamente excluyentes y podrían utilizarse ambas, por medio de aplicaciones de conversión de terceros tales como Microsoft Agent[3].
- *Entregar los datos procesados a la interface:* Es decir, limpios de todas construcciones gramaticales utilizadas por el alumno como medio de construcción de respuestas u otras preguntas que se le deseen hacer al sistema. Se busca que el submódulo de lenguaje natural se encargue de procesar la entrada de los datos de la interface, que representa la interacción con el usuario obteniendo de ésta todos los conceptos filtrados, es decir, libres de las construcciones gramaticales que los soporta en el lenguaje natural. Estos datos luego serán enviados desde la interface para su procesamiento hacia los otros submódulos.
- *Funcionar como entrada de los datos también en lenguaje natural:* Está pensado para que el alumno pueda desenvolverse con mayor facilidad al responder los cuestionarios del sistema ó ante cualquier otra exposición de datos que requiera una interacción del alumno, siempre guiada por parte del sistema. Este sistema puede funcionar tanto en forma escrita como oral, utilizando las herramientas adecuadas[4].

Con estas tres funciones básicas y una vez definida la interface, se pueden establecer las bases para construir un submódulo de lenguaje natural que sea independiente del resto del tutor. Es más, se puede abstraer aún más la estructura para lograr un submódulo completamente separado, si se desea hacer un sistema basado en tecnologías del tipo *Web services*, en las que los módulos principales se encuentren en un servidor remoto y tanto la interface como el submódulo de lenguaje natural se pueden encontrar en el cliente. De esta manera, el proceso de transformación de las entradas, se distribuirá en el cliente y se minimizará la cantidad de información a ser transferida por la red, mejorando así el rendimiento global de todo el sistema.

3. LOS SISTEMAS TUTORES INTELIGENTES

Los sistemas tutores inteligentes (STI) comenzaron a desarrollarse en los años 80 y fueron diseñados con la idea de impartir conocimiento con base en alguna forma de inteligencia para guiar al estudiante en el proceso de aprendizaje [12][13]. Su propósito es presentar un comportamiento similar al de un tutor humano, que se adapte a las necesidades del estudiante, identificando la forma en que el mismo resuelve un problema para poder brindarle ayuda cuando cometa errores. Un tutor inteligente, por lo tanto: *"es un sistema de software que utiliza técnicas de inteligencia artificial (IA) para representar el conocimiento e interactúa con los estudiantes para enseñárselo"* [14]. Wolf [15] define los STI como: *"sistemas que modelan la enseñanza, el aprendizaje, la comunicación y el dominio del conocimiento del especialista y el entendimiento del estudiante*

[3] *Microsoft Agent* es una tecnología que presenta una interface conversacional que puede, desde aceptar comandos verbales, hasta hablar (por medio de motores de voz en varios idiomas o audio grabado).Puede ser embebido dentro de páginas Web, aplicaciones, etc. para mejorar y humanizar la interacción de éstas con el usuario (Información disponible en su sitio Web: http://www.microsoft.com/msagent/). Los motores de voz pueden ser los provistos por Microsoft o por terceros, de los cuales vale la pena resaltar la empresa *lernout & hauspie*, disponible en su sitio Web: http://www.lhs.com/).

[4] Microsoft Agent ó alguna otra similar, como Dragon Naturally Speaking Voice Recognition de DragonTalk Información disponible en su sitio Web: http://www.dragontalk.com/. Consultada el 24/07/2007

sobre ese dominio". Giraffa [16] los delimita como: *"un sistema que incorpora técnicas de IA (Inteligencia Artificial) a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa"*. En la Figura 1 se puede ver la estructura general de un STI donde el sub-módulo generador de lenguaje natural se encuentra dentro del módulo del tutor y es el encargado de interactuar directamente con la interfaz, transmitiendo los conocimientos, las respuestas y las repreguntas en forma de lenguaje (hablado, escrito u de otra manera) para que el estudiante se sienta mas cómodo utilizando el sistema.

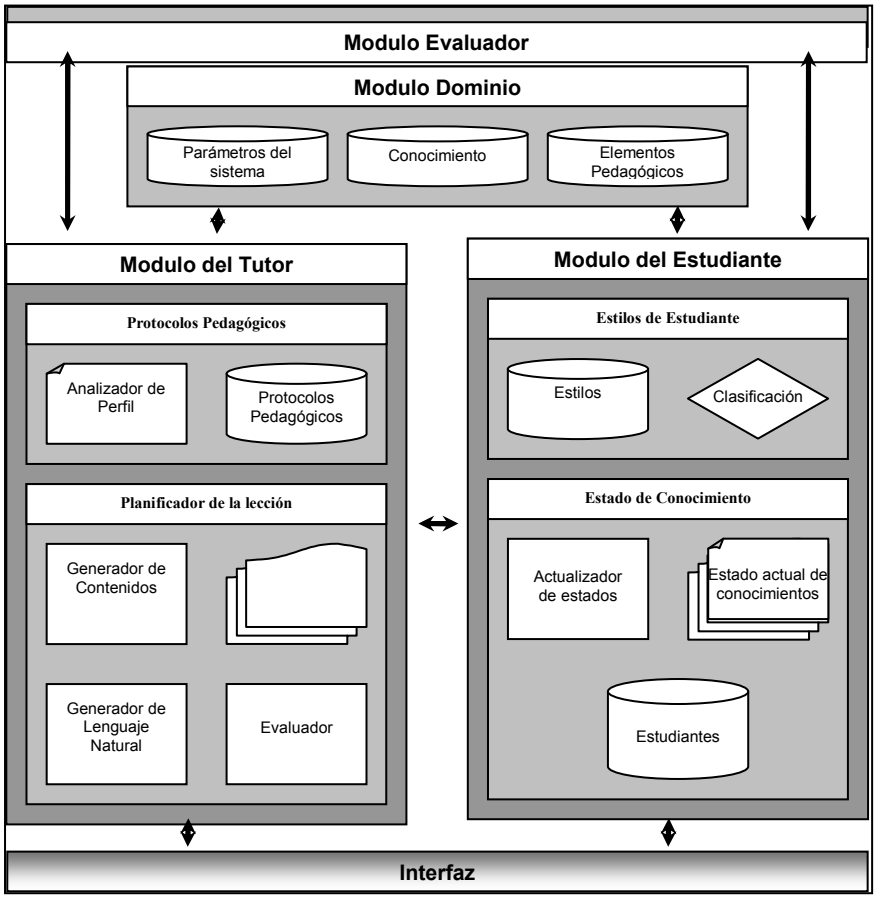

Figura 1: Modelo detallado de un Sistema Tutor Inteligente

Los STI permiten la emulación de un tutor humano para determinar *qué enseñar, cómo enseñar y a quién enseñar* a través de un *módulo del dominio*: que define el dominio del conocimiento, *un módulo del estudiante*: que es capaz de definir el conocimiento del estudiante en cada punto durante la sesión de trabajo, un *módulo del tutor*: que genera las interacciones de aprendizaje basadas en las discrepancias entre el especialista y el estudiante y finalmente *la interface*[1] con el usuario: que permite la interacción del estudiante con un STI de una manera eficiente aplicando el conocimiento sobre *cómo presentar* los contenidos.

4. LAS INTERFACES DEL SUBMÓDULO DE LENGUAJE NATURAL

Para que el sistema funcione se definirá analizarán los tipos de interacciones posibles con el módulo de lenguaje natural:

- *Interacciones de tipo informativas:* A través de estas interacciones, el módulo tutor podrá agregar datos de interés, subrayar las relaciones entre los conceptos y aclarar las dudas particulares a lo largo del dominio en este caso de la programación básica. Este tipo de interacciones surgen cuando el módulo tutor está realizando una sesión de tutelado con el usuario pero no durante los interrogatorios.
- *Interacciones de tipo pregunta:* Se deben utilizar cuando el sistema requiere hacer una pregunta directamente al usuario, relacionada con algún concepto, para verificar que el usuario puede manejarlo o para agregarlo a la pila[5] de temas desconocidos cuando éste falla en responder a la pregunta de forma satisfactoria.

[5] Una pila (stack en inglés) es una estructura de datos de tipo LIFO (del inglés Last In First Out, último en entrar, primero en salir)

- *Interacciones de tipo respuesta:* Una vez planteada la pregunta por el sistema, se pueden presentar varias situaciones de respuesta por parte del alumno, como las planteadas por Yujian [17], y otras nuevas que para poder ser implementadas en el modelo propuesto dependiendo del tipo de respuesta suministrada por el alumno, el sistema responderá de la siguiente manera:
  - *Si la respuesta es Correcta:* El sistema se encargará de hacerle saber al usuario que su respuesta es correcta, y actualizará la pila de objetivos de la lección con respecto al nuevo estado. Este es el caso más simple, ya que el sistema sólo debe comunicarle al usuario que está en lo correcto.
  - *Si la respuesta es "Near Miss"* [18]: es decir, cundo la respuesta no es la deseada pero es pedagógicamente útil. Un ejemplo puede ser la siguiente interacción:

```
P: ¿Cuáles son los tipos ordinales en Pascal?
R: Los numéricos son tipos ordinales.
```

    La respuesta es incorrecta, porque si bien los tipos enteros (*integer, byte, longint, etc.*) son ordinales, los numéricos *reales* no lo son, por lo tanto el sistema deberá decidir si debe explicarle esto al alumno, si debe utilizar alguna pista para ayudarlo a dilucidar la respuesta correcta ó si debe darle la respuesta directamente para que éste pueda reflexionar e incorporar el concepto que se intenta explicar.
  - *Si la respuesta es "no sé":* Dependiendo del caso puede pasar a una interacción del tipo pista, pero si el error es muy grave ó conceptual esto puede derivar hacia una respuesta interacción del tipo informativa. En ambos casos no se involucra una interacción de tipo respuesta.
  - *Si la respuesta es un error conceptual* [15]: También se puede optar por una respuesta interacción del tipo informativa pero que no involucra una interacción de tipo respuesta.
  - *Si la respuesta es de otro tipo:* El sistema deberá responderla con una interacción del tipo informativa, para simplificar el esquema.

En cualquiera de los casos el sistema deberá poder armar sus respuestas clasificando forma en la que tratará cada uno de los tipos de respuesta.

5. ANÁLISIS DE LOS TIPOS DE INTERACCIONES

Existen diversas categorías de interacciones:

Interacciones de tipo Pista: Con base en el modelo de pistas propuesto por Hume *et al.* [7] existen distintas formas de presentar las pistas: a través de pregunta, como afirmaciones, relacionándolas con otros conceptos, aunque todas las formas para presentar una pista, dan la misma información, independiente del modo en que se presente. Un ejemplo de este caso puede ser el siguiente:

```
P: ¿Cuáles son los tipos ordinales en Pascal?
R: Los numéricos son tipos ordinales.
```

Si el sistema decide dar una pista, podría realizarla de la siguiente manera:

```
Pista: Los números reales son del tipo numérico, pero: ¿son ordinales?
Pista: Piense en los números reales.
Pista: ¿Cuál es la característica fundamental de los ordinales?.
```

Las dos primeras son dos formas de mostrar la misma información: como pregunta y como afirmación. La última, requiere que el planificador de la lección agregue a la pila de objetivos uno nuevo que es: "*Características de los ordinales*", que deberá ser respondido de forma satisfactoria antes de continuar con la pregunta original que era: *¿Cuáles son los tipos ordinales en Pascal?*. Se

dejará al submódulo de lenguaje natural la forma en que presentarán las pistas, es decir, si serán como pregunta o afirmación.

– *Interacciones de tipo Control:* Se utilizan cuando se desea que el sistema presente las respuestas de una manera en particular, de forma forzada como pregunta, de forma forzada como afirmación y proporcionando estructuras de control para que se puedan presentar varias interacciones al mismo tiempo, lo cual lleva a la próxima clasificación:

– *Interacciones de tipo Mixtas:* En este caso el sistema requiere de una mezcla de respuestas para mejorar la interacción con él usuario. Un ejemplo podría ser el siguiente:

```
P: ¿Cuáles son los tipos ordinales en Pascal?
R: Los numéricos son tipos ordinales.
```

La respuesta del sistema puede ser una combinación de dos respuestas: una de tipo interacción informativa, seguida por una interacción de tipo pregunta, de la siguiente manera:

```
Los tipos ordinales se caracterizan en que dado un elemento cualquiera del conjunto, se
puede conocer cuál es su predecesor (salvo del primero) y el sucesor (salvo del
ultimo). Ahora: ¿Todos los tipos numéricos poseen esta característica?.
```

Aquí se ve que el sistema da la definición del concepto y luego se encarga de generar una nueva pregunta, con lo que se tienen dos interacciones, que son parte de la misma respuesta del sistema a la entrada provista por el usuario.

En la Figura 2 se transcribe un ejemplo de interacción entre un alumno y un tutor humano. A partir de una serie de interacciones reales como éstas surge la clasificación de las respuestas del tutor dentro del modelo anteriormente planteado.

```
P: ¿Cuáles son los tipos ordinales en Pascal?
A: Los numéricos son tipos ordinales.
P: No todos los tipos numéricos son ordinales. Puede decirme cuáles son los tipos numéricos que
    conoce?.
A: Los tipos numéricos son Byte, Integer, Longint,  Real, Extended, etc.
P: Si, pero de una forma más teórica, todos los tipos numéricos que usted mencionó pueden agruparse
    en dos categorías.
A: Enteros y Reales.
P: Si. Ahora, retomando la pregunta original, esos dos tipos numéricos son ordinales?.
A: Si.
P: No, los tipos numéricos reales no son ordinales, ¿Sabe usted por qué?
A: No.
P: Piense en lo siguiente: ¿Cuál es la característica fundamental de los ordinales?.
A: Que tienen un orden.
P: Si, Esa afirmación está incompleta. Plantee mejor qué significa que los ordinales tienen un
    orden.
A: Que un elemento es mayor que el otro.
P: Eso no es correcto en todos los casos, puede completar la idea.
A: No, no puedo.
P: Un conjunto ordinal es aquel en el cual el compilador puede identificar el predecesor de un
    elemento iésimo del conjunto (menos el primero) y un sucesor del elemento iésimo (menos del
    último).
P: Entonces, podría decirme por qué los reales no son un conjunto ordinal.
A: Porque dado un elemento, el compilador no puede calcular cuál es el siguiente inmediato.
P: Correcto.
```

Figura 2: Modelo de interacciones entre un estudiante y un tutor humano.

También se deben definir las funciones que caracterizaran a la interface entre el submódulo de lenguaje natural y el resto del módulo del tutor, es decir:

– *Interacciones de tipo informativas*: `I_Information (Tipo, Texto)`

Este tipo de interacción es la más simple de todas y brinda la definición del concepto a explicar desde el módulo del dominio y lo muestra en pantalla. Esta es una función del submódulo de lenguaje natural que permite mostrar el texto que se encuentra en el campo de texto. El parámetro opcional "`Tipo`" surge porque el generador de lenguaje natural no está limitado a producir salidas de manera entendibles sólo en forma de texto (escrito u oral) para el usuario humano. Esta situación se

puede aclarar a través de un ejemplo suponiéndose un STI cuyo dominio de aplicación es la enseñanza de un instrumento musical, por ejemplo: el piano. El Sistema Tutor podría estar intentando mostrar una partitura que pretende que el alumno represente en el instrumento (a través de la salida del planificador de la lección hacia el modulo de lenguaje natural), mientras que el estudiante puede tocar en un instrumento real y el submódulo de lenguaje natural deberá decodificar las notas por tonalidades y entregarlas al planificador codificadas tal como se ve en la Figura 3.

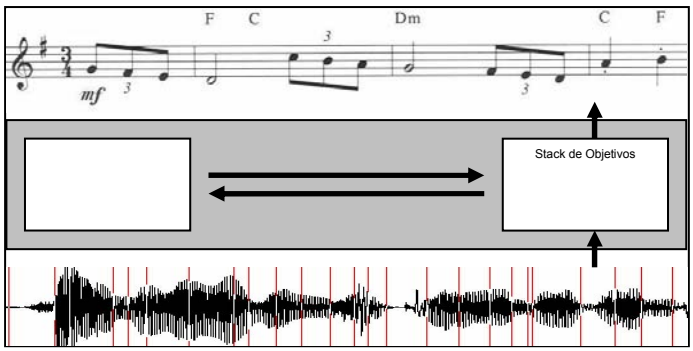

Figura 3: Interacción no convencional para producir salidas que no son texto y procesar entradas que no lo son.[1]

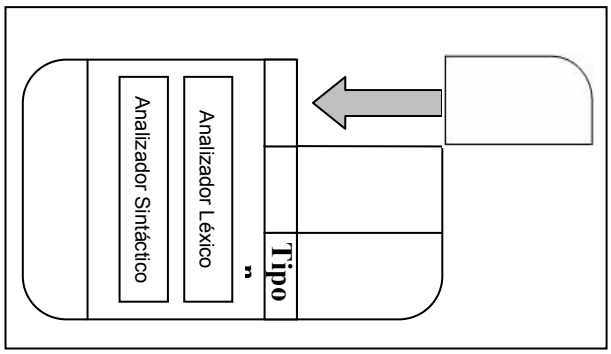

Figura 4: Submódulo de lenguaje natural como un núcleo central de funcionalidades y un conjunto de componentes adicionales.

Este tipo de interacciones específicas para algunos STI, no están limitadas por la definición de la interface, están permitidas y lo único es que se requiere un submódulo de lenguaje natural que las soporte, es decir, que cumpla con las reglas generales de la interface y que luego utilice estos `Tipos` que se dejan sin definición para casos particulares. Esto brinda la flexibilidad para que no se requiera modificar completamente la estructura del submódulo de lenguaje natural en los STI, sino que se pueda optar por una estructura de "*componentes*" para el submódulo de Lenguaje Natural donde cada uno de estos "*componentes*" se referencia con el número que indica el parámetro "`Tipo`" como se muestra en la Figura 4.

- *Interacciones de tipo pregunta:* `I_Question (Tipo, concepto`$_0$`,concepto`$_1$`,…, concepto`$_n$`)`

  Este es un método secundario que permitirá al tutor obtener los resultados de una sesión pedagógica si se utilizan protocolos de tutoría del tipo magistrales, mientras que será el método principal si se utilizan protocolos de tutoría del tipo socrático [1]. En este tipo de interacción se busca que el submódulo de lenguaje natural genere construcciones lingüísticas compatibles con las preguntas. Éstas se podrían categorizar teniendo en cuenta los modelos registrados durante las preguntas posibles a realizar por el sistema en los interrogatorios y las evaluaciones de la asignatura:

  – *Preguntas de conceptos:* Estas preguntas apuntan a que el usuario del sistema entregue la definición de un concepto en particular. Éstas son las preguntas más comunes en las sesiones socráticas y probablemente no se encuentren en las sesiones magistrales. Ejemplos de estas preguntas son las siguientes:

Pregunta real	Primitiva de Interacción
¿Cuáles son los tipos ordinales en Pascal?	I Question ("concepto", "tipos ordinales")
¿Cómo se pueden clasificar los números enteros?	I_Question ("concepto", "clasificación de enteros")

– *Preguntas de relación de conceptos:* Se intenta que el alumno relacione dos conceptos que poseen una o más características en común. Este tipo de preguntas ayuda al planificador como un indicador de que el alumno puede integrar varios conceptos. Son ejemplo de este tipo de preguntas:

Pregunta real	Primitiva de Interacción
¿Qué tienen en común la codificación "módulo	I_Question("relación", "módulo y signo",

y signo" y el binario puro?	"binario")
¿Qué características comparten los Integer, Longint y Byte?	I_Question("concepto","Integer","Longint","Byte")

–*Preguntas de ejercicios:* Con estas preguntas que si bien pueden suponerse como de concepto, se espera del alumno un resultado conciso, pero no necesariamente definiciones, ya que pueden ser resultados numéricos, códigos fuentes, etc. También se pueden enmarcar en esta categoría las preguntas de respuesta múltiple y las de "*verdadero o falso*", que pueden formar parte de exámenes o parte de la lección, según pertenezcan al planificador o no. Algunos ejemplos son:

Pregunta real	Primitiva de Interacción
¿Qué valor representa el número 179 decimal en binario?	I_Question ("ejercicio", "decimal a binario", "179")
¿Cuántas iteraciones máximas debe hacer para ordenar por el método de Burbujeo un vector de 10 posiciones?	I Question ("ejercicio", "ordenamiento", "burbujeo", "iteraciones",)

–*Pregunta de definiciones:* Son aquellas en las que el sistema espera que el alumno proporcione la definición exacta de un concepto y no solo alguna de sus características (para ello, están las preguntas de *concepto*). Esto podría enmarcarse como preguntas de exámenes ó similares para que un tutor luego las analice. Este tutor puede ser un tutor humano: Las preguntas de este tipo quedaran almacenadas en el STI, quien se encarga de procesar las respuestas como si este tipo de preguntas fuera alguna de las anteriores. Ejemplos de éstas serían:

Pregunta real	Primitiva de Interacción
Defina las partes de la CPU	I_Question ("definición", "partes CPU")
¿Qué entiende por recursividad?	I_Question ("definición", "recursividad")

–*Otras preguntas:* En esta categoría se engloban todas las preguntas que no responden a ninguna de las mencionadas anteriormente. Se pueden utilizar para mantener un diálogo fluido entre las distintas preguntas generales del sistema, cuando el planificador agrega a la cola[6] conceptos nuevos a explicar al alumno.

Pregunta real	Primitiva de Interacción
¿Sabe usted por qué?	I Question ("otras", "¿Sabe usted por que?")
Plantee mejor qué significa que los ordinales tienen un orden.	I Question ("otras", "Plantee mejor que significa que los ordinales tienen un orden.")

La clasificación de las preguntas tiene como objetivo facilitar el "*parsing*" (analizador sintáctico) de la respuesta del alumno, es decir, si el sistema tiene que interpretar una respuesta sin conocer el contexto de la pregunta, puede ser mucho más difícil obtener datos concisos. El tipo de pregunta le entrega al submódulo de lenguaje natural el contexto en el cual deberá buscar la respuesta de la próxima respuesta del alumno. En caso de que la pregunta se encuentre dentro de un bloque generado por las interacciones de control (`I_control ("comienzo")` y `I_control ("finalización")`), el submódulo de lenguaje natural deberá mantener el contexto de la última de las `I_Question` que reciba. En la mayoría de los casos generales, es muy poco probable que el planificador entregue más de una pregunta dentro `I_Question` del mismo bloque, ya que esto representaría que el alumno debería responder dos preguntas ó más dentro de la misma respuesta, aunque esta interacción no está prohibida en este protocolo. Un ejemplo podría ser:

Pregunta real	Primitiva de Interacción
P: Un conjunto de elementos Ordinal es aquel que el compilador puede identificar el predecesor de un elemento iésimo del conjunto (menos el primero) y un sucesor del elemento iésimo (menos del último). Entonces, podría decirme por qué los reales no son un conjunto ordinal?.	I_control ("Comienzo") I_Information ("Conjunto ordinal") I_Question ("concepto", "reales", "conjunto ordinal") I_control ("finalización")
Si, Esa afirmación está incompleta. Plantee mejor qué significa que los ordinales tienen un orden?.	I_control ("Comienzo") I response ("Si") I Question ("otras", "Plantee mejor que significa que los ordinales tienen un orden.") I_control ("finalización")

[6] Una cola (queue en inglés) es una estructura de datos de tipo FIFO (del inglés Firs In First Out, primero en entrar, primero en salir)

- *Interacciones de tipo respuesta:* `I_Response (Tipo_Respuesta, texto)`

Simplemente son el resultado de una respuesta del sistema quien logra responder básicamente en forma afirmativa o negativa las interacciones con los usuarios, pero se puede agregar un tipo de respuesta, que se denominará neutra, para representar las situaciones incompletas, donde la respuesta dada es solo una parte de la respuesta esperada por el sistema, pero no es la respuesta completa esperada. En todos los casos, el texto que acompaña a este tipo de interacción es opcional, el planificador puede proveer el resultado exacto que se desea agregar en la respuesta. Esto también puede ser emulado utilizando interacciones de control para generar un bloque, pero independientemente de que se encuentre el parámetro opcional o no, la clasificación de los tipos de respuesta es la siguiente:

–*Positiva*: Cuando el sistema recibe la respuesta que desea para alguno de los conceptos que han sido preguntados y se encuentran en la pila del planificador de la lección. En ese momento, el planificador la retira de la pila, pero este proceso es transparente para la interface.

Pregunta real	Primitiva de Interacción
`Si (el tipo más sencillo de esta categoría)` Es correcto, los enteros son ordinales.	`I_Response ("positiva")` `I_Question ("positiva", "Es correcto, los enteros son oridnales.")`

–*Negativa*: Cuando el sistema recibe una respuesta errónea y no desea generar pistas u otro tipo de ayuda para corregir al usuario, muestra una respuesta del tipo negativa. En la mayoría de los casos, se encontrará dentro de un bloque, ya que el sistema no solo debe limitarse a informar el estado de la respuesta del usuario, sino que deberá modificar los conocimientos del usuario para que este adquiera el concepto, si ésta fuera incorrecta, es decir, por la propia definición de STI que cautele la apropiación de conceptos. El protocolo de la interface es independiente de que la respuesta negativa se encuentre en un bloque ó no. Si no se encontrara en un bloque, el submódulo de lenguaje natural deberá mostrar la respuesta aunque este caso no debería ser la salida del planificador de la lección. Es decir:

Pregunta real	Primitiva de Interacción
`No (el tipo mas sencillo de esta categoría)` Eso no es cierto, no todos los formatos numéricos son ordinales.	`I_Response ("negativa")` `I_Question ("negativa", "Eso no es cierto, no todos los formatos numéricos son ordinales.")`

–*Neutra*: Cuando el sistema responde al usuario de una manera en la que la respuesta a la interacción no es completa, ó la esperada por el sistema. Además del caso de las respuestas incompletas a preguntas respecto de una serie de conceptos, se puede agregar este caso a las situaciones en las que el planificador no puede comprender correctamente la respuesta del usuario y por lo tanto debe reformular la pregunta para que el usuario la entienda correctamente.

- *Interacciones de tipo Pista:* `I_Hint (texto)`

Las interacciones del tipo "*pistas*", son interacciones similares a las respuestas, pero por su valor pedagógico, se ha decidido desglosarlas en una nueva categoría. La categorización utilizada para los distintos tipos de pistas es similar a la que plantea Hume [7] en su tesis doctoral, que engloba a las de Glass [18] y las de Wolf [15] donde el tratamiento de las pistas se puede ver como una combinación de las primitivas básicas planteadas en este modelo. Cada uno de los casos de las pistas se puede aclarar en forma particular:

–*Si la respuesta es del tipo Correcta:* No requiere ninguna aclaración en especial genera una interacción del tipo `I_Response ("positiva")` simple y de la manera explicada anteriormente.

- *Si la respuesta es del tipo "Near Miss*" [18]: Este es el primer tipo de respuesta que puede contener una verdadera interacción del tipo `I_Hint`. El alumno no acertó correctamente la respuesta, pero demuestra tener conocimientos teóricos sobre el tema, el sistema deberá guiarlo como se explicó anteriormente, generando una nueva pregunta orientativa del camino mental a seguir. Analizando este caso como una interacción de tipo mixta, deberá tener una forma como la siguiente:
  - *Si la respuesta es del tipo "no se":* En este caso en particular, el sistema puede optar por seguir entregándole más pistas para que el estudiante logre encontrar la solución por sus propios medios ó puede desistir a las interacciones de tipo pista para entregarle la definición del concepto pedido o directamente la respuesta a la pregunta.
  - *Si la respuesta es del tipo error conceptual* [15]: Si se encuentra con un error conceptual, el planificador debe encargarse de corregirlo, se finalizará con las interacciones de tipo pista y se continuara con interacciones de tipo `I_information`.
  - *Si la respuesta es de otro tipo:* El planificador debe encontrar la solución correcta a una respuesta no categorizada, puede optar por reformular la pregunta original (`I_Question`), por entregarle al usuario estudiante otra pista que lo acerque nuevamente al camino por el cual el STI lo desea guiar (`I_Hint`) ó por último puede desistir y entregarle la definición ó la respuesta a la pregunta que dio origen a esta interacción en primer lugar (`I_information`).

- *Interacciones de tipo Control:* `I_control (Tipo)`

Todas las interacciones de control surgen de la necesidad de poder armar interacciones complejas como las que plantean Reiter y Dale [9] y su propósito principal es el de la construcción lingüística. Varias respuestas pueden estar agrupadas para formar oraciones y estas oraciones pueden unirse luego para formar párrafos. Por la naturaleza de la interface, cada una de las llamadas particulares de estas funciones generará por sí misma una contracción lingüística correspondiente a una oración o más. Pero, una misma respuesta del sistema puede contener una `I_Response` (como resultado directo de la ultima interacción con el usuario), seguido directamente por una `I_Information` (para aclarar algún concepto errado o para ampliar el conocimiento sobre un tema en particular), y por último una `I_Question` para verificar que el usuario ha entendido la nueva información que se le presento y que puede seguir adelante con el proceso de aprendizaje. El ejemplo citado en el párrafo anterior es solo un caso de las innumerables combinaciones que pueden resultar de la interacción normal con el sistema. Por eso, es necesario definir los siguientes tipos de interacciones de control:

- *Comienzo*: Indica el comienzo de una interacción que agrupa a más de una llamada a una función. Es similar al concepto de bloque de código en cualquier compilador y con esto, el submódulo de lenguaje natural se encargará de generar la construcción lingüística equivalente a la función que pasa a través de la interface, pero en lugar de enviarla directamente hacia el usuario del sistema (en el caso más directo sería la salida por otra interface fuera del módulo del tutor), la mantiene para agregarle todas las construcciones lingüísticas que se le indiquen a continuación, hasta recibir una interacción de control que le indique que puede despachar todos los "*mensajes*" que tiene en la cola de espera como si en realidad se tratase de un único mensaje, pero mas extenso.
- *Finalización*: Indica el final del bloque de interacción y avisa al submódulo de lenguaje natural que ya puede enviar todos los mensajes que mantiene encolados hacia su interface en la forma procesada de párrafo (o la construcción lingüística equivalente). Se podría agregar que no es necesario indicarle al submódulo de lenguaje natural que encole los mensajes que deben ser entregados a su interface de forma individual (como lo son cualquier `I_Response`, `I_Hint`, `I_Information`, `I_Question`) y por lo tanto no es necesario comenzar un bloque lingüístico

con `I_Control ("comienzo")`, asimismo, si esto se realiza, no se debería obtener un resultado diferente.

Luego, no se requieren más interacciones de control sobre la interface, pero el submódulo de lenguaje natural requiere mantener, como ya he explicado antes, el contexto sobre el que espera la respuesta, y eso lo obtiene, básicamente a partir de la interacción de respuesta del alumno ó puede obtenerlo también a partir de las interacciones de tipo pistas.

En el caso de que existan varias interacciones que requieran mantener contexto, la complejidad del módulo puede forzar a guardar sólo la última interacción de preguntar ó puede guardar todas y tomar la respuesta que mejor se adecue a alguno de los contextos que mantiene almacenados. Una vez finalizada la interacción inmediata puede eliminar el contexto (ó los contextos) almacenado, ya que la nueva interacción le corresponde al sistema. Otra opción sería que sólo borrara los contextos de las preguntas al recibir nuevas preguntas ó en el momento de recibir nuevas instrucciones de control, de esta forma se pueden aceptar varias respuestas individuales del alumno (aunque se viole el principio de que el alumno no comience ninguna interacción en el sistema, sino que simplemente continúe las interacciones que comienza el STI en una sesión pedagógica, aunque si se incluye este caso quedaría definida una interface mucho más flexible).

Pregunta real	Primitiva de Interacción
Un conjunto de elementos del tipo Ordinal es aquel que el compilador puede identificar el predecesor de un elemento iesimo del conjunto (menos el primero) y un sucesor del elemento iesimo (menos del ultimo).	I control ("Comienzo") I Information ("Conjunto ordinal") I Question ("concepto", "reales", "conjunto ordinal") I_control ("finalización")

– *Interacciones de tipo Mixtas:*

```
if  (input type == I control )  {
next_input()
         while (input type != I_control) {
switch (input) {
case I Information: next input()
encode text NL(Input) break;
case I Response: next input()
encode_text_NL(Input) break;
case I_Question: push (context)
encode_text_NL(Input) next_input() break;
case I_Hint: push (context)
encode text NL(Input) next input() break;
case I Control: break;
}
Output(processed_text)
Else {
switch (input) {
case I Information: next input()
encode text NL(Input) break;
case I Response: next input()
encode_text_NL(Input) break;
case I_Question: push (context)
encode_text_NL(Input) next_input() break;
case I Hint: push (context)
encode text NL(Input) next input() break;
case I Control: break;
}
Output(processed_text)
}
```

```
       ready=false;
       next_input()
       while (ready)
{ if stack (empty) == false) push (context)
else ready=true
concepts = decode_text_NL(Input,context)

       Evaluate concepts (concepts)
       Output(concepts)
```

Código Fuente 2: Ejemplo de procesamiento de entradas.

Este tipo de interacciones, son planteadas por Yujian [17] así como también por Hume [7], en el proyecto *CircSim* y es el caso donde el STI responde de manera compleja por medio de lenguaje natural. En el caso del esquema planteado, las interacciones del tipo mixto serían simplemente combinaciones del resto de las interacciones disponibles para la interface, pudiendo realizar cualquier combinación que salga del planificador de lección y de una longitud teórica sin limite (probablemente estará acotada en la práctica por la memoria disponible para la cola de mensajes de la interface del submódulo de lenguaje natural). A continuación Se presentan dos secciones de pseudocódigos aptos para trabajar con el modelo mencionado anteriormente para mantener los estados (ver Código Fuente 1 y Código Fuente 2). Estos pseudocódigos son válidos para el proceso que toma los datos de la interface y son procesados dentro del submódulo del lenguaje natural. Con esto no se está modificando el análisis realizado a este módulo visto como una "*caja negra*", sino que este pseudocódigo es para mostrar la ventaja de almacenar el contexto en el que se espera la respuesta y cómo está se procesa.

En el pseudocódigo planteado se puede ver que cada vez que se procesa una entrada (`next_input()`) el sistema identifica el tipo de esa entrada y si es una entrada de control, entra en el ciclo que le permite procesar todas las interacciones del bloque, de otro modo, solo procede a procesar la interacción individual. Cuando recibe una interacción de tipo pregunta (`I_Question`) ó de tipo pista (`I_Hint`), el submódulo almacena él (ó los) estados de la pregunta para facilitar la decodificación de la respuesta (`push (context)`).
Luego, realiza el proceso de construcción lingüística en el lenguaje natural (`Output(processed_text)`) y es el caso de una interacción mixta. La salida del sistema (`encote_text_N(Input)`) sólo se realizará cuando todas las interacciones individuales hayan sido procesadas, de otro modo, la salida será inmediata. En el caso de la decodificación de una respuesta, que utilice el contexto como ayuda, éste quedará almacenado.
Como se observa, se procesan las entradas del usuario con cada uno de los contextos disponibles (`decode_text_NL(Input,context)`) para obtener los conceptos de la respuesta, y una vez terminado de almacenar en `concepts` se procede a evaluar cuál de todas las transcripciones es la más adecuada para alguno de todos los conceptos (`Evaluate_concepts (concepts)`)

6. CONCLUSIONES Y TRABAJOS FUTUROS

Se presentaron los diferentes tipos de interacciones, su análisis a través de ejemplos y la forma de procesar cada una de ellas. Luego, se propone: a) Ampliar los contenidos disponibles para la autoevaluación, b) Escalar el sistema informático de tal forma que permita realizar un seguimiento del alumno, de esta forma el docente puede tener una clusterización de su clase en función de las necesidades cognitivas, c) Escalar el sistema informático hacia bases de datos e interfaces capaces de interactuar con el alumno de manera autónoma y d) Incluir un módulo de autoevaluación en los Sistemas Tutores Inteligentes cuya arquitectura se está desarrollando.

7. REFERENCIAS

[1] Cataldi, Z.; Salgueiro, F.; Lage, F. y García–Martínez, R. 2005. *Sistemas Tutores Inteligentes: redes neuronales para selección del protocolo pedagógico.* V WTIAE. XI CACIC: Congreso Argentino de Ciencias de la Computación. FCAD-UNER. Concordia. Entre Ríos. 17 al 21 de octubre.

[2] Di Eugenio, B.; Michael J. T. (2001*). Can simple Natural Language Generation improve Intelligent Tutoring Systems?.* Electrical Engineering and Computer Science Department. University of Illinois at Chicago.

[3] Evens, M. W.; Spitkovsky, J.; Boyle, P.; Michael, J.; Rovick, A. A. (1993). Synthesizing tutorial Dialogues. Preoceedings of the 15th Annual Conference of the Cognitive Science Society.

[4] Freedman, Reva. 2000. *Plan-Based Dialogue Management in a Physics Tutor. Proceedings of the Sixth Applied Natural Language Processing Conference* (ANLP 2000), Seattle.

[5] Graesser, A., Person, N., Harter, D., y TRG (2001). Teaching *tactics and dialog in AutoTutor.* International Journal of Artificial Intelligence in Education.

[6] Freedman, Reva. 1999. Atlas: A Plan Manager for Mixed-Initiative, Multimodal Dialogue. AAAI-99 Workshop on Mixed-Initiative Intelligence, Orlando.

[7] Hume G., Michael, J; Rovick, A.; Evens, M. (1996), Hinting as a tactic in one-on-one tutoring. Journal of Learning Sciencies.

[8] Evens, M. W.; Stefan, B.; Ru-Charn, C.; Freedman; Glass, M; Hee Lee, Y.; Leem Seop, Shim; Woo Woo, C.; Yuemei, Z.; Yujian, Z,; Joel, A. M.; Rovick, A. A. (2001). *CIRCSIM-Tutor: An Intelligent Tutoring System Using Natural Language Dialogue.* Twelfth Midwest AI and Cognitive Science Conference, MAICS 2001, Oxford, OH, p. 16-23.

[9] Reiter, E. y Dale, R. (2000) Building natural language generation systems Studies in Natural Language Processing Cambridge University Press.

[10] Huang, X. y Fiedler, A.. 1996. Paraphrasing and aggregating argumentative text using text structure. In *Proceedings of the 8th International Workshopon Natural Language Generation*, pages 21–30.

[11] Scott, D. and Sieckenius de Souza, C. .(1990). Getting the message across in rst-based text generation. In R. Dale, C. Mellish, and M. Zock, editors, Current Research in Natural Language Generation, pages 47{73. Academic Press.

[12] Urretavizcaya, M (2001) Sistemas inteligentes en el ámbito de la educación. Revista interamericana de Inteligencia Artificial N. 12. Pp. 5-12

[13] Sancho, L (2002) Una alternativa para el uso de computadoras en educación. Educación Net. Red global de educación a distancia (DistEdNET). Universidad estatal a Distancia.

[14] VanLehn, K (1988). *Student Modelling.* M. Polson. Foundations of Int. Tutoring systems. Hillsdale. N.J. Lawrence Erlbaum Associates, 55-78

[15] Wolf, B. (1984). Context Dependent Planning in a Machine Tutor. Ph.D. Dissertation, University of Massachusetts, Amherst, Massachusetts.

[16] Giraffa, L. M. M. (1997). Seleçao e adoçao de estrateguas de ensino em Sistemas Tutores Inteligentes . Porto Alegre: CPGCC/UFRGS.

[17] Yujian Zhoul, Freedman, R, Glass, M., Michael, J.; Rovick, A. y Evens, M. (1999).What Should the Tutor Do When the Student Cannot Answer a Question?. Proceedings of the Twelfth Florida Artificial Intelligence Symposium.

[18] Glass, M. (1997). Some Phenomena Handled by the CIRCSIM-Tutor Version3 Input Understander. Proceedings of the Tenth Florida Artificial Intelligence Research Symposium, Daytona Beach, FL, 1997, pp. 21–25.