Universidad Nacional de Córdoba

Facultad de Matemática, Astronomía y Física

Trabajo final de la Licenciatura en Ciencias de la Computación

Generando instrucciones de navegación peatonal usando generación por selección

Autor: Santiago Eugenio Avalos Ambroggio

Director: Luciana Benotti

CÓRDOBA ARGENTINA

Septiembre 2014

A mi familia y amigos, que me acompañaron
durante todos mis años de estudio ...

- **Clasificación de Biblioteca:**

  -

- **Palabras Clave:**

  instructores virtuales, generación de lenguaje natural, generación de instrucciones, navegadores peatonales, landmarks, aprendizaje automático.

Resumen

En este trabajo describimos un método para desarrollar un instructor virtual de navegaciń peatonal basado en interacciones reales entre humanos. Un instructor virtual es un agente capaz de cumplir el rol de un instructor humano, y su objetivo es asistir a un usuario humano a llevar a cabo diferentes tareas dentro del contexto de una ciudad real. Para poder cumplir con su trabajo, un instructor virtual necesita poseer la capacidad de describir lo que necesita hacerse de una manera efectiva, considerando las características del entorno virtual en el cual se encuentra, y logrando comprometer al usuario en la tarea propuesta. El instructor presentado se comunica utilizando un algoritmo de generación por selección, basado en un corpus de interacciones reales anotado previamente de forma automática, generado dentro del mundo de interés. Este sistema es resistente a errores de interpretación por parte del usuario, está al tanto de la ubicación del usuario constantemente, y usa diferentes puntos de referencia de la ciudad para guiarlo. El instructor fue evaluado con usuarios reales de forma situada en StreetView de Google, en la ciudad de Edimburgo, mediante la plataforma provista por la competencia internacional GRUVE. El instructor generado fue superior al baseline provisto por la competencia en diversas métricas, como naturalidad y efectividad de las instrucciones, entre otras.

Abstract

In this work we describe a method to develop a virtual instructor for pedestrian navigation based on real interactions between a human instructor and a human pedestrian. A virtual instructor is an agent capable of fulfilling the role of a human instructor, and its goal is to assist a pedestrian in the accomplishment of different tasks within the context of a real city. In order to guide a user while performing a task, an effective instructor knows how to describe what needs to be done in a way that accounts for the nuances of the virtual environment and that is good enough to engage the trainee or the gamer in the activity. The instructor decides what to say using a generation by selection algorithm, based on a automatically annotated corpus of real interactions generated within the world of interest, and is able to react to different requests by the pedestrian. The instructor can deal with interpretation errors made by the user, is constantly aware of the pedestrian position, and it can use different city landmarks to guide him. The instructor was evaluated with real users through Google StreetView, located in the city of Edinburgh, by means of the platform provided by the international challenge GRUVE. The generated instructor was superior to the baseline system provided by this challenge in various metrics, such as naturalness and effectiveness of instructions, among others.

Índice general

Capítulo 1

Introducción

En la última década, el avance en las tecnologías de dispositivos móviles, principalmente de los smartphones o celulares inteligentes, ha permitido que las personas se encuentren en un estado de interacción constante con la red, no solo poniéndose en contacto con otras personas, sino utilizando la misma para obtener información en tiempo real sobre su entorno (tal como consultar un mapa de la ciudad, o averiguar información sobre el clima) e incluso utilizando estos dispositivos para interactuar con el mismo.

Conjuntamente, se ha ampliado la necesidad de desarrollar formas de interacción más naturales entre las computadoras y los seres humanos, incluyendo aplicaciones que utilicen lenguaje natural para interactuar con el usuario. Ejemplos conocidos de estas últimas son los sistemas *Siri* y *Google Now*, que funcionan como asistentes virtuales en teléfonos celulares.

En este ámbito, los sistemas de generación de instrucciones de navegación para peatones son un área de investigación relativamente nueva e interesante, tanto para la lingüística computacional como para el minado de datos y la geoinformática.

Una de las aplicaciones principales de esta clase de sistemas es la resolución de problemas en entornos interactivos con personas, con el objetivo de ayudar a éstas a cumplir con una tarea dada. En estos entornos es necesario generar instrucciones que describan la tarea que otra persona debe realizar. Estos sistemas son llamados "instructores virtuales", y son agentes conversacionales que asisten a un usuario a realizar una tarea específica. Un posible ejemplo simple de este tipo de agente son los sistemas de posicionamiento global (GPS) en automóviles, los cuales generan instrucciones básicas extrayendo información de mapas y de datos satelitales. Otras aplicaciones comunes de estos sistemas son el aprendizaje de idiomas [Nunan, 2004], el entrenamiento mediante simuladores [Kim et al., 2009] o en ciertas implementaciones en el área de entretenimiento, como por ejemplo, guías turísticos en mundos virtuales [Jan et al., 2009], o NPCs (del inglés *No Playable Character*) para videojuegos [Dignum, 2012].

Para poder cumplir con su trabajo, un instructor virtual necesita poseer la capacidad de describir lo que necesita hacerse de una manera efectiva, considerando las características del entorno virtual en el cual se encuentra, y logrando comprometer al usuario en la tarea propuesta. Asimismo, dependiendo de la tarea a realizar, un instructor puede tomar diferentes enfoques a la hora de interactuar con el usuario.

En este trabajo, se propone el diseño de un instructor virtual cuya función es guiar a un usuario humano a través de una representación virtual de una ciudad real. Se pretende que el instructor pueda asistir al usuario para llegar a lugares específicos, comunicándose a través de oraciones escritas en lenguaje natural. Para lograr este objetivo, se utilizará una adaptación propia del algoritmo de generación de lenguaje natural por selección sobre corpus presentado en [Benotti and Denis, 2011].

1.1. Las competencias online de generación

En la actualidad existen, en el área de ligüística computacional, competencias (del inglés *shared tasks*) para comparar sistemas de procesamiento de lenguaje natural[1]. Entre estas, las competencias GIVE [Byron et al., 2007] y GRUVE [Janarthanam et al., 2012] fueron pensadas para evaluar las principales características de algoritmos enfocados en la generación de instrucciones de navegación e interacción en simuladores de entornos físicos o geográficos.

La competencia GIVE (del inglés *Giving Instructions in Virtual Environments*) desarrolló un entorno virtual 3D para la creación y evaluación de sistemas de instrucciones [Byron et al., 2007, Koller et al., 2007]. En este framework, los usuarios recorren un escenario cerrado con distintas habitaciones, corredores y objetos interactivos (de forma similar a un FPS, o *First-Person Shooter*), dirigido mediante las instrucciones que reciben de un sistema de navegación. El objetivo principal de GIVE era comparar sistemas de navegación, que serían evaluados en los mismos escenarios por distintos usuarios a través de internet. La competencia GRUVE, presentada en [Janarthanam et al., 2012], es conceptualmente muy similar a GIVE, pero su objetivo es evaluar sistemas que instruyen a humanos situados en el mundo real o en una simulación del mundo real, como lo es Google Street View. El escenario elegido en GRUVE para evaluar sistemas de navegación peatonal, es una búsqueda del tesoro en la representación de Street View de la ciudad de Edimburgo.

1.2. Motivación

En la actualidad, la mayoría de los agentes conversacionales [Jurafsky and Martin, 2000] operan traduciendo lo que escuchan a *representaciones semánticas formales de los actos del habla* [Searle, 1976], es decir, las oraciones en lenguaje natural son traducidas a algún tipo de abstracción formal que permita trabajar sobre ellas (como por ejemplo, una fórmula en lógica de primer orden o un conjunto de pares $<$*features*, *value*$>$). Esto se hace para permitir razonamientos basados en representaciones computacionales del conocimiento. Estos métodos tienen una importante complicación: se requiere traducir la información desde el texto hasta un nivel mayor de abstracción y viceversa. Este tipo de traducción requiere, o bien una extensa redacción manual de reglas u otro método de procesamiento simbólico, o bien un extenso corpus anotado manualmente sobre el cual trabajar. En ambos casos, estas características incrementan fuertemente el costo de crear un agente conversacional para poder ser utilizado sobre un nuevo entorno, debido al costo del trabajo manual de expertos en el dominio y en el lenjuaje formal usado en la representación semántica. Este aspecto es considerado uno de los mayores inconvenientes a la hora de crear y utilizar dichos agentes.

En este trabajo, se utiliza un algoritmo de generación de lenguaje natural, que se basa en el uso de un corpus de interacciones reales entre usuarios, anotado automáticamente. Esto permite disminuir el tiempo y esfuerzo necesario, tanto en la traducción del texto entre los diferentes niveles de abstracción, como en la adaptación del sistema a diferentes mundos virtuales.

1.3. Descripción del trabajo

En el presente trabajo se propone el diseño, implementación y evaluación de un sistema de generación de instrucciones de lenguaje natural. El objetivo de este sistema, que actuará como un instructor virtual, será ayudar a un usuario real a recorrer distintas partes de una ciudad (en nuestros experimentos, hemos elegido la ciudad de Edimburgo –Escocia– para asegurarnos de que los usuarios evaluadores no estan familiarizados con el entorno). El agente podrá indicar, mediante instrucciones en lenguaje natural, cómo llegar a diferentes ubicaciones interesantes dentro de la ciudad, y asistirá al usuario en caso de que este se extravíe o no comprenda las instrucciones dadas. Google Street View será utilizado para obtener la representación virtual de la ciudad, y el planner de Google Maps será usado por el instructor para discretizar el mundo virtual. El sistema adaptará el modelo de generación por selección presentado en [Benotti

[1] http://www.aclweb.org

and Denis, 2011] para la creación de instrucciones. La selección se hará sobre un corpus de interacciones entre usuarios reales, en el cual serán registrados los diálogos entre parejas de personas, mientras una guía a la otra a través de la ciudad. Este corpus, en idioma castellano, fue recolectado como parte de este trabajo final. El agente será diseñado para adaptarse a las necesidades del framework de evaluación GRUVE y, aunque la competencia no ha tenido lugar todavía, parte de este framework será utilizado para evaluar el sistema con usuarios reales.

1.4. Objetivos

Los siguiente son los objetivos propuestos para este trabajo:

- *Adaptar los algoritmos de generación de lenguaje natural por selección para navegación peatonal:* Se adaptarán los algoritmos de generación por selección para navegación peatonal de ciudades. El agente trabajará sobre el mundo virtual representado en Google Street View.
- *Utilizar expresiones referenciales:* Para lograr una mejor integración con el usuario, se pretende que las instrucciones utilizen expresiones referenciales, y que se basen en referencias visuales del entorno para ayudar al usuario a entender y asimilar los objetivos que propone el agente.
- *Recolectar el corpus necesario*: El corpus constará de interacciones reales en donde un instructor humano guiará a un usuario a través de las calles del mundo virtual.
- *Implementar el instructor*: La implementación se adaptará a las necesidades del framework de GRUVE, para que el sistema sea utilizable en esta plataforma.
- *Evaluar el sistema*: Utilizando la plataforma GRUVE, se evaluará la capacidad del agente implementado con usuarios reales.

1.5. Estructura de la este documento

En la presente introducción hemos mencionado los objetivos que nos proponemos alcanzar en este trabajo de investigación y hemos resumido brevemente algunas de las características deseables de nuestro sistema. A continuación se describirá la estructura de este documento.

En el capítulo 2, se explicarán las formas más utilizadas de generación de lenguaje natural, sus ventajas y desventajas, y se expondrán brevemente trabajos previos en el campo.

En el capítulo 3 se profundizará sobre el método de generación de lenguaje natural por selección, que será el utilizado por nuestro agente.

En el capítulo 4, se presentará una introducción a GRUVE, el framework sobre el cual se implementará el sistema de generación.

La descripción detallada del diseño y funcionamiento del instructor virtual se desarrollará en el capítulo 5, donde también se explica la creación y anotación del corpus de interacciones utilizado.

En el capítulo 6, se exponen las métricas y métodos de prueba para el instructor, y se presentan los resultados obtenidos en la evaluación mediante la plataforma GRUVE.

Finalmente, en el capítulo 7 desarrollaremos las conclusiones obtenidas a lo largo del trabajo, y plantearemos los posibles estudios futuros que darán continuidad al trabajo sobre este tema.

1.6. Impacto del trabajo

1.6.1. Difusión y publicaciones internacionales

Una descripción inicial de este trabajo [Avalos and Benotti, 2014] fue publicada en la *14th Conference of the European Chapter of the Association for Computational Linguistics*, como

parte del workshop *Dialogue in Motion*, que fue organizado por los desarrolladores de la plataforma GRUVE. Al momento de esta publicación el instructor aún no había sido evaluado. La publicación se adjunta en el apéndice A.

1.6.2. Transferencia y vinculación tecnológica

Este trabajo ha sido beneficiado como Idea Proyecto por el Ministerio de Ciencia, Tecnología e Innovación Productiva por su potencialidad para la generación de un producto, sistema, servicio o solución de tecnología de información. El beneficio de 50.000 pesos, conocido como "Beca Jóvenes Profesionales" será íntegramente adjudicado al becario al momento de aprobar su trabajo final.

Capítulo 2

Generación automática de instrucciones

En preparación para el desarrollo del instructor virtual presentado en este trabajo, se realizó un estudio sobre los principales métodos utilizados para generar instrucciones mediante lenguaje natural. Al principio de este capítulo, se presenta una introducción a esta temática, para luego adentrarnos en los métodos de composición y selección. Además, se analizan trabajos previos en los cuales se utilizaron estos métodos para implementar agentes de navegación.

En primer término, en la sección 2.1, se da una instroducción a la generación de lenguaje natural. Luego, en la sección 2.2, se estudia el método de generación por composición, una arquitectura clásica del área de la generación que confecciona lenguaje natural utilizando algún procedimiento de composición, como por ejemplo, reglas gramaticales. Se explicarán las características principales de este enfoque, y sus ventajas y desventajas. En la sección 2.3, nos referimos a Virtual Co-Pilot, una aplicación que utiliza el método de composición para generar instrucciones de navegación para automóviles de manera automática. Este agente prioriza la utilización de landmarks para la creación de instrucciones, característica que nos interesa aplicar a nuestra implementación.

En segundo lugar se analiza el método de generación por selección, el cual genera instrucciones seleccionando candidatos de un corpus previamente anotado de forma automática. En la sección 2.4, discutimos las características sobresalientes de este enfoque, y sus limitaciones y beneficios. Este trabajo, utiliza un método de selección, que opera sobre un corpus anotado automáticamente. Este método es una adaptación del descripto en [Benotti and Denis, 2011], que fue evaluado en la competencia GIVE-Challenge. En razón de que necesitamos realizar, para nuestro trabajo, un análisis más exhaustivo de este modelo, lo presentaremos en el capítulo siguiente, en el cual detallaremos los algoritmos de anotación y selección utilizados y, como caso de estudio, analizaremos la implementación y los resultados obtenidos por el sistema en dicha competencia.

2.1. Generación de lenguaje natural

La generación de lenguaje natural es una rama de la Inteligencia Artificial y de la Lingüística Computacional cuyo objetivo es la creación de sistemas computarizados que puedan producir texto entendible para la comunicación con fines específicos. Texto se refiere aquí a un concepto general y aplicable a expresiones, o partes de ellas, de cualquier tamaño, tanto habladas como escritas. Un generador de lenguaje natural típicamente tiene acceso a un gran conjunto de conocimientos, no solo acerca del lenguaje objetivo y del dominio del problema, sino también de medios de transmisión, y necesidades y capacidad de los receptores.

Típicamente, los generadores de lenguaje natural parten de una representación no lingüística de la información y, valiéndose de esos conocimientos, producen documentos, reportes, explicaciones, mensajes de ayuda y otros tipos de textos [Reiter and Dale, 2000].

Generalmente, es posible caracterizar a un sistema de generación de lenguaje natural (de aquí en más NLG, por sus siglas en inglés) como una 4-upla <K,C,U,D>en la cual:

K es la fuente de conocimiento a usar. Contiene información acerca del dominio y es altamente dependiente de la aplicación y de los objetivos a lograr.

C representa el objetivo comunicacional, es decir el propósito por el cual se desea generar texto.

U es el modelo o perfil de usuario, es decir el usuario de los textos a generar. De este dependerá la selección de información a incluir, y la forma de transmisión.

D representa el historial de discurso. Mantener un registro de la información transmitida, permite, entre otras cosas, el uso de recursos lingüísticos como pronombres o anáforas en los enunciados.

A continuación, veremos las características fundamentales de los dos principales procesos de generación de lenguaje natural: Composición y Selección.

2.2. Generación por Composición

En este método de creación de lenguage natural, el texto es generado dinámicamente en el momento en el cual es requerido. Las instrucciones se generan a través de algún procedimiento de composición (como por ejemplo, un conjunto de reglas gramaticales). Un *acto de habla* en la lingüística y la filosofía del lenguaje es una expresión que tiene la función performativa del lenguaje y la comunicación [Austin, 1962]. Este método trabaja utilizando actos del habla como representación semántica de los enunciados a transmitir.

La generación por Composición se modela típicamente mediante la arquitectura de pipeline propuesta en [Reiter and Dale, 1997]. Este pipeline incluye los siguientes módulos:

Content-determination : Este módulo es el encargado de determinar la información que se quiere comunicar al usuario. Varios factores deben ser tenidos en cuenta a la hora de tomar esta decisión (tales como los objetivos comunicacionales, la capacidad y conocimiento del receptor sobre el mundo, y las restricciones del medio). En general, el algoritmo NLG se involucra directamente con esta selección de contenido. Aquí también debe decidirse cómo se estructurará esta información al momento de transmitirla.

Micro-planning : Se encarga de establecer cómo se hará referencia a los diferentes objetos, es decir, de la creación y utilización de expresiones referenciales; y de cómo agrupar y factorizar la información. Parte de las tareas de este módulo son la lexicalización (tarea que consiste en definir las palabras y estructuras sintácticas que serán necesarias para expresar el contenido elegido por el sistema), y la agregación (que decide la forma en la cual se mapean las estructuras creadas por el módulo de content-determination en estructuras lingüísticas tales como oraciones y párrafos).

Surface-realization : Selecciona las palabras y construcciones gramaticales que serán transmitidas en la instrucción. Este módulo se encarga del problema de aplicar reglas gramaticales a representaciones abstractas con el objetivo de producir un texto sintáctica y morfológicamente correcto.

Los enfoques composicionales actuales pueden ser clasificados como **ruled-based** (basados en reglas) o **corpus-based** (basados en corpus). En los sistemas pertenecientes al primer grupo, los tres módulos deben ser diseñados específicamente para la tarea a realizar, y se requiere una amplia tarea de adaptación para que puedan ser reutilizados en una tarea diferente. En particular, el módulo de *content-determination* debe ser implementado nuevamente para cada tarea [Reiter et al., 2003, Traum et al., 2003], y aunque para los otros dos módulos existen implementaciones,

supuestamente independientes del contexto [Krahmer and van Deemter, 2012, Gardent and Kow, 2007], en la práctica el costo de adaptar los sistemas para realizar nuevas tareas es prácticamente tan alto como el de desarrollar un sistema nuevo [DeVault et al., 2008].

Los sistemas del segundo grupo se propusieron como una manera de facilitar la adaptación de los modulos de generación de lenguaje natural a nuevas tareas. En diferentes trabajos, se analizaron técnicas estadísticas para adaptar a diferentes dominios los módulos de *content-determination* [Duboue and McKeown, 2003, Lapata, 2003], *micro-planning* [Walker et al., 2007] y *surface-realization* [Bangalore and Rambow, 2000].

Estos enfoques pusieron en evidencia las debilidades de la arquitectura de pipeline a la hora de generar lenguaje natural, a causa de las fuertes interacciones entre los distintos modulos, ya que al tratar de adaptar alguno a una nueva tarea, los restantes deben ser modificados de igual modo para poder seguir funcionando. Trabajos recientes han dejado de lado esta arquitectura y propusieron maneras de optimizar conjuntamente el proceso de generación mediante *reinforcement learning* [Dethlefs et al., 2011] y *cost-based automated planning* [Garoufi and Koller, 2011].

Todos estos métodos utilizan gran cantidad de conocimiento lingüístico y pragmático, que debe ser anotado manualmente por expertos, tanto en el lenguaje formal de representación semántica como en el dominio, para que el lenguaje generado sea efectivamente útil para la tarea sobre la cual trabaja. Esta es una de las características que generan mayor costo a la hora de diseñar un generador de este tipo. Como consecuencia, este tipo de técnicas no se usan en sistemas comerciales.

2.3. Generación de instrucciones basada en templates

En esta sección analizaremos el agente Virtual Co-Pilot [Dräger and Koller*, 2012], un sistema que genera instrucciones habladas de navegación para automóviles basadas en templates. Este sistema genera un ruta mediante un planificador, divide la ruta en diferentes objetivos, y luego utiliza OpenStreetMap[1] para obtener información del entorno y crear instrucciones en tiempo real utilizando *landmarks* (definidos como lugares prominentes del entorno, que son fácilmente reconocidos o que poseen importancia propia) presentes a lo largo del recorrido.

2.3.1. Motivación y análisis del problema

En [Burnett, 2000], se trabajó experimentalmente en un entorno de manejo, para concluir que las rutas basadas en landmarks ocasionan menos errores de interpretación y de conducción. Sin embargo, la gran mayoría de los sistemas de navegación comerciales (como ser los equipos de GPS para autos) utilizan generadores de lenguaje basados en templates, y se limitan a utilizar instrucciones basadas en la distancia al objetivo. Incluso en el ámbito académico, es muy poco común encontrar trabajos que utilicen navegación por landmarks, y los sistemas que lo hacen, utilizan mapas confeccionados especialmente para el sistema de generación particular [Malaka et al., 2004] o una combinación de múltiples recursos [Raubal and Winter, 2002] que limitan su cobertura.

En Virtual Co-Pilot se estudian soluciones para dos problemas claves en el desarrollo de sistemas para navegación de automóviles. En primer lugar, la falta de disponibilidad de recursos cartográficos, es decir, la necesidad de tener mapas detallados con anotaciones de landmarks de las ciudades en las cuales el agente de navegación debe funcionar. El segundo problema se da cuando el usuario no sigue las instrucciones de acuerdo al plan del instructor, y por lo tanto este debe adecuar su plan de manera automática para poder reaccionar ante estos cambios.

2.3.2. Generación de instrucciones

Cada vez que el agente Virtual Co-Pilot (desde ahora VCP) quiere guiar al conductor hacia un objetivo, genera tres tipos de mensajes:

[1] www.openstreetmap.org

- *Decision points (dps)*: generado en las intersecciones en las cuales se necesita un maniobra, como girar. Se da al conductor en forma inmediata (por ejemplo, "now turn right").
- *Confirmation message (cmg)*: se genera cuando el usuario completa una instrucción con éxito (en el ejemplo anterior, si se giró en el sentido indicado) para indicar que está en la dirección correcta.
- *Preview message (pmg)*: se generan en la calle sobre la cual está el *decision point*, antes de llegar a él, y describen la ubicación en la cual debe ser ejecutada la maniobra.

Al definir un objetivo, VCP calcula el conjunto completo de instrucciones necesarias para llegar a él. Al trabajar en un entorno interactivo, el agente monitorea constantemente la posición y dirección del usuario, y en caso de que este no siga la ruta esperada, el sistema reacciona planeando una nueva ruta desde el punto de posición actual del usuario y generando un nuevo conjunto de instrucciones.

A continuación, veremos en detalle las tareas que cumple cada módulo de la arquitectura de generación de lenguaje natural implementada en VCP:

2.3.3. Content-determination

Planificación

Para obtener información del entorno en donde funciona, VCP utiliza OpenStreetMap, un mapa on-line virtual, que provee información actualizada sobre las ciudades del mundo. Este mapa funciona de manera similar a la wikipedia, permitiendo que usuarios de todo el planeta anexen información continuamente. El agente lo utiliza para dos tareas: obtener información de la red de calles y rutas, lo que permite crear planes de navegación, y recopilar información topográfica (tal como la ubicación) y semántica (características de tipo y forma) sobre los posibles landmarks a utilizar en la generación de instrucciones. La información geográfica en este mapa está representada en forma de nodos, definidos mediante latitud y longitud, que representan puntos en el espacio, y segmentos, que unen distintos nodos, y representan las calles. Para obtener un plan hacia el objetivo, VCP busca el camino más corto en el grafo conformado por nodos y segmentos. El resultado es una lista ordenada de segmentos que representan las calles que deben ser recorridas para completar la tarea.

Determinación del contenido

Una vez obtenido el plan, éste es dividido en episodios, cada uno asociado a una instrucción. Se considera que un episodio termina cada vez que, en el plan, dos segmentos cambian de nombre (ya que esto indica un dps) o cada vez que se continúa por la misma calle, pero se alcanza una intersección de varios segmentos, y los que conforman la calle transitada están separados por un ángulo lo suficientemente abierto (por ejemplo, una curva en la cual la calle se conecta a otra via menor, lo cual genera un dps).

2.3.4. Microplanning

Agregación

Al ser un sistema que utiliza instrucciones verbales, VCP debe considerar el tiempo que demoran estas instrucciones en transmitirse al usuario. Es indeseable que dos instrucciones se superpongan; asimismo, si se retrasa una instrucción para terminar la anterior, se corre el riesgo de que el usuario pierda un dps, o incluso interprete una instrucción de forma errónea.

Esto es solucionado mediante la utilización de los pmg, utilizando el planificador para calcular *trigger positions* en las cuales se dan los mensajes. En el caso de que algún episodio sea demasiado corto como para acomodar todos los mensajes necesarios, se eliminan el pmg del episodio y el cmg de la instrucción anterior, y se agrega al mensaje del episodio anterior la información necesaria (por ejemplo: "Now turn right, and then turn left after the church").

Como consecuencia de la necesidad de aglomerar distintos mensajes dependiendo del tiempo

de presentación, VCP calcula todo el conjunto de instrucciones necesarias para cumplir un plan, en vez de calcular sólo la necesaria para completar un episodio. Si el usuario no cumple con una instrucción correctamente, el conjunto de instrucciones debe ser recalculado.

Descripción de landmarks

VCP prioriza aquellos landmarks que son fácilmente visibles, para poder ser notados al conducir, y genéricos, es decir, que puedan ser utilizados indistintamente en varias ciudades. Luego los separa en dos categorías:

- Street furniture (landmarks de calle): objetos genéricos que estan instalados en las calles: señales de stop, semáforos, y cruces peatonales. Se considera que estos tienen *salience*, o prominencia, ya que son intrínsecamente importantes para cualquier conductor.
- Visual landmarks (landmarks visuales): edificios que si bien no estan relacionados con la infraestructura de la calle, atraen la atención del conductor. En este grupo se encuentran las iglesias, estaciones de servicio, supermercados, etc.

2.3.5. Surface-realization

Selección de landmarks

Una vez que VCP determina un nuevo episodio, elige un grupo de landmarks como candidatos, basándose en la cercanía al dps. Dentro del grupo, se eliminan los candidatos que no son únicos (es decir, si hay más de un candidato del mismo tipo, todos son eliminados). Para los episodios en los cuales existen múltiples landmarks de calle del mismo tipo, se conservan los primeros tres, agregando expresiones referenciales (por ejemplo, "at the second traffic light") para referirse a ellos. Si el dps esta a no más de tres intersecciones, se agrega un candidato de la forma "at the third intersection". Finalmente, para ser considerado, un landmark debe ser visible desde el último segmento del episodio.

Dentro de los candidatos finales, el sistema da precedencia a los landmarks visuales antes que a los pertenecientes a la calle, y a estos últimos antes que a las intersecciones. En caso de no contar con ningún candidato disponible, el sistema utiliza instrucciones basadas en distancia.

Luego este proceso, VCP obtiene una representación semántica, que transforma en lenguaje natural utilizando modelos o *templates* sintácticos (como ser "Turn direction preposition landmark"). Podemos ver abajo un ejemplo de esta información, junto con su traducción final:

```
Preview message p1:
  Trigger position:     Node3 - 50m
  Turn direction:       right
  Landmark:             church
  Preposition:          after

Turn instruction t1 :
  Trigger position:     Node3
  Turn direction:       right

Instruction: "Turn right after the church"
```

2.3.6. Evaluación y resultados

Este sistema se evaluó utilizando un simulador de manejo sobre una representación virtual 3D de una cuidad real, la cual contenía edificios y calles tal como podían ser observados en la realidad. Los sujetos de evaluación fueron 12, balanceados por género.

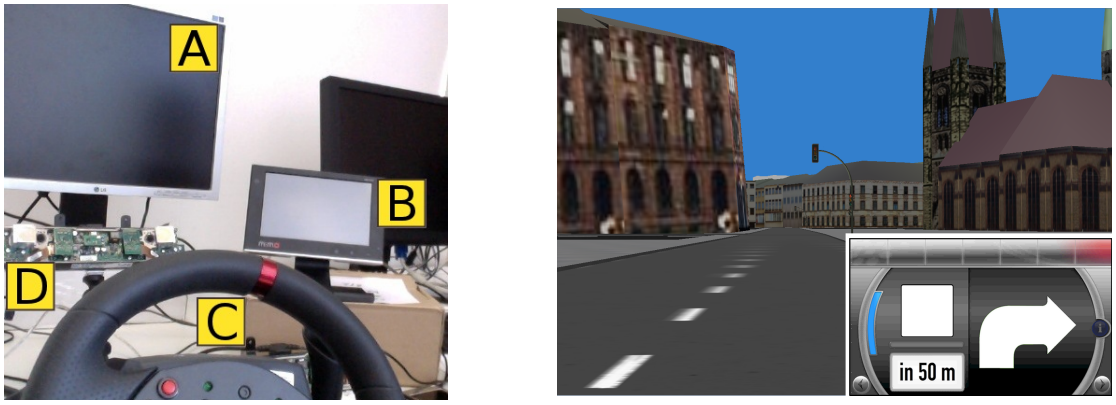

Figura 2.1: Simulador de evaluación y vista del usuario.

En la figura 2.1 podemos ver el simulador, y la representación desde el punto de vista del usuario.

Cada uno de los sujetos de prueba debía manejar tres rutas de aproximadamente 1 km en el simulador. Cada ruta tenía entre tres y cinco episodios, y la cantidad de landmarks presentes fue suficiente como para que el sistema no deba utilizar nunca instrucciones basadas en distancia. Se registró el movimiento de los ojos en los usuarios mediante un eye-tracker Tobii IS-Z1, y se vocalizaron las instrucciones mediante MARY, un sistema de sintesis de voz open-source.

Se hicieron pruebas en tres entornos diferentes. El primero, utilizado como baseline, presentaba instrucciones basadas en distancia junto con una flecha indicando la direccion deseada (al igual que un sistema comercial GPS). El segundo, cambiaba las instrucciones por las que generaba VCP, y el tercero agrega además un icono que representa el landmark elegido junto a la flecha de dirección.

Luego de cada prueba, los participantes contestaron el cuestionario DALI [Pauzié, 2009], que consta de preguntas referidas a la cantidad de trabajo cognitivo (visual, auditivo, stress percibido, etc) y un cuestionario que reflejaba métricas subjetivas de evaluación, como ser la confianza que tienen en el sistema, o la importancia que daban al hecho de tener referencias visuales de las instrucciones recibidas.

La evaluación reportó que no existieron diferencias significativas entre VCP y el sistema baseline en cuanto al tiempo de para completar la tarea, promedio de errores de manejo, o el cuestionario DALI.

	All Users		Males		Females	
	B	VCP	B	VCP	B	VCP
Total Fixation Duration (seconds)	4.9	**3.5**	**2.7**	4.1	7.0	**2.9**
Total Fixation Count (N)	21.8	**15.4**	**13.5**	16.5	30.0	**14.3**
"The system provided the right amount of information at any time"	**3.9**	2.9	**4.2**	3.3	**3.5**	2.5
"I was insecure at times about still being on the right track"	**2.3**	3.2	**1.9**	2.8	**2.6**	3.5
"It was important to have a visual representation of route direction"	4.3	**4.0**	4.2	4.2	4.3	**3.7**
"I could trust the navigation system"	3.6	**3.7**	**4.1**	3.7	3.0	**3.7**

Cuadro 2.1: Resultados de VCP. Valores promedio (B=baseline). Valores mejores remarcados.

En cuadro 2.1 pueden verse los resultados. Es notable la disminución significativa en el número de veces en que los usuarios femeninos observan la pantalla de navegación (Total Fixation Count), y en la duración de cada observación (Total Fixation Duration). También puede verse que en el cuestionario presentado, las mujeres tienden a evaluar de manera más positiva al sistema VCP que al baseline (sobretodo en lo referido a la confianza en el sistema y a la necesidad de la pantalla de navegación). Los usuarios varones evaluaron peor a VCP en cuanto al tiempo de

presentación de las instrucciones y en el sentido de seguridad sobre la ruta que proporcionaba.

2.3.7. Conclusiones sobre Virtual Co-Pilot

Considerando los resultados obtenidos por VCP, debemos analizar las características que nos resultan atractivas a la hora de diseñar nuestro agente. En primer lugar, VCP logra guiar con éxito al usuario en la completitud de la tarea presentada, expresándose en lenguaje natural de manera entendible y práctica. Además, puede encontrar un plan eficiente dentro del entorno presentado y adaptarse a nuevos mundos virtuales mientras estos posean características similares. El uso de landmarks en VCP también es un punto sobresaliente que queremos incluir en nuestro navegador, así como también su capacidad de adaptarse a usuarios que por alguna causa no acatan de manera correcta las instrucciones. Por otro lado, nuestro trabajo se diferencia de VCP en cuanto al medio de navegación, ya que queremos implementar un navegador peatonal. Esta diferencia produce cambios fundamentales en el diseño, ya que el tiempo que tarda una instrucción en transmitirse deja de ser un problema mayor en el sistema. Esto permite eliminar una de las mayores desventajas, a nuestro criterio, de VCP, que es la planificación adelantada de todo el conjunto de instrucciones. Consideramos que este es un punto negativo ya que en la navegación peatonal es mucho más probable que el usuario no siga al pie de la letra una instrucción, lo que ocasionaría, si se implementa este método, un cálculo constante de instrucciones que nunca serían utilizadas. Además, como fue explicado en la sección 2.2, en la cual se presentaron las desventajas de los sistemas de generación composicional, se tuvo que incurrir en dos procesos de anotación manual para poder implementar este sistema: primero, sobre el recurso cartográfico utilizado (OpenStreetMaps), para poder identificar y clasificar landmarks; y segundo, en el momento de diseñar los templates necesarios para generar las instrucciones.

2.4. Generación por Selección

En el capítulo 3 analizaremos este método en detalle. Aquí haremos una introducción a sus características principales, ventajas y desventajas. La generación por selección apunta a disminuir el costo de desarrollo de los agentes conversacionales seleccionando oraciones desde una base de datos. Los métodos de selección trabajan directamente sobre un corpus previamente generado. Actualmente, la mayoría de los sistemas que usan generación por selección utilizan oraciones creadas por humanos, pero que no fueron recopiladas en un contexto real de interacción entre distintas personas [Leuski et al., 2006, Kenny et al., 2007]. Usar diálogos reales obtenidos en situaciones en las cuales los interlocutores se encuentran inmersos en el entorno que generó la interacción, podría mejorar significativamente la manera en que los usuarios evalúan la naturalidad de las instrucciones.

El método de selección utiliza un algoritmo que se encarga de elegir, sobre un conjunto de datos anotados previamente, cuál es el candidato más adecuado para una situación determinada en la cual se requiere generar una instrucción. A diferencia de los métodos de composición, no se trabaja en el nivel de la generación propiamente dicha del lenguaje natural, sino que la problemática principal en estos sistemas es elaborar un algoritmo de selección que funcione lo mejor posible en el entorno sobre el cual se trabaja. Los corpus utilizados por estos métodos poseen no sólo las interacciones en lenguaje natural, sino que también registran información pertinente al mundo en el cual fueron elaboradas las oraciones, incluyendo una representación abstracta de las reacciones causadas sobre el entorno por la persona que recibió una instrucción.

En general, toda generación por selección utiliza algún estímulo para decidir qué oración seleccionar. Visto de esta forma, la tarea de selección puede ser considerada una tarea de clasificación [Chu-Carroll and Carpenter, 1999] de la siguiente manera: Cuando se presenta un nuevo estímulo –por ejemplo, una pregunta– se lo describe primero como un vector que manifiesta sus principales características, luego se lo compara con vectores de estímulos conocidos –preguntas dentro de las interacciones del corpus– para encontrar el grupo de estímulos que mejor lo representan, y finalmente la reacción asociada a ese grupo –la respuesta– es devuelta. La principal desventaja que tiene este método es que ignora completamente el contenido de las reacciones anotadas en el corpus. Trabajos posteriores proponen considerar estas reacciones a la hora de

seleccionar, comparando por ejemplo las preguntas del usuario con las respuestas conocidas, no con los estímulos conocidos [Leuski et al., 2006, Leuski and Traum, 2011]. De esta manera, la tarea de selección se trabaja como un *information retrieval problem*. Este enfoque obtiene, en la práctica, mejores resultados en cuanto a la relevancia de la reacción con el estímulo dado.

Las ventajas de los métodos de selección son varias. En primer lugar, da la posibilidad de trabajar con oraciones de mayor complejidad, y más parecidas a oraciones pertenecientes a interacciones reales humanas. Comparados con agentes basados en composición, los sistemas por selección requieren de menos recursos para ser adaptados a un entorno para el cual no fueron diseñados (por ejemplo un mundo virtual diferente). Finalmente, no es necesaria ninguna clase de anotación manual o intervención de un experto para identificar el significado de las instrucciones.

La mayor desventaja de la generación por selección, es que el diálogo resultante puede no ser completamente coherente [Shawar and Atwell, 2003, 2005, Gandhe and Traum, 2007]. Orkin and Roy [Orkin and Roy, 2009] señalan problemas de coherencia graves en agentes conversacionales orientados a tareas que utilizan la generación por selección sobre un corpus de interacciones entre humanos diseñado para un restaurante virtual.

Otra dificultad de este método es la obtención de corpus sobre los cuales hacer el proceso de selección. Aunque existen corpus de interacciones humanas orientadas a tareas disponibles al público [Gargett et al., 2010], la necesidad de obtener un corpus específicamente diseñado sobre el mundo virtual en el cual se trabaja ocasiona que este corpus deba ser recopilado nuevamente al cambiar el entorno.

Finalmente, los métodos de selección son muy sensibles a los errores en el corpus. Al considerar como semántica de las instrucciones sólo sus reacciones, el sistema no puede identificar en primera instancia instrucciones ambiguas o erróneas. Este problema puede tratarse, por ejemplo, haciendo un proceso de revisión que analice las interacciones registradas, eliminando del corpus las que puedan ocasionar problemas mas adelante.

Capítulo 3

Generación de instrucciones por selección

En este capítulo, analizamos las técnicas presentadas por [Benotti and Denis, 2011], que serán las que luego aplicaremos para desarrollar el agente implementado en este trabajo. En este sistema, al cual llamaremos UCM (del inglés *Unsupervised Conversational Model*), se utilizó un algoritmo de selección sobre un corpus que fue anotado automáticamente, para desarrollar un instructor virtual cuya función fue asistir a usuarios a completar determinadas tareas. Luego, el agente fue evaluado con usuarios reales de manera online en el marco de la GIVE-2.5 Challenge.

En la sección 3.1, daremos una descripción detallada de la tarea para la cual se desarrolló este agente. Luego, en la sección 3.2, presentaremos la creación del corpus utilizado, y la implementación del algoritmo de selección. Finalmente, en la sección 3.3, mostraremos el comportamiento del agente UMC en una situación real de ejecución, examinaremos la evaluacion previa que se hizo del sistema, y por ultimo presentaremos los resultados obtenidos por el sistema al ser evaluado de manera on-line en la competencia GIVE2.5-Challenge.

3.1. Descripción de la tarea

La tarea principal a solucionar consiste en guiar a un usuario humano a través de un contexto virtual con el objetivo de encontrar un "tesoro". El entorno sobre el cual se trabaja es el mundo virtual creado para la competencia GIVE-2 (este puede observarse en detalle en las Figuras 3.1 y 3.4). Este está formado por distintas habitaciones, que el usuario recorre desde una perspectiva en primera persona. Es posible interactuar con diversos botones a lo largo del espacio, que abren o cierran puertas; además se deben sortear o desactivar alarmas en el piso de las habitaciones.

Para completar esta tarea, interactúan necesariamente una pareja de usuarios, cada uno cumpliendo un rol diferente para lograr el objetivo. Definiremos estos roles de la siguiente manera:

- **Direction Giver (DG)**: Es quien posee conocimiento completo sobre el mundo virtual o real, y conoce la manera de resolver el objetivo planteado. Su función es dar instrucciones a su compañero para guiarlo en la resolución del objetivo.
- **Direction Follower (DF)**: No conoce el mundo ni sabe como resolver el objetivo, pero es quien lo recorre y puede ocasionar cambios en él. Su función es interpretar las instrucciones del DG para lograr cumplir con el objetivo planteado.

El instructor virtual debe cumplir la tarea del Direction Giver, guiando al Direction Follower a través del mundo virtual.

3.2. Algoritmo de Selección

El algoritmo de selección utilizado por UCM consiste en dos etapas principales: la primera, llamada de anotación, consiste en formar una nueva base de datos, a partir del corpus, que será

utilizada para obtener instrucciones. La segunda, llamada de selección, consiste en seleccionar del corpus anotado la instrucción más apropiada para una situación determinada.

En la seccion 3.2.1 se presentará el corpus sobre el cual se produjo la anotación, el GIVE-2 Corpus. En la sección 3.2.2 analizaremos la etapa de anotación, y la creación del corpus anotado. Luego, en la sección 3.2.3 estudiaremos el funcionamiento del algoritmo de selección, en el que se detallan los criterios utilizados para diferenciar a los diferentes candidatos, y la comunicación del sistema con el usuario.

3.2.1. El GIVE2-Corpus

En este agente, se trabajó sobre el GIVE-2 Corpus [Gargett et al., 2010], un corpus gratuito de interacciones entre humanos recopilado mientras cooperan para resolver una búsqueda del tesoro sobre el entorno virtual de GIVE-2. Éste está escrito en alemán e inglés, y consiste en todas las instrucciones emitidas por el DG, y todas las manipulaciones hechas por el DF sobre el mundo, con su correspondiente marca de tiempo. Además, la posición y orientación del DF fue registrada cada 200 milisegundos, haciendo posible conocer la información exacta acerca de sus movimientos.

Para evaluar UCM, se utilizó la parte en inglés del corpus, que estaba formada por 63 discursos escritos en los cuales un sujeto, actuando como DG, guiaba a otro, el DF, en un tarea de búsqueda del tesoro en 3 diferentes entornos 3D pertenecientes a GIVE-2.

3.2.2. Anotación

Como se vio en el capítulo anterior, los métodos de selección requieren un corpus de interacciones en el cual los usuarios humanos hayan tenido el mismo objetivo para el cual se implementó el agente. El sistema selecciona instrucciones de este corpus, comparando sus reacciones asociadas con la reacción esperada del usuario en un momento dado.

En GIVE-2, el estado del mundo se representa como un conjunto de literales de lógica de predicados que describen cada aspecto, tanto estático como dinámico, del entorno. Los literales que representan propiedades estáticas del entorno virtual describen el escenario, mientras que los dinámicos describen cada uno de los cambios aplicados en el entorno por el jugador. Estos últimos no solo incluyen objetos del mundo virtual (un botón puede estar presionado o no, una puerta abierta o cerrada), sino que también hacen referencia al usuario (registrando su posición y orientación, que es variable).

La idea básica de esta etapa es directa: asociar en el corpus cada oración con su correspondiente reacción. Se puede asumir que una reacción captura la semántica de su instrucción asociada. Sin embargo, definir 'reacción' implica dos problemas más sutiles: la determinación de límites y la discretización.

Definición de límites

Los límites (del inglés *boundaries*) de una reacción pueden ser definidos de la siguiente manera: *la reacción Rk asociada a la instrucción U_k comienza justo después de que la instrucción U_k es emitida y termina justo antes de que la siguiente instrucción U_{k+1} se emite.* En el siguiente ejemplo la instrucción 1 se corresponde con la reacción <2, 3, 4>, la instrucción 5 se corresponde con la reacción <6>, y la instrucción 7 con <8>:

DG(1): hit the red you see in the far room
DF(2): [enters the far room]
DF(3): [pushes the red button]
DF(4): [turns right]
DG(5): hit far side green
DF(6): [moves next to the wrong green]
DG(7): no

DF(8): [moves to the right green and pushes it]

Este ejemplo nos muestra que la definición dada de límite para la reacción no siempre captura la semántica de manera correcta. Por ejemplo, se puede argumentar que la reacción asociada a (1) abarca demasiada información, ya que esta contiene a (4), una acción que no es consecuencia directa de la instrucción (1). Incluso se puede ver que instrucciones interpretadas incorrectamente (como la 5) y correcciones (como la 7) resultan en asociaciones incorrectas entre las instrucciones y las reacciones.

Sin embargo, aun con los problemas presentados, UCM usa esta simple definición de límite para las reacciones, ya que permite completar esta primera etapa sin recurrir a ningún tipo de anotación manual. En el capítulo 5 se presentan los recaudos que fueron tomados a la hora de implementar esta técnica a nuestro trabajo, para encontrar una solución a este tipo de instrucciones problemáticas.

Discretización

El segundo problema que abordamos es la discretización de la reacción misma. Toda tarea de selección necesita, para poder decidir qué decir a continuación, un *planner* y un *planning problem*, es decir: una especificación de cómo funciona el mundo virtual (un planificador que describe cuales acciones son posibles), una manera de representar el estado del mundo en un momento dado, y una manera de representar el objetivo de la tarea. Como vimos anteriormente, la tarea está definida (Sección 3.1) y pueden especificarse de forma completa las características que definen el estado del mundo (Sección 3.2.2).

GIVE-2 utiliza LazyFF, una implementación en Java del planificador FF [Hoffmann and Nebel, 2001], al que se le describen las condiciones mediante PDDL (Planning Domain Definition Language) [Hsu and Wah, 2006], el lenguaje estándar de definición de dominios de planificación.

Estas características pueden utilizarse para discretizar la reacción, pero es importante tener en cuenta el nivel de granularidad utilizado en el planner (es decir, la especificidad que utiliza para distinguir entre diferentes acciones), ya que se debe lograr que la discretización capture lo mejor posible la diversidad deseada entre las instrucciones.

Es conocido que no existe una única forma de discretizar una acción en diferentes sub-acciones (por ejemplo, la acción 2 anterior podría ser descompuesta en "enter the room" o en "get close to the door", "pass the door"). Aunque este algoritmo no depende de una discretización particular, es necesario notar que para obtener un funcionamiento correcto, la misma forma de discretización utilizada en la fase de anotación debe ser usada en la fase de selección.

Definición de *reacción asociada*

Definamos ahora 'reacción' de una manera formal. Sea S_k el estado del mundo virtual en el momento de emisión de la instrucción U_k, S_{k+1} en el momento de emitir $U - k + 1$ y *Acts* la representación de las acciones posibles dentro del mundo virtual. La reacción asociada a U_k se define de la siguiente forma: *la secuencia de acciones devuelta por el planner con* S_k *como el estado inicial,* S_{k+1} *como el final, y Acts como las acciones.* Dada esta definición, la fase de anotación del corpus consiste en asociar cada instrucción a su reacción (discretizada).

La utilización del planner para definir las reacciones es consecuencia de la necesidad de "normalizar" el concepto de equivalencia entre estas. Resulta claro que existen diversas maneras de realizar los mismos cambios en el mundo virtual, y un planificador provee una manera de establecer formalmente qué acciones son necesarias para lograr un cambio específico en el estado del mundo, definiendo asi cuáles reacciones producen el mismo cambio a través de diferentes acciones.

En el cuadro Algoritmo 1 podemos ver el algoritmo que implementa esta anotación. Abajo puede verse un fragmento del corpus resultante.

```
Utterance: make a left and exit the room
Reaction: < move(b2-room-1-9,room-1-9), move(room-1-9,room-1-8),
```

Algorithm 1 Algoritmo de anotación.

Acts ← *world possible actions*
for all *Utterance* $U_k \in Corpus$ **do**
  $S_k \leftarrow$ *world state at* U_k
  $S_k + 1 \leftarrow$ *world state at* $U_k + 1$
  $U_k.reaction \leftarrow plan(S_k, S_k + 1, Acts)$
end for

```
            move(room-1-8,room-1-7), move(room-1-7,room-1-6),
            move(room-1-6,room-1-3), move(room-1-3,room-1-4),
            move(room-1-4,room-1-5), move(room-1-5,d3-room-1-5) >

Utterance: go forward and turn 90 degrees
Reaction: < move(d3-room-1-5,d3-room-2), move(d3-room-2,room-2) >
```

3.2.3. Selección

En esta sección se describe cómo se ejecuta la fase de selección cada vez que UCM necesita generar una instrucción. Supongamos que la tarea actual puede solucionarse mediante el plan P (este plan es la secuencia de acciones que necesitan ser ejecutadas en el mundo virtual para poder completar dicha tarea). El algoritmo de selección consiste en elegir, dentro del corpus, un conjunto de candidatos C que pueden orientar al usuario a seguir los pasos necesarios para completar esa tarea. Es decir, candidatos cuya reacción asociada sea un prefijo, o tramo inicial, del plan actual P.

Definimos:

$$C = \{U \in Corpus \mid P \text{ starts with } U.Reaction\}$$

En otras palabras, una oración U pertenece al conjunto C si las primeras acciones del plan actual P son exactamente iguales a la reacción asociada a U en el corpus.

Dentro del lenguaje natural, dos oraciones son consideradas paráfrasis semánticas cuando transmiten el mismo significado, utilizando diferentes expresiones. Dentro del contexto del corpus, el contenido semántico de una instrucción no está realacionado a su significado en lenguaje natural, sino que se encuentra definido por su reacción asociada, por lo tanto, todas las oraciones que son prefijos del plan P, pueden ser consideradas *paráfrasis pragmáticas* en este contexto, y todas son adecuadas para ser seleccionadas por el algoritmo.

Esta diferencia con la noción estándar de paráfrasis es clara de ver si se observa el Cuadro 3.1, en el cual se listan todos los candidatos para un momento particular en una partida. Mientras que a simple vista "straight" ("derecho") y "go through the opening on the left" ("andá a través de la abertura en tu izquierda") tienen significado diferente, ambas tienen como reacción asociada un prefijo del plan, y por lo tanto son consideradas paráfrasis.

En el momento en que P cambia, como resultado de alguna acción por parte del DF, se llama nuevamente a este algoritmo para regenerar el conjunto de candidatos C y adecuarlo así a los cambios en la situación del mundo virtual. El cuadro Algoritmo 2, describe formalmente el algoritmo de selección.

Algorithm 2 Algoritmo de selección

$C \leftarrow \emptyset$
$action \leftarrow nextAction(currentObjective)$
for all *Utterance* $U \in Corpus$ **do**
  if $action \approx U.Reaction$ **then**
    $C \leftarrow C \cup U$
  end if
end for

Para seleccionar qué información transmitir, UCM elige siempre la oración cuya reacción es

más larga, es decir, el mayor prefijo del plan actual, de entre los cuales están asociados a un candidato. Esto permite maximizar el avance en el plan en cada instrucción. En el caso en que el plan no cambie, lo que significa que el usuario no responde ante el candidato elegido, UCM emite las instrucciones siguientes, siempre utilizando el mismo criterio.

Por otro lado, puede ocurrir que el conjunto de candidatos C quede vacío, ya que el algoritmo no puede encontrar en el corpus ninguna reacción equivalente al plan actual (o ya ha utilizado todas las disponibles). En este caso, una instrucción estandar y neutral, "go", es emitida.

Es importante notar que la discretización utilizada en ambas fases (anotación y selección) afecta directamente al comportamiento del instructor. Elegir una granularidad muy amplia, podría ocasionar que muchas instrucciones no tengan ninguna reacción asociada, ya que esta no es capturada por el algoritmo de anotación. Análogamente, si el usuario llega a situaciones que no fueron contempladas puntualmente al crear el corpus, una granularidad muy fina podria ocasionar que el conjunto de candidatos creado por el algoritmo de selección sea vacío.

3.3. Funcionamiento del agente

A continuación, veremos en detalle cómo se comporta UCM en una situación real, con usuarios humanos. Primero, en la secciń 3.3.1 examinaremos un ejemplo paso a paso del funcionamiento del algoritmo de selección, y veremos cómo genera el conjunto de candidatos, selecciona una instrucción, e interactúa con el usuario. Luego, en la sección 3.3.2, presentaremos la evaluación local a la que UCM fue sometido al ser desarrollado. Finalmente, en la sección 3.3.3, mostraremos los resultados obtenidos por este sistema en la competencia GIVE-2.

3.3.1. Ejemplo de ejecución

En esta sección presentamos un extracto de una interacción entre UCM y un usuario. Recordemos que el agente está cumpliento el rol de DG, y debe guiar al usuario a través del entorno. Las imágenes han sido tomadas de [Benotti and Denis, 2011].

En cada figura puede verse un mapa en 2D desde arriba, que representa el escenario, y la representación 3D desde el punto de vista del usuario.

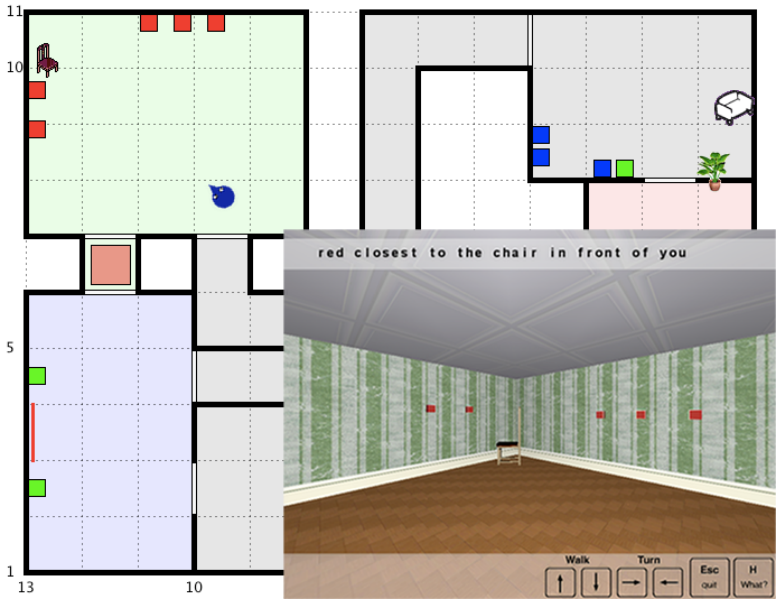

Figura 3.1: "red closest to the chair in front of you".

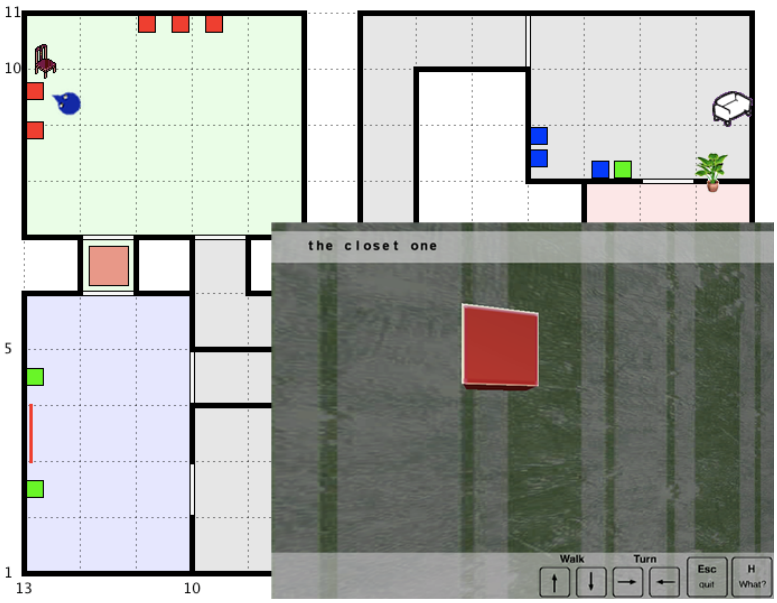

Figura 3.2: “the closet one”.

En la Figura 3.1, podemos ver que el jugador (representado por un punto azul) esta entrando a la habitación de arriba a la izquierda, en la cual hay una serie de objetos: una silla y cinco botones rojos, dos a la izquierda de la silla y tres a la derecha. El objetivo actual del plan es oprimir el boton rojo más cercano a la silla. La primera instrucción generada por el instructor es “red closest to the chair in front of you” (“el rojo más cercano a la silla en frente tuyo”). Luego de recibir la instrucción, el usuario se aproxima al boton, como puede verse en la Figura 3.2. Como resultado de esta nueva posición, existe un nuevo plan, por lo que se genera un nuevo conjunto de candidatos y el sistema elige “the closet one” (“el más cercano”, con un error de ortografía). Lo primero que debemos notar es cómo los errores del corpus son transparentes al sistema. Debido a la anotación automática, y a que el sistema trabaja sobre la semántica asignada, sin preocuparse de ninguna forma de la sintaxis de la instrucción, este tipo de errores nunca son corregidos.

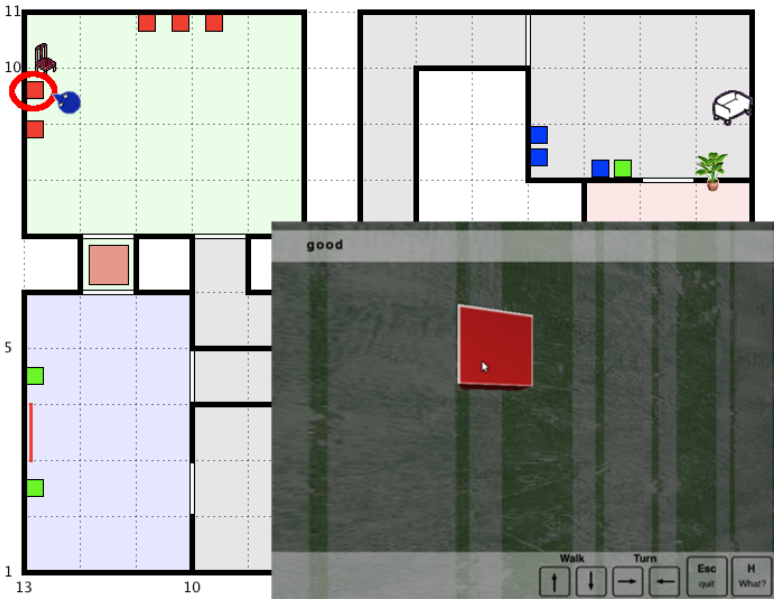

Figura 3.3: “good”.

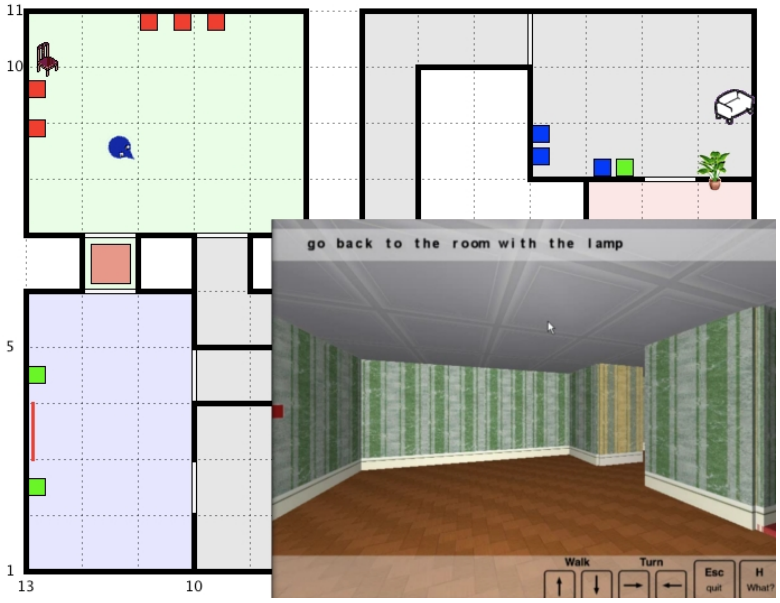

Figura 3.4: "go to the room with the lamp".

Inmediatamente después de que el usuario hace click en el botón, el sistema selecciona otra oración, correspondiente al nuevo plan. Como la posición del jugador no cambió, la única diferencia en el plan es que el botón ya no necesita ser presionado. Como se ve en la Figura 3.3, es muy común que en un sistema orientado a tareas el instructor humano proporciona un feedback (positivo o negativo) sobre su accionar, esto causa la existencia de instrucciones que no tienen contenido semántico significativo, como "good" ("bien"), que es la que UCM retornó en el ejemplo.

Luego de recibir este feedback positivo, el usuario se dio vuelta y caminó hacia adelante, y la nueva acción en el plan es dejar la habitacion (ver Figura 3.4). La instrucción seleccionada es "go back to the room with the lamp" ("vuelve a la habitación con la lámpara"). Esta oración guarda una referencia a una interacción anterior. Aunque parezca extraño, este enunciado es válido para el plan actual incluso si nunca se hubiera visitado tal habitación previamente, ya que es uno de los candidatos posibles para el plan actual. Esto ocurre porque el sistema no guarda representaciones de acciones pasadas del jugador, con lo cual no es posible muchas veces mantener la coherencia global en la interacción.

L	go
yes	left
straight	now go back
go back out	now go back out
closest the door	down the passage
go back to the hallway	nowin to the shade room
go back out of the room	out the way you came in
exit the way you entered	ok now go out the same door
back to the room with the lamp	go back to the door you came in
Go through the opening on the left	okay now go back to the original room
okay now go back to where you came from	ok go back again to the room with the lamp
now i ned u to go back to the original room	Go through the opening on the left with the yellow wall paper

Cuadro 3.1: Paráfrasis candidatas en el momento de abandonar la habitación en la Figura 3.4.

En el Cuadro 3.1, podemos ver un listado de todos los candidatos posibles cuando se deja la habitación en la Figura 3.4. Para el sistema, cada uno de estos es una paráfrasis de la instrucción seleccionada. Como se explicó antes, UCM elige en primer lugar el candidato cuya reacción es más larga. Es posible ver la gran variedad de opciones muy diferentes que componen el conjunto de candidatos, desde instrucciones cortas que sólo hacen referencia a una dirección ("left", "right"), hasta otras que hacen referencia incluso a las primeras interacciones ("okay now go back to the

original room").

Es notable el hecho de que, al no considerar la orientación del usuario, UCM lista como candidatos algunas oraciones que son incorrectas para el contexto actual. Tal es el caso de "left", ya que desde el punto de vista actual del jugador, no se debe girar, sino que se tiene que avanzar hacia adelante.

3.3.2. Evaluación local del sistema

UCM fue evaluado por 13 sujetos de prueba que utilizaron el sistema para resolver la busqueda del tesoro sobre uno de los mapas de la competencia GIVE-2. El sistema se compara con un grupo de instructores humanos, y con los tres mejores sistemas (basados en generación por composición) que participaron en la competencia (NA, NM y Saar). En dicha competencia, el usuario recorría distintas habitaciones buscando un tesoro escondido. Botones interactivos distribuidos en el entorno podían abrir puertas y desactivar alarmas ubicadas en el suelo de las habitaciones. El jugador ganaba la partida si podía encontrar el tesoro, mientras que perdía si activaba alguna alarma a lo largo del recorrido. A continuación describiremos las métricas utilizadas en la evaluación, y presentaremos los resultados.

Métricas objetivas

Las metricas objetivas comparadas fueron las siguientes:

- Task success: corridas completadas con éxito.
- Task Canceled: corridas no terminadas.
- Task Lost: corridas en las cuales el usuario perdió.
- Mouse actions: Cantidad de acciones mediante el cursor
- Time: promedio de tiempo por tarea (calculado sobre juegos exitosos)
- Utterances: promedio de instrucciones por juego (calculado sobre juegos exitosos)

Métricas subjetivas

Las métricas subjetivas fueron obtenidas a partir del cuestionario GIVE-2, que fue presentado a los usuarios luego de cada juego. Este cuestionario consta de 22 afirmaciones, que cada sujeto de prueba debe calificar dependiendo de su experiencia personal en la evaluación. Su objetivo es evaluar la efectividad, fiabilidad, y naturalidad de las instrucciones, así como la inmersión del jugador en el mundo, y la confianza que brinda el instructor.

Resultados

Los resultados obtenidos por las métricas objetivas se resumen en el Cuadro 3.2.

	Human	NA	Saar	NM	UCM
Task Success	100 %	47 %	40 %	30 %	70 %
Task Canceled	0 %	24 %	n/a	35 %	7 %
Task Lost	0 %	29 %	n/a	35 %	23 %
Time (sec)	543	344	467	435	692
Mouse Actions	12	17	17	18	14
Utterances	53	224	244	244	194

Cuadro 3.2: Resultados de las métricas objetivas.

Puede observarse que UCM obtiene mejores resultados que todos los demás sistemas en cuanto al éxito de la tarea. Ademas, dado el bajo número de acciones de mouse requeridas, se puede concluir que el sistema ayuda a los usuarios a identificar mejor los objetos que necesitan manipular dentro del mundo. Esto se correlaciona con los resultados de las métricas subjetivas. Es notable también que mientras UCM obtuvo los mejores resultados en cuanto a éxito, también utilizó el menor promedio de instrucciones por juego.

De las 2081 instrucciones que el agente emitió durante las 13 partidas de evaluación, 1304 (63 %) fueron exitosas, es decir, se obtuvo la reacción esperara por parte del usuario. Las restantes 777 (37 %) fallaron. La mayoría de los fallos se debieron a problemas derivados de errores de orientación (40 %) e instrucciones de corrección dentro del corpus (25 %).

Dentro del cuestionario que representa las métricas subjetivas, 14 de los 22 ítems tienen como objetivo evaluar la efectividad y fiabilidad de las instrucciones. UCM obtiene resultados iguales o apenas menores que los otros sistemas en todas estas preguntas, menos en las tres mostradas en el Cuadro 3.3.

NA	Saar	NM	UCM
Q5: I was confused about which direction to go in			
29	5	9	-12
Q6: I had no difficulty with identifying the objects thesystem described for me			
18	20	13	40
Q22: I felt I could trust the system's instructions			
37	21	23	0

Cuadro 3.3: Resultados significativos de las métricas subjetivas.

Los bajos índices obtenidos en los items 5 y 22 pueden estar relacionados a las instrucciones errónea, debido a problemas de orientación o a correcciones. Los valores inesperadamente altos en el ítem 6, que evalúa indirectamente la calidad de las expresiones referenciales, demuestran la eficiencia del proceso de referenciado del sistema, aun cuando el algoritmo de UCM no trabaja en forma directa en la creación de este tipo de instrucciones.

El resto del cuestionario evalúa la naturalidad de las instrucciones y la inmersión del usuario en la tarea a solucionar. En estos ítems, UCM fue calificado como el instructor que genera mejor disposición para realizar la tarea y el que emite instrucciones de manera más natural.

3.3.3. Análisis de los resultados obtenidos en GIVE-2.5

En GIVE-2.5, este trabajo fue presentado bajo en nombre de CL. La competencia se llevó a cabo entre el 21 de julio y el 29 de agosto de 2011. Durante este periodo, se obtuvieron cerca de 500 juegos válidos, evaluando 8 sistemas de forma online, en 3 entornos virtuales distintos.

Ademas de CL, los sistemas evaluados incluyeron 4 basados en estrategias de composición (A, C, L y T), uno que utilizó árboles de decisión basados en un corpus humano anotado (B) y dos que utilizaban un corpus para predecir la comprensibilidad de las expresiones referenciales (P1 y P2).

Las métricas objetivas de evaluación fueron las siguientes:

- Task success: corridas completadas con éxito.
- Duration: duración de un juego.
- Distance: distancia recorrida en un juego.
- Actions: cantidad de manipulaciones de objetos.
- Instruccions: cantidad de instrucciones emitidas.
- Words: cantidad de palabras usadas por el sistema.
- Error rate: errores totales sobre cantidad total de acciones.
- Speed: distancia total sobre cantidad total de tiempo.
- Instruction speed: cantidad total de instrucciones sobre cantidad total de tiempo.
- Words per instruction: Longitud de las instrucciones en cantidad total de palabras
- Word rate: Cantidad total de palabras sobre cantidad total de tiempo.

Ademas, también se dio a los jugadores un cuestionario al final de cada juego, para evaluar las métricas subjetivas.

Resultados

En líneas generales, los resultados ubican a los sistemas C, CL, L, P1, P1 y T por sobre A y B, siendo C y L los mejores calificados en las métricas subjetivas. En el Cuadro 3.4, se listan los resultados de las principales métricas objetivas.

	A	B	C	CL	L	P1	P2	T
Task success	42 %	32 %	70 %	58 %	68 %	66 %	65 %	58 %
Error rate	21 %	49 %	10 %	11 %	12 %	9 %	15 %	19 %
Duration	687	701	538	539	341	407	415	480
instructions	165	281	254	183	211	241	235	160

Cuadro 3.4: Resultados de las métricas objetivas.

En cuanto al sistema CL, este obtuvo un 58 % de éxito en sus jugadas, y fue uno de los sistemas con menor promedio de instrucciones generadas y menor promedio de error por parte de los usuarios.

El cuadro 3.5 muestra las métricas subjetivas mas notables para CL.

	A	B	C	CL	L	P1	P1	T
Q1 - "Overall, the system gave me good instructions".	-18	-31	54	24	47	31	10	-3
Q2 - "I was confused about which direction to go in".	-22	-16	52	27	31	26	16	17
Q3 - "I could easily identify the buttons the system described to me".	37	3	60	46	42	39	16	23
Q6 - "The system's instructions came too late or too early".	-6	-10	36	-3	34	24	19	2
Q7 - "The system immediately offered helpwhen I was in trouble".	-13	1	52	17	38	48	35	1
Q10 - "I felt I could trust the systemś instructions".	0	-25	69	38	52	44	30	12

Cuadro 3.5: Resultados de las métricas subjetivas.

Evaluando las métricas subjetivas, los usuarios consideraron que en general, CL dio buenas instrucciones, de una manera clara (Q1 y Q2) y que hacia buen uso de las expresiones referenciales (Q3). Además, el sistema obtuvo buen puntaje en cuanto a su confiabilidad (Q10).

Sin embargo, el agente no estuvo entre los mejores clasificados respecto a la asistencia al usuario cuando éste estaba perdido (Q7), y al manejo de tiempos a la hora de dar instrucciones (Q6).

3.4. Conclusiones

En este capítulo se estudió a UCM, un sistema que genera instrucciones en lenguaje natural, utilizando un método de selección sobre un corpus de interacciones humanas anotado automáticamente. En nuestro trabajo, deseamos aplicar las mismas técnicas, en un entorno diferente. El objeto de este análisis, fue considerar qué características nos interesa implementar en nuestro agente, y qué problemas debemos considerar a la hora de hacerlo. Claramente, los modelos por selección necesitan una base de datos de interacciones específicamente creada en el contexto de la tarea a solucionar, por lo que el GIVE2-Corpus no será útil en este trabajo. Como se vio en la sección 3.3, la mayor parte de los errores de UCM fueron consecuencia de dos características: en primer lugar, la falta de consideración de la orientación del jugador, ya que incluye instrucciones equivocadas en el conjunto de candidatos; en segundo lugar, las instrucciones erróneas, y de corrección, que están dentro del corpus, ocasionaron que el jugador recibiera indicaciones

que no se corresponden, semánticamente, con el comportamiento deseado. En tercer lugar, se hace necesario sustituir la instrucción "go" que se emite cuando no existen candidatos en una situación determinada. En este caso, se requeriría implementar una instrucción que brinde más información al usuario sobre lo que debe hacer.

Capítulo 4

Introducción a GRUVE

En este capítulo explicaremos la arquitectura y funcionamiento de GRUVE, framework utilizado en la evaluación del instructor virtual presentado en este trabajo. Primero, en la sección 4.1, analizaremos las motivaciones para crear frameworks sobre los cuales puedan ser evaluados distintos sistemas de navegación en forma online. Luego se dará una breve introducción a Google StretView, herramienta que este framework utiliza para representar virtualmente el entorno. En la sección 4.3 describiremos las características fundamentales de GRUVE, y explicaremos su arquitectura y funcionamiento. En la sección 4.4 se verán mas en profundidad los módulos que conforman el sistema de navegación. Finalmente, en la sección 4.5, se describirá la competencia de evaluación de agentes de navegación propuesta por este framework.

4.1. Evaluación en instructores virtuales

Los sistemas de generación de instrucciones de navegación para peatones son un área de investigación relativamente nueva. Estos sistemas generan instrucciones de dirección verbales para que los usuarios se dirijan desde un punto A hacia otro B dentro del mundo real, o de alguna representación de un mundo virtual. Las técnicas utilizadas varían, desde caminos formados por una única instrucción que contiene toda la información (Google Maps) y sistemas que generan instrucciones de manera incremental (CORAL [Dale et al., 2003]), hasta sistemas de diálogo completamente interactivos (DeepMap [Malaka and Zipf, 2000]).

Uno de los mayores problemas en desarrollo de estos sistemas es la evaluación de los mismos utilizando usuarios en el mundo real. Estas evaluaciones son caras, su organización es complicada y es necesario mucho tiempo para completarlas; además de que deben ser utilizadas durante todo el ciclo de desarrollo del proyecto, no sólo cuando este está finalizado. Consecuentemente, se hace necesaria una plataforma común para comparar efectivamente la performance de diferentes sistemas de navegación verbal desarrollados por diferentes equipos que utilizan variedad de técnicas.

En el capítulo anterior, se estudió un sistema de generación de instrucciones en lenguaje natural que participó en la competencia GIVE [Byron et al., 2007, Koller et al., 2007]. En este capítulo, introduciremos otra plataforma de evaluación de sistemas de generación.

El framework GRUVE, presentado en [Janarthanam et al., 2012], es un sistema basado en una interfaz web que contiene una simulación de un mundo real en el cual los usuarios pueden recorrer las calles de ciudades reales mientras interactúan con diferentes sistemas de navegación. GRUVE provee un agente de navegación propio, llamado *Buddy System*, que puede ser utilizado para probar el framework. A diferencia de GIVE, el objetivo de GRUVE es evaluar sistemas que instruyen a humanos situados en el mundo real o en una simulación de él, como lo es Google StreetView.

Antes de describir más en profundidad este framework, introduciremos StreetView, la herramienta de Google utilizada para generar el mundo virtual en donde se implementará este trabajo.

4.2. Google Street View

Google Street View es una aplicación asociada a Google Maps y Google Earth que proporciona vistas panorámicas en primera persona, permitiendo a los usuarios ver y recorrer de la forma en que lo haría un peatón diferentes partes de las ciudades seleccionadas. Se introdujo en los Estados Unidos en el 2007. Desde su lanzamiento y hasta el 2013, se ha expandido a 31 países europeos, 5 latinoamericanos, 11 asiáticos, 4 africanos y a la Antártida. El usuario interactúa con el sistema mediante los cursores del teclado o el mouse, moviéndose a través de caminos predefinidos. El relevamiento de las distintas ubicaciones se forma a través de fotografías de 360 grados, tomadas mediante vehículos adaptados que recorren las ciudades seleccionadas. Para la toma de fotografías se tiene en cuenta el clima, horario y temperatura, de esa manera se obtienen fotografías parejas. Google provee provee una API, llamada Google Maps API, que permite, mediante el uso de JavaScript, utilizar las funcionalidades de Maps y Street View, y agregarlas a una página web o aplicación externa.

4.3. Arquitectura y diseño de Gruve

A continuación, explicaremos cada módulo que forma parte del framework, y detallaremos cuales estan diréctamente relacionados con nuestro trabajo. La arquitectura del sistema se muestra en la Figura 4.1. El framework consta de los módulos detallados a continuación.

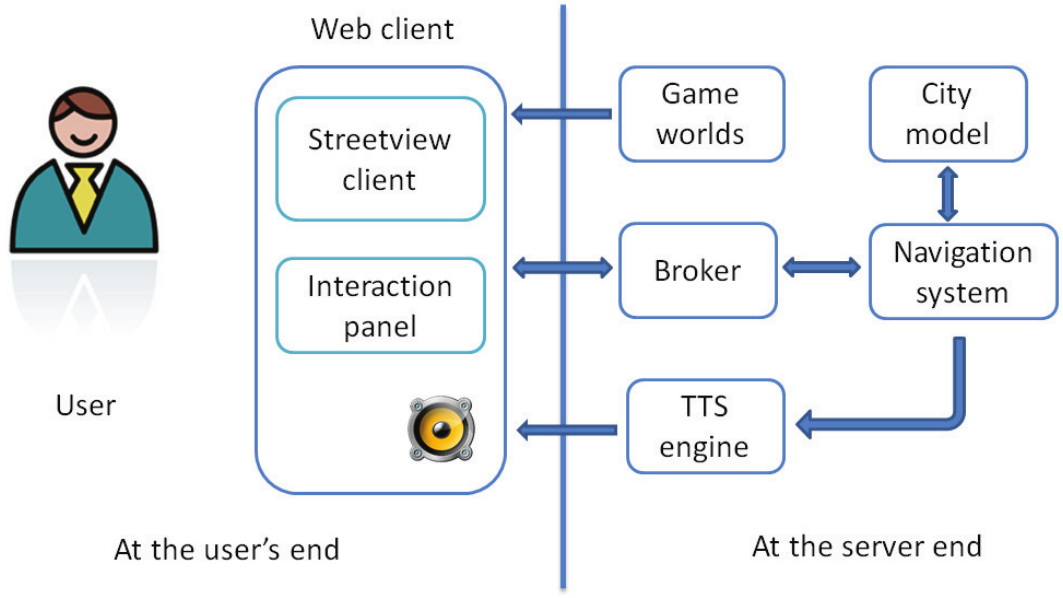

Figura 4.1: Arquitectura de Gruve. Tomado de [Janarthanam et al., 2012].

4.3.1. Game-worlds

El game-world es la representación virtual de la ciudad de Edimburgo que provee Google Street View. La tarea principal de GRUVE es una "caza del tesoro" en la cual los usuarios deben resolver distintas partes de un puzzle para descubrir la localización de un cofre. Para poder completar esta tarea, se interactúa con distintos NPC (del inglés *Non-Player Character*, un personaje controlado por la computadora) y se obtienen indicios sobre la ubicación de la siguiente pista. Esto genera un set de tareas de navegación para obtener distintas pistas hasta encontrar el tesoro. Nuestro agente será diseñado para trabajar sobre los gameworlds de GRUVE.

4.3.2. Broker

Este módulo funciona como un servidor web que conecta a los clientes con su correspondiente sistema de navegación, y se encarga de que el framework pueda ser utilizado por múltiples usuarios simultáneamente. Al conectarse un nuevo cliente, se le asigna uno de los posibles sistemas de navegación. De esta manera, GRUVE se asegura de poder evaluar los sistemas de manera imparcial, ya que el usuario no puede saber cual de los agentes de navegación está utilizando.

4.3.3. Navigation system

El sistema de navegación provee a los usuarios instrucciones para alcanzar su destino. Los diferentes sistemas funcionan como servidores independientes, cuando un cliente se conecta al server, se crea una instancia de uno de los sistemas de navegación y se le asigna exclusivamente al nuevo usuario. GRUVE implementa su sistema de navegación propio llamado *Buddy System*, que está basado en templates. Este módulo es el más importante en relación a nuestro trabajo, ya que nuestro objetivo principal es implementar otro sistema de navegación, basado en generación por selección.

4.3.4. TTS engine

El sistema de navegación puede transmitir información al usuario de manera hablada. Para esto, se pueden utilizar dos motores que convierten las oraciones escritas generadas en oraciones habladas. Estos son: Cereproc TTS Engine[1] y Google TTS[2].

4.3.5. City Model

El framework posee una base de datos llamada "City Model", que guarda información sobre el mundo virtual. Esta está basada en una fuente de datos abierta llamada OpenStreetMaps, y consiste de lo siguiente:

- Street network data: un sistema de nodos que representan calles e intersecciones.
- Amenities: cajeros automáticos, baños públicos, etc.
- Landmarks: Otras estructuras representativas, como ser iglesias, restaurantes, etc.

Esta base de datos provee información sobre la ciudad al sistema de navegación.

4.3.6. Web client

El cliente es un programa de JavaScript/HTML que corre sobre un navegador web. En la Figura 4.2 podemos ver su aspecto desde el punto de vista de un usuario. El cliente está formado por dos partes, el panel de streetview y el de interacción. El panel de interacción se muestra mas en detalle en la Figura 4.3.

[1] https://www.cereproc.com/
[2] http://code.google.com/p/jgoogletexttospeech/

Figura 4.2: El cliente de Gruve, tal como lo ve el usuario.

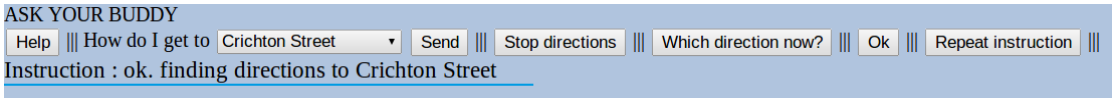

Figura 4.3: Panel de interacción mediante el cual el usuario y el sistema de navegación se comunican.

- Panel de Streetview: presenta una simulación del mundo real al usuario. Utiliza Google StreetView para mostrar una vista panorámica de la ubicación de usuario y de sus alrededores. Sobre esta representación se agrega un gameworld generado por el server, en el cual el usuario puede interactuar con los distintos personajes.
- Panel de Interacción: Mediante este panel, el usuario interactúa con el sistema de navegación. Además de obtener información sobre su objetivo en forma de instrucciones, también se puede interactuar con el sistema de navegación mediante oraciones de texto prediseñadas (como ser “What do I do now?” y “Repeat instruction”.)

4.4. El sistema de navegación

Dentro de GRUVE, el sistema de navegación consta de seis módulos que cumplen diferentes tareas. El objetivo de este sistema es obtener información sobre el mundo virtual y los estímulos de parte del usuario, y generar instrucciones adecuadas, en lenguaje natural. La representación abstracta del lenguaje se maneja mediante *Dialogue Acts o DA's*, estructuras que guardan información sobre lo que el usuario o sistema desean transmitir. En la figura 4.4, podemos ver el flujo de información entre cada uno de los módulos, que se detallan a continuación.

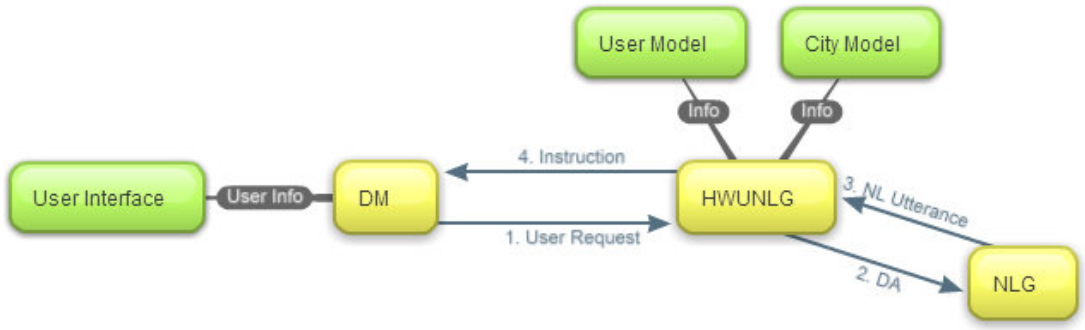

Figura 4.4: Flujo de información en Gruve, desde el usuario hasta el sistema de generación.

4.4.1. Dialogue Management (DM)

Módulo central, es el encargado de manejar las interacciones entre el sistema y el usuario. Recibe como input desde el User Interface la localización del usuario y sus DA's, y las comunica a los módulos de calculo de rutas y generación de lenguaje natural. Una vez que estos generan una instrucción para comunicar al usuario, DM es el encargado de transmitirla.

4.4.2. Route Instruction Generation (HWUNLG)

Módulo que genera informacion sobre las rutas adecuadas para un determinado plan de recorrido en el mundo virtual. HWUNLG se encarga de tomar la parte siguiente del plan y determinar el contenido necesario a transmitir para que el usuario pueda cumplirlo. El algoritmo se encarga de seleccionar la sección del plan que será representada, los *landmarks* a incluir, el manejo del conocimiento previo por parte del usuario, y el posible error en la posición actual de este. Genera una DA que será utilizada por el módulo NLG.

4.4.3. Natural Language Generation (NLG)

Este módulo es el encargado de generar instrucciones en lenguaje natural. Esta tarea se hace a partir de la información proveniente del módulo HWUNLG. En el *Buddy System*, este módulo compara la dirección de la próxima intersección en el plan actual a la orientación del usuario, y utiliza templates anotados manualmente para formar la instrucción que sera transmitida al jugador. En la implementación de nuestro agente, presentada en el capítulo 5, sustituiremos este módulo por uno propio, el cual, partiendo de la misma información, utilizará un algoritmo de selección sobre un corpus de interacciones humanas para generar las instrucciones.

4.4.4. User Model

Este módulo utiliza las trazas GPS de los recorridos de cada usuario para representar el conocimiento de este sobre el diseño de las calles en el mundo virtual. Cada usuario es identificado por el mail que ingresó para comenzar a utilizar la aplicación. El módulo puede responder consultas sobre el conocimiento del usuario sobre el mundo (como por ejemplo, cuantas veces visito un nodo particular). El módulo NLG puede utilizar esta información para adaptarse al conocimiento previo del usuario sobre la ciudad.

4.4.5. City Model

Módulo que representa la ciudad sobre la cual se establece el mundo virtual. Puede responder consultas sobre la red de calles y la ubicación de diferentes negocios especiales. También puede identificar entidades cercanas a una posición, y generar recorridos entre dos puntos dentro de

la ciudad. El módulo HWUNLG utiliza al City Model para diseñar planes de recorrido sobre el mundo virtual.

4.5. La competencia de evaluacion online

Gruve planea organizar una competencia abierta para sistemas de generación de instrucciones para peatones, en la cual varios de estos sistemas puedan ser evaluados. Diferentes grupos de investigación podrán utilizar las diferentes interfaces y módulos para desarrollar sistemas de navegación y a cada equipo se le proveerá un set de herramientas y documentación para utilizar el framework. Se organizará una evaluación web de los sistemas desarrollados para la competencia, en la cual usuarios de todo el mundo se podran conectar al servidor de Gruve y utilizar alguno de los sistemas para completar la busqueda del tesoro en la ciudad de Edimburgo. En esta competencia, se plantearon los siguientes escenarios de evaluacion:

Error en GPS: La ubicación por GPS en los teléfonos actuales genera error [Zandbergen and Barbeau, 2011]. Por lo tanto, un escenario a evaluar es la robustez del sistema frente a errores en la ubicación del usuario.

Modalidades de salida: La salida del sistema puede ser presentada en dos modalidades diferentes: texto y habla. Se considera de interés evaluar estas modalidades en iguales condiciones y comparar los resultados.

Ruido en la voz del usuario: El sistema soporta que el usuario requiera información (como ser un pedido de direcciones, o una aclaración sobre una instrucción dada) mediante comandos de voz. Para sistemas que toman como input pedidos hablados por parte del usuario, se considera importante el manejo de ruido en el canal de comunicación, ya sea por la pérdida de datos debido al medio, o por el ambiente (como ser ruidos de tráfico o del entorno del peatón). Pueden ser evaluados escenarios con diferentes niveles de ruido.

Adaptación a los usuarios: Usuarios que retornan al juego pueden haber adquirido conocimiento sobre el escenario del juego. Es interesante evaluar cómo se adapta el sistema de navegación a usuarios que poseen un mayor conocimiento espacial y visual del mundo presentado.

Capítulo 5

El Instructor Virtual

El instructor virtual implementado en este trabajo, es un agente de lenguaje natural cuya tarea consiste en ayudar a los usuarios a llegar a un destino deseado dentro del mundo virtual. Nuestro método para el desarrollo de un instructor consiste en las dos etapas principales: una fase de anotación y una fase de selección. En este capítulo, describiremos el proceso que se llevó a cabo para diseñar e implementar este instructor. En la Sección 5.1, presentaremos un listado de las principales características que queremos implementar en el agente. Más adelante, en la Sección 5.2 se describe la fase de anotación. Esta se realiza sólo una vez, de forma automática, y consiste en la generación de un corpus formado por asociaciones entre instrucciones y reacciones sobre el mundo virtual. Luego, en la Sección 5.3 se presenta el algoritmo utilizado para realizar la selección de oraciones cada vez que el instructor virtual genera una instrucción, y se analizan las modificaciones aplicadas a las técnicas presentadas por [Benotti and Denis, 2011], para solucionar los problemas señalados en el capítulo anterior. Por último, en la Sección 5.4 se mostrará nuestro sistema funcionando, mediante una partida de ejemplo. Se describirán los conjuntos de candidatos y los criterios de selección de instrucciones; y se explicará la forma de comunicación entre el agente y el usuario.

5.1. Características principales

Las características más sobresalientes que queremos añadir a nuestro agente son:

Expresiones referenciales : Siguiendo los resutados del estudio [Dräger and Koller*, 2012], siempre que pueda, el sistema debe basar sus instrucciones en los *landmarks* disponibles, y en expresiones referenciales acerca del entorno que rodea al usuario. Queremos evitar las oraciones basadas en distancia, similares a un GPS.

Refinamiento del corpus : Buscamos refinar el proceso de creación del corpus mediante una evaluación adicional de todas las instrucciones generadas, para evitar las asociaciones erróneas y las instrucciones de corrección.

Utilización de la orientación : El agente debe considerar la orientación del usuario para seleccionar candidatos.

Resistencia a la falta de datos : Se pretende que el instructor pueda guiar al usuario en la dirección correcta en caso de no contar con candidatos en el corpus.

Conocimiento del DG : Queremos evaluar si el nivel de conocimiento del *Direction Giver* sobre la ciudad, afecta la calidad de sus instrucciones.

5.2. Generación del Corpus de selección

Como se describió en el Capítulo 3, el corpus consiste un registro de interacciones entre dos personas que cumplen diferentes roles: el Direction Giver (DG), que tiene conocimiento sobre cómo llevar a cabo la tarea y crea las instrucciones, y el Direction Follower (DF), que recorre el entorno cumpliendo dichas instrucciones. El mundo en donde se implementará nuestro

instructor será la representación virtual Street View, de Google, la cual se describe en el capitulo 4. En este entorno, no podemos utilizar el GIVE2-Corpus, ya que este fue creado recolectando interacciones entre usuarios mientras estos cumplian con una una tarea diferente a la propuesta en este trabajo, y sobre otro mundo virtual. Por esta razón, debemos primero crear un corpus propio de interacciones entre humanos, para luego aplicarle el algoritmo de anotación.

Para utilizar una forma de anotación automática, necesitamos tres características fundamentales: una representación del mundo virtual en un momento dado, una especificación de cómo funciona ese mundo (un planificador) y una manera de representar el objetivo o tarea a realizar.

5.2.1. Tareas a realizar

Considerando el objetivo principal de GRUVE, que es una búsqueda del tesoro sobre la ciudad de Edimburgo, se tomó como tarea llegar a seis de las principales calles que se utilizarán en la competencia, desde un punto de partida común. Los destinos elegidos se listan a continuación, y en la Figura 5.1 se puede observar su ubicación en la ciudad:

- South College St
- Nicholson St
- Buccleuch Ln
- Windmill St
- Clerk St
- Chambers St

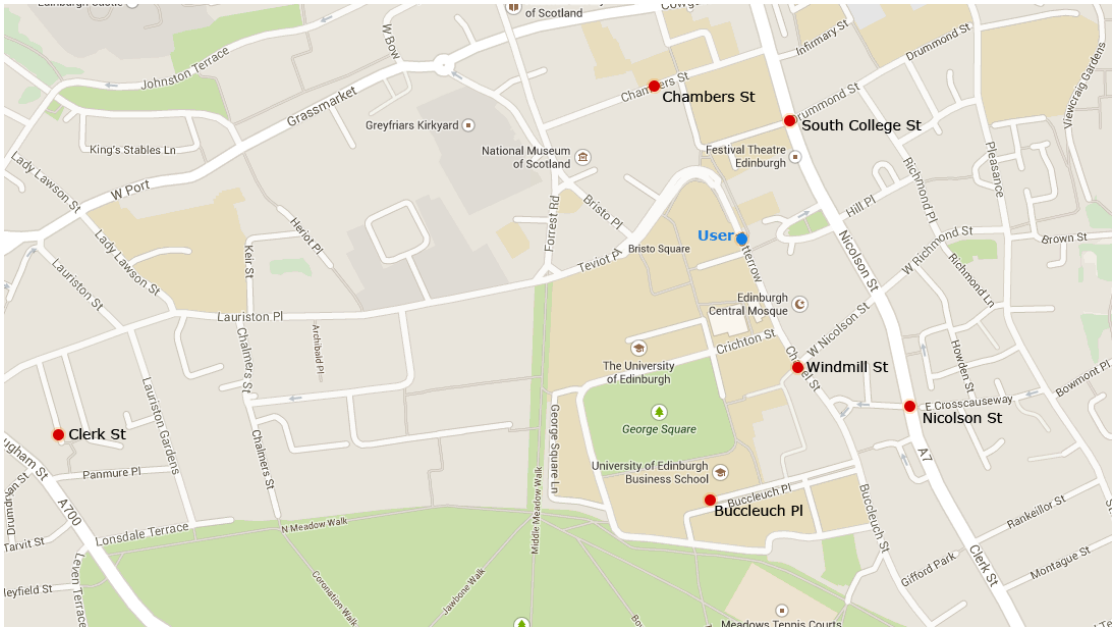

Figura 5.1: La posición inicial, en azul, y los 6 objetivos de la partida.

5.2.2. Planificación y reacción

En el capítulo 3, definimos 'reacción' como los cambios que produce el usuario sobre sí mismo, y sobre el mundo virtual, al ejecutar una instrucción recibida. Esta definición depende de la representación de las acciones posibles en el mundo virtual, y debe ser adaptada a las características del entorno sobre el cual el instructor es implementado. Además, como se vio en el capítulo 3, se debe normalizar la definición de reacción mediante alguna herramienta que permita agrupar los distintos conjuntos de acciones que producen el mismo cambio sobre el mundo.

En este mundo virtual, la representación del movimiento se abstrae mediante un grafo de nodos, en el cual cada nodo representa un cruce entre dos calles de la ciudad. GRUVE utiliza

el planificador de Google Maps, que puede calcular la ruta óptima desde cualquier punto de partida a un destino seleccionado (este plan consiste en la lista de nodos que el usuario debe recorrer para llegar al destino deseado). Como, a diferencia de GIVE2, el DF no puede ocasionar cambios al mundo que le rodea mientras recorre el entorno virtual, esto simplifica el proceso de discretización. De esta manera, los átomos notables en este mundo son:

- user position: latitud y longitud, indicando la posición relativa respecto al mundo.
- user orientation: ángulo entre 0° y 360°, indicando la rotación del punto de vista; además de un valor de *pitch* que indica la altura de la mirada respecto al horizonte.

Considerando estos átomos, podemos definir las reacciones asociadas a cada instrucción de la siguiente manera: *una reacción será la posición y orientación en la cual un usuario se ubica en el mundo virtual en el momento de recibir una instrucción, y su posición y orientación final, luego de completar dicha instrucción.*

Una vez definidas las reacciones y seleccionadas las tareas, podemos crear el corpus de interacciones.

5.2.3. Recopilación del corpus

Para la creación del corpus, se utilizó una versión ligeramente modificada de GRUVE wizards-desk. Esta herramienta provee un mapa de la ciudad, sobre el cuál se pueden calcular planes de recorrido, y está conectada al GRUVE web-client, permitiendo a un usuario humano actuar como DG, generando instrucciones para asistir al DF en la finalización de la tarea y monitoreando su progreso. En la Figura 5.2, podemos ver esta aplicación en funcionamiento. En el panel de la derecha, el DG observa el punto de vista del DF en Google Street View, y puede recorrer la ciudad para estudiar el entorno. Además, sobre la izquierda, se encuentra el plan actual marcado sobre un mapa del mundo virtual. Mediante el panel inferior, el DG envía instrucciones al jugador.

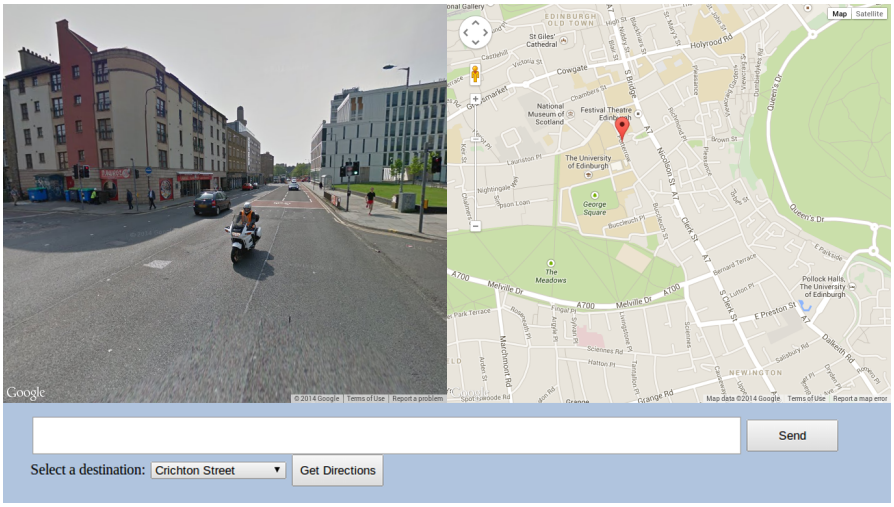

Figura 5.2: Wizard-desk, desde el punto de vista del DG.

Como se discutió en el capitulo 3, en el momento de ejecutar la anotación automática sobre el corpus, instrucciones malinterpretadas y correcciones resultan en asociaciones instrucción-reacción claramente incorrectas. Como queremos minimizar la cantidad de errores dentro del corpus, decidimos que una misma persona cumpla los roles de DG y DF. Aunque esta decisión puede ocacionar perdida de naturalidad en el diálogo, también nos permite eliminar la ambigüedad en la interpretación de instrucción por parte del DF, y eliminar las instrucciones de "corrección" (instrucciones que no son de utilidad para completar la tarea, pero que se hicieron para corregir un error anterior del DG, o una acción incorrecta del DF), por lo que decidimos implementarla.

En total, 14 instructores humanos participaron en la recopilación del corpus. Cada uno completó las 6 tareas cumpliendo con ambos roles, como se explicó en la sección anterior. Se instruyó

a los DG para que nunca utilicen instrucciones basadas en distancia. Estos podían usar Google Maps y Google Street View para observar los tramos faltantes de camino, y elegir expresiones referenciales adecuadas. En caso de no existir *landmarks* sobresalientes, se recomendó hacer referencia a las intersecciones de calles.

En este punto, el corpus está formado por 648 instrucciones, distribuidas entre 84 registros de interacciones (14 usuarios, que completaron 6 tareas cada uno). Estos archivos guardan información sobre cada jugada. A medida que el usuario utiliza GRUVE, los registros son actualizados cada vez que el jugador se mueve o interactua con el agente, compilando una base de datos que guarda el tiempo transcurrido, la posición del jugador, el objetivo actual de la partida, y las instrucciones del DG.

Luego de recopilar y anotar el corpus, se decidió someterlo a una evaluación, para asegurarse de que no existan instrucciones ambiguas dentro del mismo (ver Sección 5.2.5).

5.2.4. Anotación automática

Una vez formado el corpus, se asoció cada instrucción del DG con la correspondiente reacción del usuario, mediante el algoritmo de anotación descripto en Algoritmo 3, análogo al presentado en el capítulo 3.

Cada instrucción generada por un DG fue numerada en orden, en relación a cada tarea. Por ejemplo: si la instrucción dada por el tercer DG, durante la realización de la segunda tarea, fue "Ir hacia adelante y cruzar la plaza", entonces esa instrucción fue numerada de la siguiente forma:

$$32 - \text{“}Ir\ hacia\ adelante\ y\ cruzar\ la\ plaza''.$$

Al ser asignadas las tareas en un orden específico arbitrario para cada DG, esta notación se incluyó para mantener el orden de generación de las instrucciones, necesario para evaluar el efecto del aumento de los conocimientos de la ciudad en la efectividad del DG (esta métrica es explicada en el capítulo 6).

Algorithm 3 Algoritmo de anotación.

```
LOG ← Add player logs
for all Payer P_p ∈ LOG do
  for all Run R_r ∈ P_p do
    for all Utterances U_k ∈ R_R do
      Time_I ← Timestamp in U_k
      Time_f ← (Timestamp in U_{k+1}) − Time_I
      U_k.start ← U_k.position, U_k.orientation
      U_k.finish ← U_{k+1}.position, U_{k+1}.orientation
      U_k.id ← p.r.k
    end for
  end for
end for
```

Veamos el funcionamiento del algoritmo en detalle:

$LOG \leftarrow Add\ player\ logs$

Cada registro (del inglés *log*), de cada jugador, es acumulado manteniendo el orden en el cual fue creado.

$for\ all\ Payer\ P_p \in LOG$
$for\ all\ Run\ R_r \in P_p$
$for\ all\ Utterances\ U_k \in R_R$

Analizaremos cada instrucción generada (*Utterances*), en cada objetivo (*Run*) de cada partida, para cada jugador (*Player*).

$Time_I \leftarrow Timestamp\ in\ U_k$
$Time_f \leftarrow (Timestamp\ in\ U_{k+1}) - Time_I$

Es necesario registrar el tiempo que utilizó el DF en completar cada instrucción. Esto se logra midiendo el tiempo que transcurrió entre cada par de instrucciones consrcutivas emitidas por el DG.

$U_k.start \leftarrow U_k.position,\ U_k.orientation$
$U_k.finish \leftarrow U_{k+1}.position,\ U_{k+1}.orientation$

Como se detalló en la Sección 5.2.2, la reacción asociada esta compuesta por las ubicaciones iniciales y finales del usuario, mientras este ejecuta una instrucción. Por lo tanto, registramos la posición y orientación del usuario (*position* y *orientation*) al momento de recibir una instrucción (U_k), y al momento de recibir la siguiente (U_{k+1}). La posición de un jugador esta representada en coordenadas cartesianas, por lo que está compuesta por valores de longitud y latitud. La orientacción en cambio, consta de un ángulo que indica la dirección de la vista del usuario, y una altura (llamada *pitch*), que indica la distancia de su punto de vista con el horizonte.

$U_k.id \leftarrow p.r$

Finalmente, para cada instrucción a la que asociamos una reacción, se registra el número de identificación del DG que la emitió (p, entre 1 y 14) y el orden en el cual el objetivo, al cual esta instrucción pertenece, fue cumplido por el DG (r, entre 1 y 6).

5.2.5. Evaluación de las instrucciones

Aun habiendo evitado las instrucciones erróneas conocidas, todavía existe la posibilidad de encontrar dentro del corpus interacciones que pueden resultar poco claras debido a la forma de escritura o expresión de los usuarios que actúan como DG. El siguiente método se implementó como recurso para eliminar las instrucciones que pudiesen resultar problemáticas.

Cada instrucción del corpus se recreó en el mundo virtual por 3 usuarios evaluadores cumpliendo el papel de DF. Las distintas reacciones fueron comparadas luego con la reacción original asociada a cada instrucción. Se consideró que, luego de cumplir con una instrucción, el usuario debía terminar a menos de 15 metros de la instrucción original. Este método esta basado en la metodología de aprendizaje por refuerzos [Sutton and Barto, 1998].

La evaluación otorgaba entre 1 y 4 puntos por la correcta utilización de *landmarks* y entre 1 y 5 puntos por la claridad de la instrucción. Cada usuario evaluador podía, en lugar de ejecutar una instrucción, marcarla como "no comprensible". Si una instrucción obtenía 2 marcas de este tipo, era automáticamente removida del corpus. De esta manera, se evaluó si la intención original del DG al crear la instrucción se plasmó en la redacción de la misma.

Para cada tarea, se seleccionaron sólo las instrucciones cuya puntuación superaran un umbral de aceptación de 4 puntos, es decir, las cuales obtuvieron como mínimo 4 puntos entre los 9 posibles (4 por *landmarks* y 5 por claridad). Dentro del corpus se agregó, al identificador *id* de cada instrucción, la calificación obtenida (ver Algoritmo 3). Es decir, si la instrucción "Ir hacia adelante y cruzar la plaza" fue calificada con un 7, quedó registrada de la siguiente forma:

$$327 - \text{"}Ir\ hacia\ adelante\ y\ cruzar\ la\ plaza''.$$

En esta etapa se eliminaron 17 instrucciones, 15 fueron eliminadas por tener interpretaciones ambiguas (no superaron el umbral, o no llegaron al objetivo) y 2 fueron rechazadas al estar marcadas como "no comprensibles".

5.2.6. Corpus final anotado

El corpus final quedó conformado por 631 instrucciones, cada una con un número de identificación que codifica el usuario que la generó, su orden de generación, y el puntaje obtenido.

Cada entrada en el corpus quedó compuesta de la siguiente forma:

text: La instrucción recibida por el DF.
id: Número de identificación único. Codifica el id del usuario que generó la instrucción, y el orden y evaluación de la misma.
time: Tiempo en el cual el DF llevó a cabo la reacción asociada.

start: Posición en la cual el DF recibió la instrucción.
finish: Posición a la cual llegó el DF luego de completar la instrucción.

Abajo, podemos ver un segmento del corpus final utilizado por el algoritmo de selección, escrito en el lenguaje JSON[1].

```
{
 "instrucciones": [
  { "text":  "camina derecho hasta crichton st",
    "id": 115, "time": 131.904,
    "start": {"lat":55.945717, "lng":-3.187079, "ang":150, "pitch":0},
    "finish": {"lat": 55.944725, "lng":-3.186292, "ang": 150, "pitch":0}
  },
  { "text":  "camina una cuadra hasta un quiosco rojo a la derecha",
    "id": 348, "time": 143.148,
    "start": {"lat":55.944725, "lng":-3.1862929, "ang":150, "pitch":0},
    "finish": {"lat": 55.944418, "lng":-3.188129, "ang": -29.4, "pitch":-13.4}
  }
 ]
}
```

En este ejemplo pueden verse dos instrucciones. Observando los números de *id* , podemos notar que "camina derecho hasta crichton st" fue generada por el DG 1 mientras completaba su primer objetivo, y fue calificada con 5 puntos, mientras que "camina una cuadra hasta un quiosco rojo a la derecha" corresponde al DG 3 en su cuarto objetivo, y tiene un puntaje de 8.

En cada instrucción se registró el tiempo (*time*, en segundos) en que fue ejecutada por el DF, y su posición (*lat*,*lng*) y orientación (*ang*, *pitch*) inicial y final. Mediante estas posiciones, se puede calcular que el DF recorrió 120 metros para cumplir la primera instrucción, y 150 metros para la segunda. Además, durante la primera instrucción, el usuario no cambió su orientación entre las posiciones inicial y final, mientras que en la segunda, modificó el ángulo y altura de su punto de vista.

5.3. Algoritmo de Selección

En esta sección se expondrá la implementación realizada en este trabajo, del algoritmo de selección presentado en el capítulo 3. Mostraremos el algoritmo final en la Sección 5.3.1. Luego describiremos puntos claves del método, tales como los criterios de filtrado del conjunto de candidatos (sección 5.3.2), y las distintas formas para seleccionar la instrucción adecuada entre esos candidatos (sección 5.3.3).

5.3.1. Implementación del algoritmo

El algoritmo utilizado por nuestro agente se muestra en Algoritmo 4. El entorno virtual de GRUVE es mucho más extenso que los gamewords de GIVE. Por esta razón, el usuario debe recorrer grandes distancias para cumplir con la tarea actual dentro del plan. Esta característica, sumada a que el jugador no puede introducir cambios en el entorno (como por ejemplo, presionar botones) ocasiona que el plan no necesite ser actualizado cada vez que el DF cambia su posición. En cambio, el plan se revisa en dos situaciones: cada vez que el usuario solicita ayuda mediante el panel de interacción, y cuando el jugador se aproxima a un nuevo nodo (ie: intersección de calles), perteneciente al plan actual o no. El corpus de interacciones utilizado se encuentra en una base de datos JSON, que es cargada al iniciarse el instructor. Este algoritmo es ejecutado cada vez que el agente necesita generar una nueva instrucción.

Para implementar una de las características que queríamos en nuestro sistema, se modificó el comportamiento del algoritmo cuando no se encuentran candidatos viables. Si el conjunto de

[1] http://json.org/

candidatos C es vacío, el algoritmo genera una instrucción genérica que indica en qué sentido debe avanzar el usuario para llegar al siguiente nodo del plan.

Estas instrucciones se calculan comparando la orientación actual del usuario con la posición del siguiente nodo objetivo, para obtener un ángulo de rotación desde el punto de vista del jugador; y de la calle sobre la cual se encuentra, para obtener el nombre de la intersección a la cual se quiere llegar. A continuación, podemos ver las instrucciones posibles:

- "Continue caminando hacia..."
- "Gire ligeramente a la derecha en..."
- "Gire a la derecha en..."
- "Dar la vuelta hacia a la derecha y caminar hacia"
- "Gire ligeramente a la izquierda en..."
- "Gire a la izquierda en..."
- "Dar la vuelta hacia a la izquierda y caminar hacia"
- "Dar la vuelta y caminar hacia..."

Por último, si el usuario no reacciona ante una instrucción, el sistema espera a ser consultado para emitir una oración nueva. Al utilizar un candidato, este es quitado del conjunto de instrucciones posibles, por lo que en el caso de que el usuario se quede inmóvil, y pida asistencia, el instructor selecciona el siguiente mejor candidato, y no repite instrucciones. Esto ocurre hasta que el conjunto de candidatos se encuentra vacío, momento en el cual es sistema se comporta de la manera descripta arriba.

Algorithm 4 Algoritmo de Selección

$position \leftarrow User.position$
$orientation \leftarrow User.orientation$
$objective \leftarrow selected\ destination$
$mode \leftarrow selected\ mode$
$nextNode \leftarrow plan(position, objective)$
$C \leftarrow getCandidates(corpus, position, orientation, nextNode)$
if $C \neq \emptyset$ **then**
  $utt \leftarrow selCandidate(C, mode)$
else
  $rotation \leftarrow getRelativePosition(orientation, nextNode)$
  $utt \leftarrow generateGenericINS(position, rotation, nextNode)$
end if

Veamos el funcionamiento del algoritmo en detalle:

$position \leftarrow User.position$
$prientation \leftarrow User.orientation$

En primer lugar, obtenemos la posición y orientación del jugador.

$objective \leftarrow selected\ destination$
$mode \leftarrow selected\ mode$

Luego, definimos el objetivo actual. Este depende del destino (de entre los 6 posibles) al cual el jugador se dirige en ese momento. El modo (*mode*) indica que mtodo se usará para seleccionar la instrucción, una vez formado el conjunto de candidatos; este puede ser *random*, que elige al azar sobre todos los elementos del conjunto, o *lategen*, que selecciona el candidato generado con mayor experiencia (ver Seccion 5.3.3).

$nextNode \leftarrow Plan(position, objective)$
$C \leftarrow getCandidates(corpus, position, orientation, nextNode)$

Ahora se busca, mediante el planificador *plan*, el siguiente nodo al cual debe llegar el usuario. Recordemos que para el planificador, un *node* es una intersección entre dos calles. Una vez que tenemos el siguiente paso del plan, la función *getCandidates* forma el conjunto de candidatos posibles en el corpus, a partir de la posición y orientación del jugador, y del nodo objetivo actual. Esta función se ve en detalle en la Sección 5.3.2.

$if\ C \neq \emptyset\ then$

$utt \leftarrow selCandidate(C, mode)$
$else$
$rotation \leftarrow getRelativePosition(orientation, nextNode)$
$utt \leftarrow generateGenericINS(position, rotation, nextNode)$

Finalmente, debemos comprobar si existen candidatos para la situación actual del usuario. Notemos que *utt* guarda la instrucción final que será enviada al jugador. Si el conjunto de candidatos no se encuentra vacío, se utiliza la función *selCandidate* para seleccionar uno, dependiendo del modo seleccionado (esta función se explica en detalle en la Seccin 5.3). Si el conjunto de candidatos esta vacío, entonces el algoritmo calcula el ángulo en el cual debe rotar el usuario su orientación (*rotation*) para estar de frente al objetivo (*nextNode*) y, a partir de ese ángulo, genera una instruccion posible utilizando templates, mediante la técnica explicada al principio de esta Sección.

5.3.2. Filtrado de candidatos

En el momento de generar una instrucción, el sistema cuenta con el objetivo actual del plan, y con la posición y orientación del usuario. La funcion *getCandidates* recorre cada instrucción del corpus, seleccionando las que pueden ser consideradas candidatos. Para considerar como candidato a un elemento del corpus, este debe cumplir dos condiciones: *equivalencia de ubicación* y *equivalencia de reacción*.

1. Equivalencia de ubicación

Para definir dos posiciones como equivalentes, analizamos dos métricas:

Posición: escrita como coordenadas de latitud y longitud. Se considera que dos posiciones son iguales si se encuentran a menos de 15 metros de diferencia. Se utilizó un método de cálculo de distancias sobre puntos en una esfera[2]. En la Figura 5.3 pueden observarse, para un usuario en un punto de la ciudad, el área en la cual los candidatos tienen posiciones válidas.

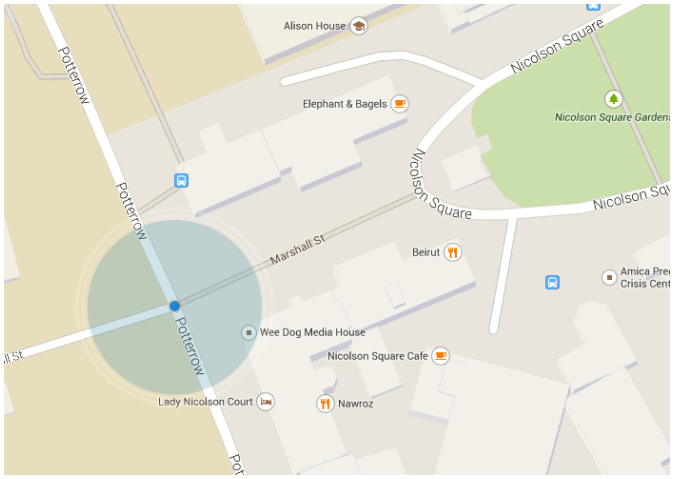

Figura 5.3: La posición del usuario, en azul, y el área válida para los candidatos.

Orientación: La orientación está dada por un ángulo, que forma un vector de visión con respecto al norte cartesiano indicando hacia donde está orientada la vista del usuario; y un número real, que indica la altura del punto de vista del jugador con respecto al horizonte. Se considera que dos orientaciones son iguales si sus vectores de visión no están separados por más de 40 grados. En la Figura 5.4, puede verse área en la cual se consideran equivalentes estos vectores.

[2] http://www.movable-type.co.uk/scripts/latlong.html

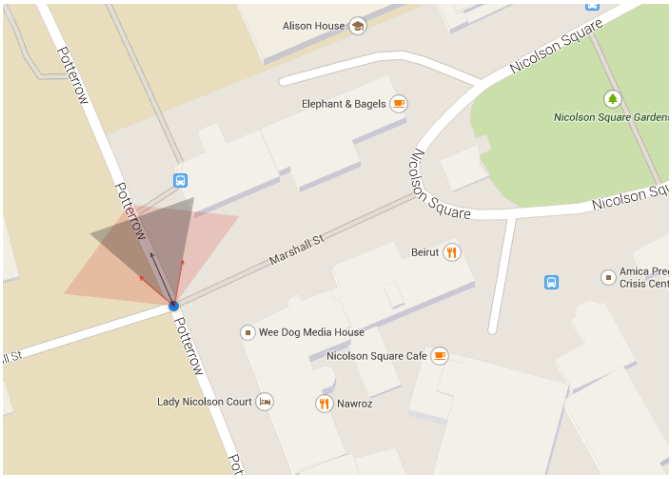

Figura 5.4: El campo de visión del usuario, en negro, y el área de visión considerada equivalente en rojo.

2. Equivalencia de reacción

Para definir dos reacciones como equivalentes, tomamos en consideración primero la posición y orientación iniciales de ambas reacciones, luego, si estas son equivalentes, sólo se verifica que las posiciones finales sean iguales. Esto significa que una instrucción puede ser considerada candidato solamente si la ubicación actual del jugador es equivalente a la ubicación inicial de la reacción asociada, y la posición final de esta, está los suficientemente cerca del objetivo actual del plan. Como las instrucciones están basadas en *landmarks*, es posible que el usuario cambie continuamente su orientación buscando referencias. Decidimos entonces no tomar en cuenta la orientación final de la reacción asociada, ya que consideramos más importante lograr que el usuario se mueva en la dirección correcta, que hacer coincidir su punto de vista final, el cual puede ser corregido mediante la instrucción siguiente. Abajo, podemos ver un nodo de un plan en el mapa, y el área en la cual la posición de las reacciones son equivalentes.

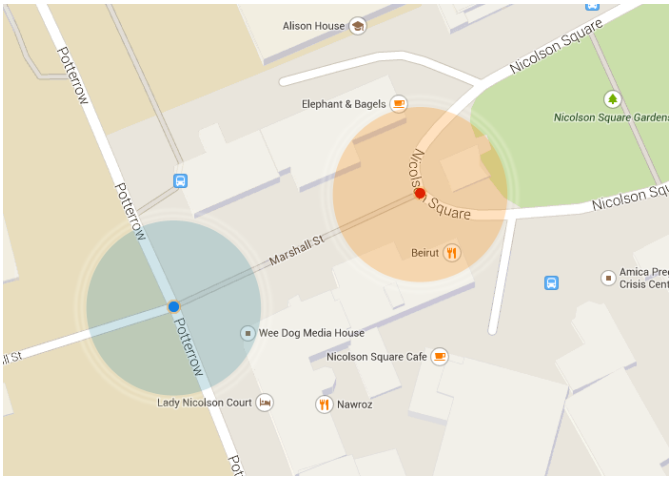

Figura 5.5: La posición del usuario, en azul, el siguiente nodo del plan, en rojo, y el área válida para la posición inicial (azul) y final (rojo) de las reacciones asociadas a candidatos.

5.3.3. Criterios de selección de instrucciones

Una vez formado el conjunto de candidatos, la función *selCandidate* selecciona uno de ellos para transmitir al usuario. En función de la evaluación de la experiencia del DG, se implementaron dos métodos de elección posible entre los candidatos.

El primero (*random*), considera que en este punto todos los candidatos son equivalentemente correctos, y no considera ninguna de sus características para discriminarlos, por lo que elige uno de ellos al azar.

El segundo (*lategen*), selecciona el candidato que fue generado por los usuarios con mayor experiencia en recorrer la ciudad. Cada instrucción posee un **id** diferente, que guarda información sobre el orden en que fueron generadas por el DG, y cuánta experiencia previa poseía éste sobre la ciudad en el momento de emitirla. En el caso de que dos o más candidatos tengan el mismo valor de experiencia previa, se elige el que obtuvo mejor puntaje en la evaluación de instrucciones (ver Sección 5.2.5).

5.4. Ejemplo de ejecución

A continuación presentamos un extracto de una interacción entre nuestro sistema y un usuario humano. Recordemos que el agente está cumpliento el rol de DG, y debe guiar al usuario a través de la ciudad. En cada figura puede observarse el punto de vista del usuario, el panel de interacción, y un mapa de la ciudad con la posición y orientación actual del jugador.

Figura 5.6: "avanza hasta la esquina donde esta el cartel rojo y dobla hacia la izquierda en Marshall".

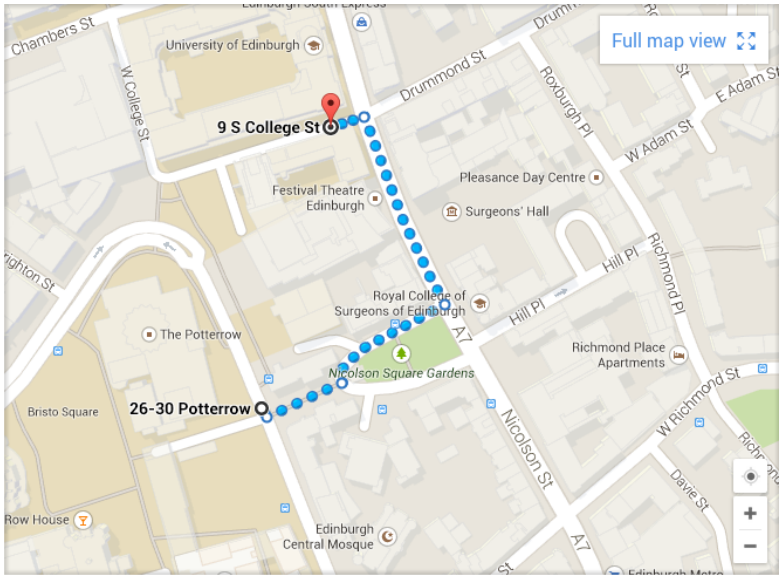

Figura 5.7: Los puntos blancos indican nodos en el plan.

En la Figura 5.6, podemos ver al jugador parado frente a una intersección entre dos calles, en la ciudad de Edimburgo. En la Figura 5.7 podemos ver el plan actual marcado en un mapa. El objetivo actual es llegar a la intersección entre Poterrow St y Marshall St., la primera instrucción generada por el DG es "avanza hasta la esquina donde esta el cartel rojo y dobla hacia la izquierda en Marshall". Luego de recibir la instrucción, el usuario llega a la plaza, como puede verse en la Figura 5.8. Como resultado del movimiento del jugador, éste llega a una nueva intersección, por lo que existe un nuevo plan, y se genera un nuevo conjunto de candidatos. La instrucción ahora es "en la bifurcación tomá hacia la derecha y seguí hasta nicholson st".

Figura 5.8: "en la bifurcación tomá hacia la izquierda y seguí hasta nicholson st".

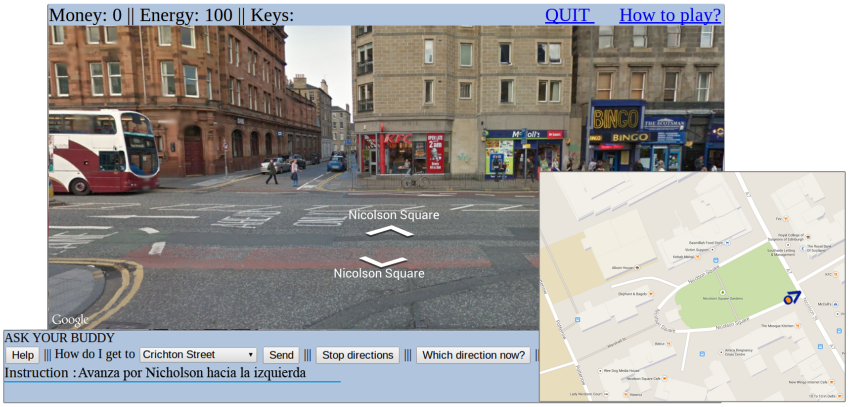

Figura 5.9: "Avanza por Nicholson hacia la izquierda".

Notemos que existen dos caminos que el jugador puede elegir, y que, aunque ambos cumplen el objetivo de llegar a la calle Nicolson, posicionan al GF en lugares distintos. En este caso, aunque fue instruido para seguir por la izquierda, el usuario avanzó por el camino de la derecha, lo que ocaciona un cambio en el plan. Como se puede observar en la Figura 5.9, el sistema elige ahora la instrucción "Avanza por Nicholson hacia la izquierda". En el Cuadro 5.1 se listan todos los candidatos posibles que el agente podía elegir en ese momento. Estos se muestran como aparecen en el corpus, por lo que pueden observarse errores de ortografía y capitalización.

Es interesante notar que, a primera vista, no todas las instrucciones poseen la misma semántica. Dos candidatos, "Avanza por Nicholson hacia la izquierda" y "gira hacia la izquierda", pueden ocasionar comportamientos diferentes en el usuario, sin embargo, ambas poseen una reacción equivalente, por lo que, como se vió en el Capítulo 3, son consideradas paráfrasis pragmáticas. Esta situación se da gracias a la cercanía entre las dos vías que permiten cruzar la plaza. Se puede ver que si el usuario hubiese avanzado por la izquierda hasta llegar a la calle Nicholson,

gira hacia la izquierda
Tome Nicolson hacia la izquierda
cuando llegues a la esquina dobla a la izquierda
Dobla a la izq en el KFC
Gire hacia la izquierda en la calle Nicolson
Gira a la izquierda en la esquina del restoran rojo
cuando llegues nicolson no sigas, dobla a la izquierda
Avanza por Nicholson a la izquierda
Segui por Nicolson, a la izquierda
dobla a la izquierda en el negocio rojo
Dobla sobre esta calle, a la izquierda

Cuadro 5.1: Candidatos al momento de llegar a la intersección en la Figura 5.9.

"gira hacia la izquierda" no sería parte de los candidatos, ya que la posición inicial de su reacción asociada se encontraria demasiado lejos de la posición actual del usuario. Además, en este punto del recorrido se dio una de las instrucciones que luego serían eliminadas en la etapa de evaluación: "Segui por esta hacia la derecha", en donde el DG confundió la dirección del plan. En particular, y como consecuencia de las condiciones necesarias para actualizar el plan (la llegada del usuario hasta un nodo nuevo), si el usuario solamente modifica su punto de vista luego de recibir una instrucción del tipo "gira hacia la izquierda", el sistema no genera una instrucción nueva, hasta que el jugador lo pide explícitamente mediante el panel de interacción.

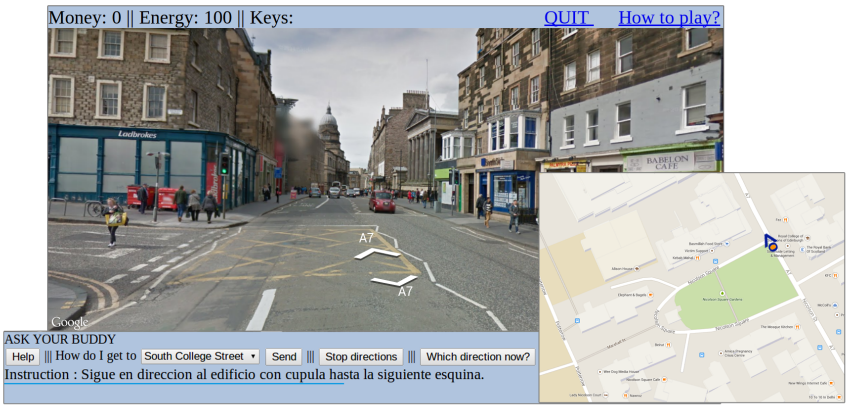

Figura 5.10: "Sigue en dirección al edificio con cúpula hasta la siguiente esquina".

En la figura 5.10, vemos que el usuario avanzó hasta la siguiente intersección, por lo que el sistema seleccionó "Sigue en dirección al edificio con cúpula hasta la siguiente esquina". Como se mencionó anteriormente, la mayoría de los objetivos en GRUVE estan separados por grandes distancias. En este caso, el siguiente nodo del plan esta ubicado a varios metros de distancia, y el sistema no generará nuevas instrucciones hasta que el usuario llegue a destino, o atraviese una intersección que está fuera del plan actual (indicando que tomó una ruta equivocada).

5.5. Características principales implementadas

A continuación sintetizamos las características principales que nos propusimos implementar en nuestro instructor, y analizamos la manera en la que fueron trabajadas.

Expresiones referenciales : Se instruyó a los *Direction Givers* para que, al crear el corpus, utilicen expresiones referenciales que incluyan landmarks y características del entorno siempre que sea posible. Las instrucciones que incluyeron esta característica obtuvieron mejor puntaje en la evaluación a la cual se sometió el corpus. Aunque no se añadieron instrucciones

basadas en distancia, existen en el corpus oraciones que no hacen uso de expresiones referenciales (además de las instrucciones generadas por templates en caso de que no existan candidatos); sin embargo, el algoritmo sólo las usa cuando no existe una alternativa.

Refinamiento del corpus : Se generó el corpus utilizando la misma persona en ambos roles para evitar instrucciones de corrección. Además, se sometió el corpus a un proceso de evaluación, para eliminar las instrucciones ambiguas o mal redactadas. Debe aclararse que esta metodología puede afectar negativamente la naturalidad de las instrucciones, característica que fue evaluada luego con usuarios reales (ver Capítulo 6)

Utilización de la orientación : El agente fue implementado considerando la orientación del usuario en el momento de seleccionar candidatos.

Resistencia a la falta de datos : Se pretendía que el instructor dispusiese de una forma de guiar adecuadamente al usuario en caso de no contar con candidatos para ello en el corpus. Si este es el caso, el agente utiliza un método de generación por templates para orientar al jugador en la dirección correcta.

Conocimiento del DG : Se supuso inicialmente que el nivel de conocimiento de la ciudad del DG mejoraría sus instrucciones. Es por ello que el agente puede ser configurado para seleccionar siempre las instrucciones que cada DG generó con mayor conocimiento (habiendo realizado más recorridos) sobre la ciudad. En la evaluación del sistema, se pondrá a prueba si esta forma de selección mejora la tasa de éxito de los usuarios.

Capítulo 6

Evaluación y Resultados

El instructor virtual implementado, fue puesto a prueba en un entorno real de interacción con usuarios humanos, para observar su comportamiento y analizar si se alcanzaron las metas propuestas en el Capítulo 5.

En este capítulo presentaremos la evaluación llevada a cabo, expondremos cómo fue el proceso de evaluación, los métodos y métricas utilizados, y analizaremos los resultados.

En la sección 6.1 mostraremos los detalles del proceso de evaluación. Describimos las herramientas utilizadas, la tarea a realizar por cada evaluador humano, y los agentes que serán sometidos a prueba. Además, se definen las métricas, tanto objetivas como subjetivas, que fueron utilizadas para comparar los distintos sistemas.

Luego, en la sección 6.2, presentamos los resultados de las pruebas, y establecemos comparaciones entre los resultados obtenidos por cada sistema.

6.1. Proceso de evaluación

En esta Sección se expondrán las características y metodología mediante la cual se evaluó el sistema implementado en este trabajo. En la Sección 6.1.1, presentamos el mundo virtual y las herramientas de software utilizadas. Luego, en la Sección 6.1.2, se describe la tarea que deben cumplir los evaluadores. En la Sección 6.1.3, se listan los sistemas comparados, exponiendo sus principales diferencias, y en las Secciones 6.1.4 y 6.1.5 se detallarán las métricas objetivas y subjetivas utilizadas, respectivamente. Finalmente, en la Sección 6.1.6, describiremos, paso a paso, cómo se desarrollaron las pruebas.

Antes de presentar las distintas métricas, , debemos describir en qué ha consistido la evaluación, como se definió la tarea que cada evaluador debía completar, cómo hemos definido la noción de juego exitoso y juego fallido, y que sistemas compararemos.

6.1.1. El mundo virtual y las herramientas de evaluación

Para evaluar el instructor virtual, se utilizó la plataforma GRUVE, la cual expusimos en el capitulo 4. Esta nos permite probar y comparar nuestro agente con otros sistemas evaluados en un entorno idéntico. El mundo virtual sobre el cual se evaluaron los instructores fue una representación virtual de la ciudad de Edimburgo. En ésta los usuarios pueden recorrer de forma peatonal las distintas calles y plazas, pero no pueden interactuar con la ciudad de ninguna forma, ni entrar a los edificios o negocios.

6.1.2. Descripción de la tarea a completar

Al momento de desarrollar este trabajo, la plataforma GRUVE no se encontraba completamente implementada, por lo que se decidió no utilizar para la evaluación los mismos objetivos que se pretendían a futuro en la competencia homónima. Como se nombró anteriormente en el

capitulo 5, la tarea establecida al recopilar el corpus fue guiar al usuario a llegar a seis puntos de interés, distribuidos a través de la ciudad. Por lo tanto, esta misma tarea será la que deben cumplir los usuarios al evaluar los sistemas. Es claro ver que existe más de una forma posible para llegar a los objetivos planteados; sin embargo, si el usuario ignora o malinterpreta una instrucción, cambiando el recorrido esperado, el plan se adapta inmediatamente a la nueva situación, por lo que se decidió no tomar en cuenta las instrucciones que no son cumplidas, y definir una partida como exitosa cuando en élla el jugador alcanza efectivamente todos los destinos. Se estableció que alcanzar un objetivo es posicionarse a menos de 15 metros del mismo, que es la distancia en la cual dos posiciones se consideran equivalentes.

6.1.3. Sistemas comparados

Para la evaluación, fueron considerados 3 instructores diferentes:

Agente A1: El instructor implementado en este trabajo. Utiliza técnicas de generación de lenguaje natural por selección, y prioriza las instrucciones basadas en *landmarks*. Cuando existe más de un candidato para una situación dada, A1 selecciona uno de manera aleatoria.

Agente A2: Este instructor es similar a A1, excepto en el comportamiento a la hora de seleccionar entre varios candidatos, situación en la cual A2 elige la instrucción que tiene el mayor índice de experiencia. En la recopilación del corpus, se completó cada tarea en orden diferente por cada DG, por lo que se espera que en cada conjunto de candidatos para un plan, las instrucciones generadas en las últimas corridas de cada instructor humano fueron creadas con un mayor conocimiento de la ciudad.

Agente B: El *buddy system*, el instructor de GRUVE. Este agente se comporta como un GPS, y está basado en generación de lenguaje natural por plantillas. Sus instrucciones siempre hacen referencia a la intersección siguiente en el plan, y no utiliza *landmarks*.

6.1.4. Métricas objetivas

Mediante la recolección de información durante las partidas, se midieron distintas métricas objetivas, resumidas a continuación:

- Tareas exitosas: partidas completadas con éxito.
- Tareas fallidas: partidas fallidas.
- Destinos exitosos: objetivos completados con éxito en una partida.
- Destinos fallidos: objetivos que no fueron alcanzados en una partida.
- Distancia: distancia total recorrida en una partida.
- Instrucciones: cantidad de instrucciones emitidas en una partida.
- Palabras: cantidad de palabras usadas por el sistema.
- Tiempo: tiempo total transcurrido en una partida.

Lo que se buscó es obtener datos estadísticos de cada evaluador para poder establecer correlaciones entre las métricas, y luego conocer el promedio y varianza a nivel global de cada una, para poder comparar los sistemas implementados con nustros métodos, A1 y A2, con el agente B, nuestro baseline, que cumple con la misma funcionalidad mediante técnicas diferentes.

6.1.5. Métricas subjetivas

Varios de los atributos que nos interesan evaluar no pueden ser medidos a partir de los datos recolectados durante las partidas de prueba. Estas características, de carácter subjetivo, son evaluadas mediante un cuestionario entregado a cada sujeto evaluador.

El cuestionario está basado parcialmente en el utilizado en la competencia GIVE [Koller et al., 2010], y consta de 19 preguntas. Este formulario pide a los jugadores que califiquen diferentes sentencias acerca del instructor, referidas a la efectividad, fiabilidad, y naturalidad de las instrucciones, así también como a la inmersión del jugador en el mundo, y a la confianza que

brinda el agente. Las preguntas son calificadas con valores comprendidos entre 0 y 10, siendo 0 una respuesta completamente negativa, y 10 una respuesta completamente positiva.

A continuación, listamos el cuestionario completo:

Q1: *El sistema utiliza palabras y frases fáciles de entender.*
Q2: *Tuve que releer las instrucciones para entender que necesitaba hacer.*
Q3: *El sistema me dio información útil sobre mi avance a medida que progresaba el recorrido.*
Q4: *Estuve confundido acerca de lo que debía hacer a continuación.*
Q5: *Estuve confundido acerca de la dirección en la cual debia avanzar.*
Q6: *Tuve problemas para reconocer los objetos, lugares o edificios que el sistema me describia.*
Q7: *El sistema me dio mucha información innecesaria.*
Q8: *El sistema me dio demasiada información al mismo tiempo.*
Q9: *El sistema me ofreció ayuda inmediatamente cuando estuve en problemas.*
Q10: *El sistema me dio instrucciones muy tarde.*
Q11: *El sistema me dio instrucciones muy temprano.*
Q12: *Las instrucciones que me dio el sistema estuvieron bien formadas.*
Q13: *Las instrucciones del sistema sonaron robóticas o inexpresivas.*
Q14: *Las instrucciones del sistema fueron repetitivas.*
Q15: *Perdí la noción del tiempo mientras resolvía la tarea dada.*
Q16: *Disfruté resolviendo la tarea.*
Q17: *Interactuar con el sistema fue dificultoso o molesto.*
Q18: *El sistema se comportó de manera amigable.*
Q19: *Sentí que podia confiar en las instrucciones del sistema.*

Además de este cuestionario, también recolectamos los siguientes datos relativos a cada sujeto de prueba que participó en la evaluación: edad, experiencia en el uso de la computadora y de uso de videojuegos (para evaluar su habilidad para desplazarse en entornos virtuales 3D).

6.1.6. Método de evaluación

En total, 30 personas participaron de forma voluntaria en la evaluación de los sistemas. Cada uno de los agentes fue utilizado por 10 personas diferentes. Cabe aclarar que la única característica común que se buscó en los participantes fue que no tuviesen conocimiento previo de la ciudad de Edimburgo. Esto significa que no se ha normalizado el perfil de el grupo evaluador en cuanto a atributos como edad, sexo, educación, acceso a computadoras o uso de videojuegos, aun cuando estos podrían modificar los resultados obtenidos.

En cada evaluación, se procedió de la siguiente manera: Primero, el jugador ingresa a la plataforma GRUVE mediante un navegador, y se identifica mediante una dirección de correo electrónico. Luego, el sistema elige un instructor al azar, sin notificar al usuario, hasta completar la cantidad deseada de pruebas (10 por agente). Durante la ejecución del programa, el jugador debe llegar a cada objetivo (partiendo desde un lugar común al cúal se retorna automáticamente cuando un destino es alcanzado) en el siguiente orden: South College St, Nicholson St, Buccleuch Ln, Windmill St, Clerk St y Chambers St. El sistema notifica al jugador cada vez que el objetivo actual es alcanzado. Sin embargo, en cualquier momento de la partida, el jugador puede elegir terminar con el objetivo actual; si la posición del jugador al terminar un objetivo se encuentra a mas de 15 metros de éste, el objetivo pasa a considerarse como fallido. Si el jugador falla en llegar a alguno de los seis destinos, se considera fallida la partida. Una vez que el usuario completa o falla todos los objetivos, se considera que su partida fue terminada, y el sistema le entrega el cuestionario, que es completado en el mismo navegador.

6.2. Análisis de los Resultados

En esta sección presentamos los resultados del proceso de evaluación y analizamos las métricas que son de nuestro mayor interés. Además, estableceremos relaciones entre distintas métricas que nos permitirán comprender mejor los resultados. En primer lugar, analizaremos las métricas

objetivas, que fueron presentadas en la sección 6.1.4. Estas nos permiten observar distintos factores y características referidos al funcionamiento de los distintos sistemas comparados. Luego analizaremos los resultados del cuestionario presentado a los evaluadores, las métricas subjetivas, y combinaremos los resultados obtenidos para presentar conclusiones sobre la evaluación.

6.2.1. Descripción de los participantes

Como fue señalado en el punto anterior, participaron en la evaluación 30 sujetos, distribuidos al azar entre los tres agentes. Con el objeto de contar con una apreciación sobre la composición de los tres grupos (asignados a cada agente) en algunas características que pueden ser de interés en relación con el desempeño de la la tarea, se presentan su edad, experiencia de uso frecuente en computadoras y con videojuegos.

Agente	Edad promedio	Uso frecuente de computadoras	Uso frecuente de videojuegos
B	27.6	80 %	70 %
A1	24.9	90 %	80 %
A2	27.4	70 %	80 %
Total	26.6	80 %	76.7 %

Cuadro 6.1: Descripción de los participantes

Como se advierte, en los aspectos presentados, los tres grupos de participantes son similares.

6.2.2. Resultados de las métricas objetivas

En los Cuadros 6.2 y 6.3, pueden verse los resultados obtenidos por las métricas objetivas, listadas en la sección 6.1.4. En las mediciones de distancia y tiempo de juego, así como las de cantidad de instrucciones y de palabras, solo fueron consideradas en las partidas exitosas.

Agente	Tareas exitosas	Tareas fallidas	Destinos exitosos	Destinos fallidos
B	70 %	30 %	91.7 %	8.4 %
A1	80 %	20 %	96.7 %	3.4 %
A2	80 %	20 %	96.7 %	3.4 %

Cuadro 6.2: Resultados de las métricas objetivas.

Agente	Distancia (metros)	Tiempo (segundos)	Instrucciones	Palabras
B	3024.47	972.65	68.86	413.57
A1	2983.13	1223.04	48.75	418.62
A2	3080.87	1236.20	50.00	437.75

Cuadro 6.3: Resultados de las métricas objetivas. Valores promedio para las 10 partidas de cada agente.

Los agentes A1 y A2 no muestran diferencias notables entre ellos, situación que se mantiene constante a lo largo de la evaluación en todos los aspectos considerados. Sin embargo, en cuanto a la tasa de éxito, ambos superan levemente a B, tanto en tareas como en destinos exitosos. Debe ser notado que B, que utiliza un algoritmo de generación de lenguaje natural basado en plantillas, incurrió en errores al nombrar ciertas calles a lo largo del mapa, especialmente cuando estas cambian de nombre en determinadas alturas. Aunque Los agentes A1 y A2 no registraron errores en las instrucciones seleccionadas, un 8.4 % de las mismas (67 en total) fueron instrucciones genéricas (ver capítulo 5) generadas en momentos en los cuales el algoritmo no encontró candidatos posibles. Como se ve en el Cuadro 6.3, no existe gran diferencia entre los 3 agentes en cuanto a la distancia total recorrida. Sin embargo, las pruebas exitosas del agente B, duraron en promedio cerca de 4.5 minutos menos de tiempo en completarse (un 25 % menos), y utilizaron, en promedio, aproximadamente 18 instrucciones más que las de los agentes A1 y A2 (un 16 % más). Estos resultados son acordes a los obtenidos por UCM (ver capitulo 3), que

utilizaba las técnicas en las cuales se basan los agentes A1 y A2. También es notable que los algoritmos A1 y A2, aún emitiendo menos instrucciones, utilizan mayor cantidad de palabras. En la Figura 6.1, podemos observar el tiempo que le tomó a cada evaluador completar la prueba. En la Figura 6.2 vemos la cantidad de instrucciones y la cantidad de palabras respectivamente. Cada gráfico tiene marcado el promedio de cada agente con una línea horizontal, en el cual se nota claramente las diferencias observadas entre el agente B y los agentes A1 y A2.

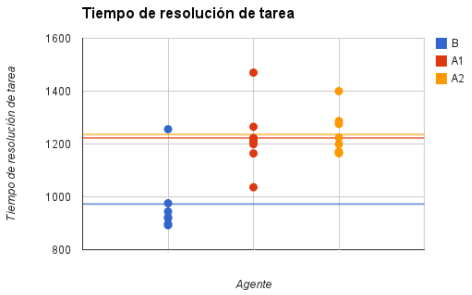

Figura 6.1: Tiempo utilizado por cada agente.

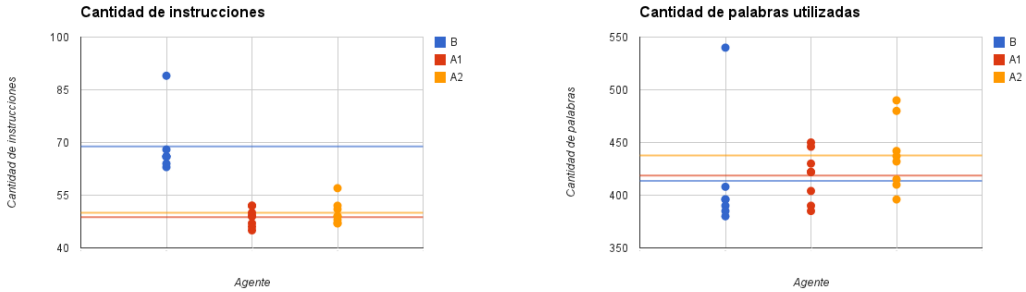

Figura 6.2: Cantidad de instrucciones y palabras por agente.

Una posible explicación para el mayor tiempo y cantidad de palabras utilizadas por los agentes A1 y A2, reside en la utilización de *landmarks*. Mientras que el instructor B utiliza instrucciones prediseñadas de no más de 9 palabras y solo hace referencia a intersecciones de calles, los agentes A1 y A2 utilizan *landmarks* sobresalientes del entorno, y deben generar expresiones referenciales para describirlos. Además, el jugador debe estar constantemente atento al entorno que lo rodea, para lograr identificar estos puntos interesantes, lo que puede derivar en un mayor tiempo de recorrido en las partidas que utilizan estos agentes.

6.2.3. Resultados de las métricas subjetivas

En el Cuadro 6.4, se listan los resultados promedio de las respuestas al cuestionario, diferenciados por agente utilizado. Se han remarcado las preguntas cuyas respuestas muestran una asociación con el agente utilizado por el usuario (A1, A2 o B) estadísticamente significativa (χ^2 $p < 0{,}001$, se utilizó la prueba chi-cuadrado descripta en [Blalock, 1979]). Además, aunque no resultan estadísticamente significativas, Q6 y Q12 muestran diferencias similares entre los agentes, en cuanto a mayores problemas para reconocer objetos o edificios, y a la correcta formulación de las instrucciones. En ambas métricas, los agentes A1 y A2 obtuvieron calificaciones apenas por debajo de B. Analicemos estos resultados:

Q4 - "Estuve confundido acerca de lo que debía hacer a continuación"
Q6 - "Tuve problemas para reconocer los objetos, lugares o edificios que el sistema

Agente	Q1	Q2	Q3	**Q4**	Q5	Q6	Q7	Q8	Q9	Q10
B	8.6	4.1	2.5	**5**	3.3	3.1	1.5	1.6	1.7	2.7
A1	8.5	4.2	2.7	**3.4**	3.2	4.1	1.3	1.5	1.6	2.6
A2	8.6	4.3	2.6	**3.5**	3.1	4	1.5	1.5	1.8	2.8

Agente	Q11	Q12	**Q13**	**Q14**	Q15	Q16	Q17	**Q18**	**Q19**
B	2.7	8	**8.1**	**9**	2.3	3.3	2.8	**4.9**	**2.4**
A1	2.7	7	**4.1**	**4.7**	2.4	3.2	2.9	**7.6**	**7.2**
A2	2.8	7.2	**4.2**	**4.4**	2.2	3.4	2.9	**7.5**	**7.1**

Cuadro 6.4: Resultados promedio de las métricas subjetivas. Remarcadas las diferencias estadísticamente significativas, $p < 0{,}001$.

me describia"

Q12 - "Las instrucciones que me dio el sistema estuvieron bien formadas"

El promedio muestra que el agente B provocó mayor confusión a la hora de indicar las próximas acciones a seguir. Esto puede ocurrir debido al error entre las instrucciones y los nombres de las calles, como fue explicado en la sección 6.2.2.

El mayor puntaje del agente B en las metricas Q6 y Q12, puede deberse a la utilización de plantillas, anotadas manualmente, para la creación de instrucciones.

Aun cuando el promedio muestra estas diferencias, esta medida es muy sensible a los casos extremos. En los gráficos presentados en la Figura 6.3 resulta claro que estos resultados no son consistentes a lo largo de todas las evaluaciones.

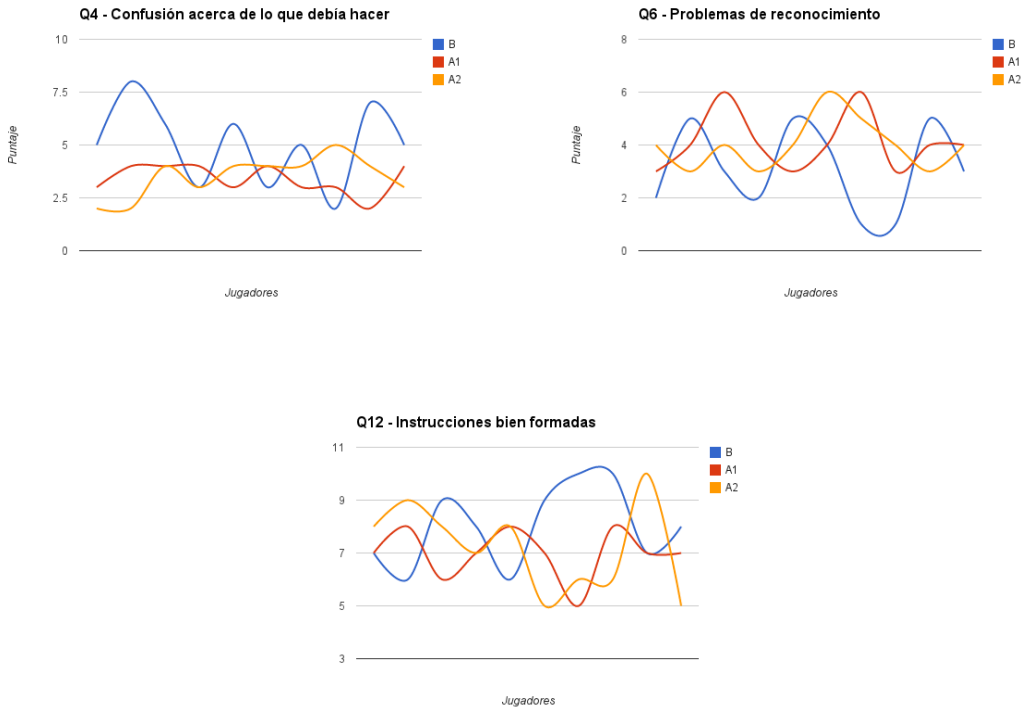

Figura 6.3: Resultados no consistentes a lo largo de las evaluaciones.

Q13 - "Las instrucciones del sistema sonaron robóticas o inexpresivas"

Q14 - "Las instrucciones del sistema fueron repetitivas"

Que las instrucciones del agente B sean suenen mas robóticas y repetitivas que las de los agentes A1 y A2 es una consecuencia directa de los algoritmos que son utilizados para generar

lenguaje natural. El agente B utiliza 9 plantillas anotadas previamente para todas sus instrucciones, por lo que genera mucha menos variación en las oraciones de la que posee el corpus utilizado por los agentes A1 y A2.

Q18 - "El sistema se comportó de manera amigable"
Q19 - "Sentí que podía confiar en las instrucciones del sistema"

Los agentes A1 y A2 recibieron mejores calificaciones en cuanto a su "amigabilidad" y a su confianza. Creemos que la primera se da por la mayor naturalidad que tienen sus instrucciones (a diferencia de la estructura de GPS que presentan las del agente B). El mayor índice de confianza puede ser consecuencia del problema del agente B relacionado al modo de nombrar las calles, expuesto a principio de la sección 6.2.2, situación que probablemente llevó a los jugadores a calificar negativamente la confianza en este agente.

En estos dos últimos grupos de resultados (Q13, Q14, Q18 y Q19), las diferencias en la calificación de los jugadores sí son consistentes a los largo de todas las evaluaciones. Esto puede verse de manera clara en los gráficos presentados en la Figura 6.4. Como se puede apreciar, siempre que el agente B obtuvo un promedio más alto (métricas Q13 y Q14) o mas bajo (métricas Q18 y Q19) que los agentes A1 y A2, esa relación entre las calificaciones se mantuvo estable a los largo de todos los sujetos de prueba, situación que no ocurria con las métricas Q4, Q6 y Q12.

Figura 6.4: Resultados consistentes a lo largo de las evaluaciones.

6.2.4. Relaciones entre los resultados

En esta sección, se cruzan distintos atributos de los estudiados en este trabajo con los resultados de la evaluación en cuanto a tiempo de juego y tasa de éxito de los diferentes agentes. El objetivo de este análisis es advertir si existe alguna asociación entre características del agente o de los sujetos de prueba y los resultados obtenidos.

Entre las características de los sujetos de prueba, se seleccionó la experiencia frecuente de uso de computadoras y videojuegos. De las métricas subjetivas, se utilizaron las que mostraron mayor diferencia, en promedio, entre los agentes evaluados.

Experiencia en computadoras y videojuegos por éxito

Como se ve el Cuadro 6.5, existe una clara relación entre el uso frecuente de computadoras y videojuegos con el éxito de la tarea, indistintamente del agente utilizado. Más aún, mientras que 4 jugadores sin experiencia frecuente en el uso de computadoras solucionaron correctamente la tarea, solamente 1 jugador sin experiencia en uso de videojuegos pudo hacerlo.

	En computadoras				En Videojuegos			
	SI		NO		SI		NO	
	Exito	Fallo	Exito	Fallo	Exito	Fallo	Exito	Fallo
B	7	1	0	2	7	0	0	3
A1	7	2	1	0	7	1	1	1
A2	6	1	2	1	8	0	0	2
Total	20	4	3	3	22	1	1	6

Cuadro 6.5: Experiencia frecuente relacionada al éxito en la tarea.

Problemas de reconocimiento por tiempo.

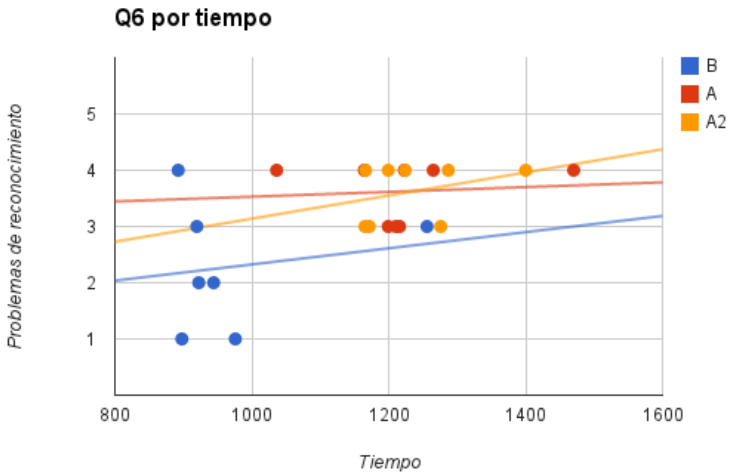

Figura 6.5: Q12-"Tuve problemas para reconocer los objetos, lugares o edificios que el sistema me describia".

Los comparación entre estos resultados mustra que los jugadores con menos problemas para reconocer objetos a los cuales hacían referencia las instrucciones, completaban la tarea en menos tiempo. También puede apreciarse que los resultados del agente B en cuanto a los problemas de reconocimiento estan más dispersos, y que en general son más bajos que los de los agentes A.

Formación de las instrucciones por tiempo.

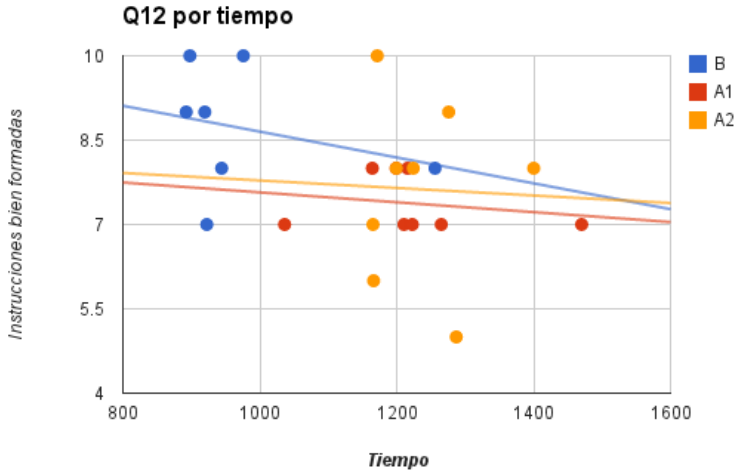

Figura 6.6: Q12-"Las instrucciones que me dio el sistema estuvieron bien formadas".

Este grafico muestra una mayor dispersión entre los resultados. Puede observarse que mientras mejor fueron formadas las instrucciones, las partidas duraron menos tiempo. Aunque esta relación es sutil, es más notoria para el agente B.

Formación de las instrucciones por éxito.

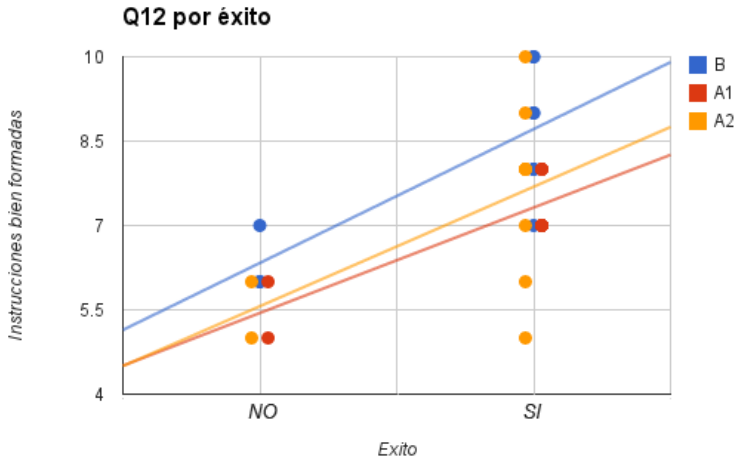

Figura 6.7: Q12-"Las instrucciones que me dio el sistema estuvieron bien formadas".

La relación entre la formación correcta de las instrucciones y el éxito de la tarea se muestra mucho más fuerte que con el tiempo. Es claro en los resultados que las posibilidades de concluir correctamente la tarea de un jugador aumentaban notoriamente si éste entendió mejor las instrucciones.

Agente amigable por tiempo

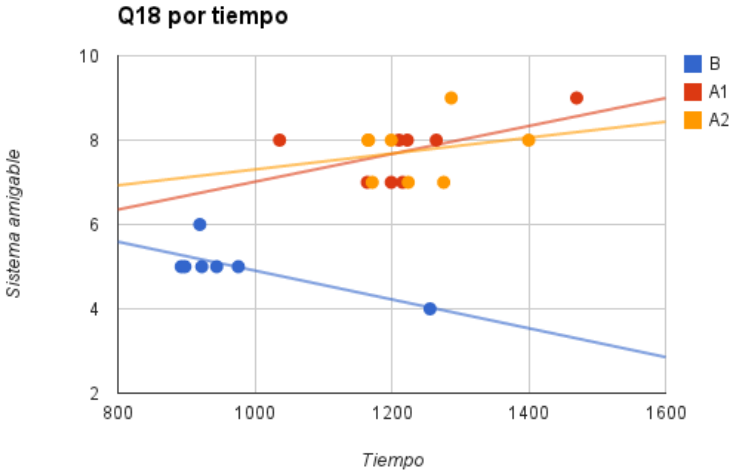

Figura 6.8: Q18-"El sistema se comportó de manera amigable".

Al observar el gráfico 6.8 vemos que los resultados se concentran cerca de los valores promedio, en el caso de los agentes A1 y A2, y que la tendencia de B se ve afectada notoriamente por un caso extremo. Esto nos da a entender que no existe mayor correlación entre la "amigabilidad" del sistema y el tiempo que llevo a los jugadores a completar la tarea.

Confianza por tiempo

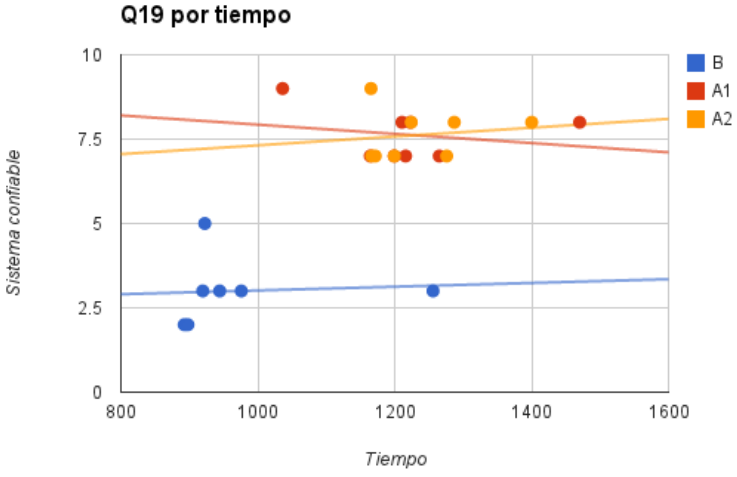

Figura 6.9: Q19-"Sentí que podia confiar en las instrucciones del sistema."

Al igual que en el cruce anterior, la poca distribucion de los resultados, que se concentran cerca de los valores medios, nos hace pensar que tampoco existe una relación observable entre la confianza que tiene el jugador sobre el sistema y el tiempo que le llevó completar la tarea.

Agente amigable por éxito

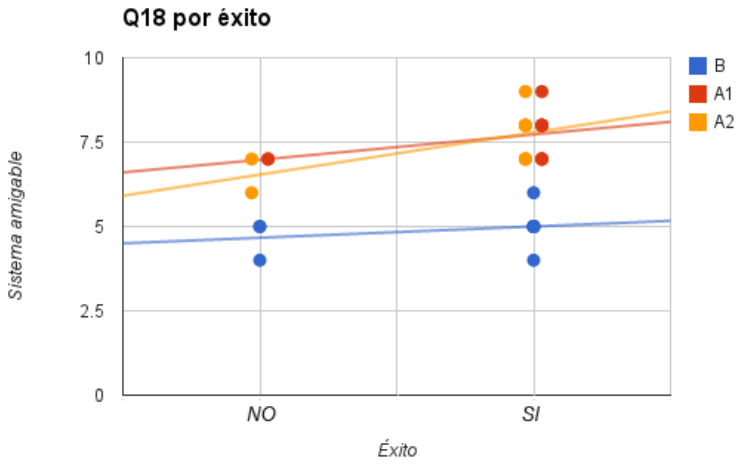

Figura 6.10: Q18-"El sistema se comportó de manera amigable".

Al igual que en la comparación entre la correcta formación de instrucciones y el tiempo de juego, la relación entre un sistema considerado amigable y el éxito de la tarea parece estar presente de una manera muy sutil. Aun así, en este gráfico puede observarse la clara diferencia en la consideración de "amigabilidad" entre los agentes de tipo A, y el B.

Confianza por éxito

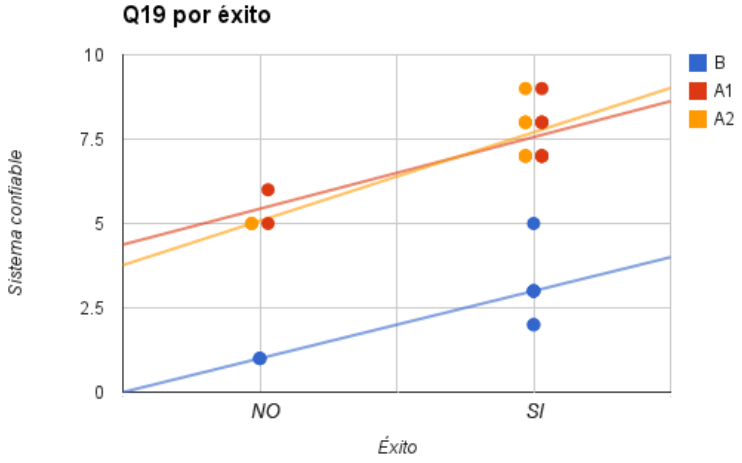

Figura 6.11: Q19-"Sentí que podia confiar en las instrucciones del sistema."

La relación entre la confianza del jugador en el agente y el éxito en la tarea aparece mucho más marcada que la comparación anterior. Como puede verse en el gráfico, todos los jugadores que completaron exitosamente la tarea calificaron su confianza en el agente con un valor mayor o igual a 7, mientras que los usuarios que fallaron nunca asignaron más de 6 a su apreciación en este ítem.

6.3. Conclusiones de la evaluación

Luego de analizar los resultados obtenidos, se pueden enunciar las siguientes conclusiones: En cuanto a la tasa de éxito de los agentes, A1 y A2 obtuvieron mejores resultados que B; además, cometieron menos errores, lo que constituía una de las características que nos propusimos mejorar con respecto al sistema UCM, presentado en el Capítulo 3. Sostenemos que el hecho de haber tomado en cuenta la orientación del usuario, haber recopilado un corpus sin instrucciones de corrección, y haber sometido a ese mismo corpus a un proceso de evaluación, contribuyeron a evitar que nuestro sistema transmita instrucciones incorrectas al usuario. Por otra parte, haber implementado un método muy simple de generación por templates solucionó el problema de tener que emitir una instrucción "go" (que resulta ambigua y puede inducir al usuario a cometer fallas en la partida) cuando se da el caso de no tener información suficiente en el corpus.

Con respecto a algunas cualidades de las instrucciones, notamos que, a pesar de haber creado el corpus (para el caso de los agentes A1 y A2) con una misma persona actuando de *Direction Giver* y *Direction Follower*, no se registraron en la evaluación apreciaciones que hicieran referencia a una menor naturalidad en las instrucciones, con respecto al agente B. Entendemos además, que el uso de expresiones referenciales y de landmarks ayudó a que los sistemas A1 y A2 fuesen mejor calificados no sólo en cuanto a la naturalidad de las instrucciones, sino también respecto a su repetitividad y expresividad.

Por último, debemos destacar que no existieron diferencias notables entre los agentes A1 y A2 en ningún aspecto. El objetivo de estos agentes diferenciados, era evaluar si las instrucciones generadas con mayor conocimiento de la ciudad por parte del DG resultaban más efectivas, lo que, en principio y a partir de los resultados obtenidos, no resulta evidente.

Capítulo 7

Conclusiones y trabajo futuro

En este trabajo, nos propusimos diseñar, implementar y evaluar un sistema de generación de instrucciones de lenguaje natural. En una primera etapa, se estudiaron implementaciones de distintos métodos de generación de lenguaje natural, que utilizan tanto estrategias de composición, por plantillas, como de selección, sobre un corpus humano. Luego de esta etapa, se decidió crear el sistema adaptando las técnicas utilizadas en [Benotti and Denis, 2011] para construir un instructor virtual sobre un mundo virtual diferente, que cuenta con sus propias reglas de interacción, y conjunto de tareas. Para ello fue necesario también recopilar un corpus de interacciones completamente nuevo, y adaptarlo a nuestras necesidades mediante un proceso de anotación, manteniendo la premisa de que este último fuese automático.

Al momento de adaptar las técnicas mencionadas anteriormente, nos propusimos también mejorar ciertos aspectos que, a nuestro criterio, estaban afectando negativamente el funcionamiento del agente. De esta forma, se modificaron métodos referidos a la recopilación del corpus, para eliminar instrucciones que consideramos innecesarias, y para obtener instrucciones lo menos ambiguas posible; de esta manera, se logró formar un corpus sin ninguna instrucción errónea o de corrección, que era uno de los objetivos planteados. Además, se decidió priorizar la utilización de instrucciones que hacen referencia a distintos *landmarks* del entorno a través de expresiones referenciales, ya que en el estudio de trabajos previos esta característica mejoraba la naturalidad y apreciación por parte del usuario de dichas instrucciones. Por último, se clasificaron las interacciones del corpus dependiendo del conocimiento sobre el mundo virtual que asumimos poseían los distintos *Direction Givers* al momento de emitirlas, para observar si las indicaciones que fueron formadas con mayor conocimiento obtenían mejores resultados en situaciones reales.

Una vez formado el corpus, se modificó también el comportamiento del algoritmo de selección, de dos maneras que consideramos podrían mejorar su funcionamiento: en primer lugar, se toma en cuenta la orientación del jugador a la hora de confeccionar grupos de candidatos, para no incluir instrucciones que pueden generar comportamientos incorrectos. Además, se modificó el comportamiento del algoritmo al no encontrar candidatos viables para una situación dada, aplicando una técnica de generación por plantillas que guiaba al usuario en la dirección general del objetivo.

El algoritmo fue evaluado con usuarios reales, y analizando estas pruebas llegamos a la conclusión de que los cambios en el corpus lograron mejorar apreciaciones sobre el agente tales como la confianza del usuario y la comprension general de las instrucciones. Más aún, se pudo observar que el algoritmo de selección utilizó reiteradas veces las instrucciones genéricas creadas ante la falta de candidatos, guiando al usuario en la dirección correcta incluso cuando el corpus no poseía los datos necesarios para hacerlo.

A partir de los resultados obtenidos, no podemos elaborar conclusiones sobre la influencia del conocimiento del DG sobre la ciudad a la hora de generar instrucciones. El agente programado para usar este tipo de instrucciones no mostró diferencias notorias, en ninguno de los aspectos evaluados, con respecto al agente que seleccionó candidatos al azar a la hora de generar instrucciones. Para evaluar de una mejor manera esta característica, creemos que sería necesario contar con dos grupos de DGs humanos, uno que posea un alto conocimiento de la ciudad representada,

y otro que nunca la hayan recorrido, para formar dos conjuntos de instrucciones en donde el grado de conocimiento de la ciudad se encuentre bien representado.

Finalmente, consideramos que sería interesante en el futuro expandir la implementación de este trabajo, para añadir las características necesarias para que sea presentado a la competencia GRUVE, cuando ésta se lleve adelante, ya que los resultados de las evaluaciones locales hacen suponer que el agente obtendría buenos resultados en dicha competencia.

Apéndice A

Trabajo presentado en el *Workshop on Dialogue in Motion*, perteneciente a la conferencia EACL 2014

A Natural Language Instructor for pedestrian navigation based in generation by selection

Santiago Avalos
LIIS Group, FaMAF
Universidad Nacional de Córdoba
Córdoba, Argentina
santiagoe.avalos@gmail.com

Luciana Benotti
LIIS Group, FaMAF
Universidad Nacional de Córdoba
Córdoba, Argentina
luciana.benotti@gmail.com

Abstract

In this paper we describe a method for developing a virtual instructor for pedestrian navigation based on real interactions between a human instructor and a human pedestrian. A virtual instructor is an agent capable of fulfilling the role of a human instructor, and its goal is to assist a pedestrian in the accomplishment of different tasks within the context of a real city. The instructor decides what to say using a generation by selection algorithm, based on a corpus of real interactions generated within the world of interest. The instructor is able to react to different requests by the pedestrian. It is also aware of the pedestrian position with a certain degree of uncertainty, and it can use different city landmarks to guide him.

1 Introduction and previous work

Virtual instructors are conversational agents that help a user perform a task. These agents can be useful for many purposes, such as language learning (Nunan, 2004), training in simulated environments (Kim et al., 2009) and entertainment (Dignum, 2012; Jan et al., 2009).

Navigation agents generate verbal route directions for users to go from point A to point B in a given world. The wide variety of techniques to accomplish this task, range from giving complete route directions (all route information in a single instruction), to full interactive dialogue systems which give incremental instructions based on the position of the pedestrian. Although it can recognize pre-established written requests, the instructor presented in this work is not able to interpret utterances from the pedestrian, leaving it unable to generate a full dialogue. The instructor's decisions are based on the pedestrian actual task, his position in the world, and the previous behavior from different human instructors. In order to guide a user while performing a task, an effective instructor must know how to describe what needs to be done in a way that accounts for the nuances of the virtual world and that is enough to engage the trainee or gamer in the activity.

There are two main approaches toward automatically producing instructions. One is the selection approach, in which the task is to pick the appropriate output from a corpus of possible outputs. The other is the composition approach, in which the output is dynamically assembled using some composition procedure, e.g. grammar rules.

The natural language generation algorithm used in this work is a modified version of the generation by selection method described in (Benotti and Denis, 2011).

The advantages of generation by selection are many: it affords the use of complex and human-like sentences, the system is not bound to use written instructions (it may easily use recorded audio clips, for example), and finally, no rule writing by a dialogue expert or manual annotations is needed. The disadvantage of generation by selection is that the resulting dialogue may not be fully coherent (Shawar and Atwell, 2003; Shawar and Atwell, 2005; Gandhe and Traum, 2007).

In previous work, the selection approach to generation has been used in non task-oriented conversational agents such as negotiating agents (Gandhe and Traum, 2007), question answering characters (Leuski et al., 2006) and virtual patients (Kenny et al., 2007). In the work presented in this paper, the conversational agent is task-oriented.

In Section 2 we introduce the framework used in the interaction between the navigation agent and the human pedestrians. We discuss the creation of the human interaction corpus and the method for natural language generation in Section 3; And in Section 4 we explain the evaluation methods and

Proceedings of the of the EACL 2014 Workshop on Dialogue in Motion (DM), pages 33–37,
Gothenburg, Sweden, April 26-30 2014.

the expected results.

2 The GRUVE framework

One of the major problems in developing systems that generate navigation instructions for pedestrians is evaluating them with real users in the real world. This evaluations are expensive, time consuming, and need to be carried out not just at the end of the project but also during the development cycle.

Consequently, there is a need for a common platform to effectively compare the performances of several verbal navigation systems developed by different teams using a variety of techniques.

The GIVE challenge developed a 3D virtual indoor environment for development and evaluation of indoor pedestrian navigation instruction systems (Byron et al., 2007; Koller et al., 2007). In this framework, users walk through a building with rooms and corridors, and interact with the world by pressing buttons. The user is guided by a navigation system that generates route instructions.

The GRUVE framework presented in (Janarthanam et al., 2012) is a web-based environment containing a simulated real world in which users can simulate walking on the streets of real cities whilst interacting with different navigation systems. This system focus on providing a simulated environment where people can look at landmarks and navigate based on spatial and visual instructions provided to them. GRUVE also provides a embedded navigation agent, the Buddy System, which can be used to test the framework. Apart from the virtual environment in which they are based an important difference between GIVE and GRUVE is that, in GRUVE, there is a certain degree of uncertainty about the position of the user.

Figure 1: Snapshot of the GRUVE web-client.

GRUVE presents navigation tasks in a game-world overlaid on top of the simulated real world. The main task consists of a treasure hunting similar to the one presented in GIVE. In our work, we use a modified version of the original framework, in which the main task has been replaced by a set of navigation tasks.

The web-client (see Figure 1) includes an interaction panel that lets the user interact with his navigation system. In addition to user location information, users can also interact with the navigation system using a fixed set or written utterances. The interaction panel provided to the user consists of a GUI panel with buttons and drop-lists which can be used to construct and send requests to the system in form of abstract semantic representations (dialogue actions).

3 The virtual instructor

The virtual instructor is a natural language agent that must help users reach a desired destination within the virtual world. Our method for developing an instructor consists of two phases: an annotation phase and a selection phase. In Section 3.1 we describe the annotation phase. This is performed only once, when the instructor is created, and it consists of automatically generating a corpus formed by associations between each instruction and the reaction to it. In Section 3.2 we describe how the utterance selection is performed every time the virtual instructor generates an instruction.

3.1 Annotation

As described in (Benotti and Denis, 2011), the corpus consists in recorded interactions between two people in two different roles: the Direction Giver (DG), who has knowledge of how to perform the task, and creates the instructions, and the Direction Follower (DF), who travels through the environment following those instructions.

The representation of the virtual world is given by a graph of nodes, each one representing an intersection between two streets in the city. GRUVE provides a planner that can calculate the optimal path from any starting point to a selected destination (this plan consists in the list of nodes the user must travel to reach the desired destination). As the DF user walks through the environment, he cannot change the world that surrounds him. This simplifies the automatic annotation process, and

the logged atoms are:

- user position: latitude and longitude, indicating position relative to the world.
- user orientation: angle between 0-360, indicating rotation of the point of view.

In order to define the reaction associated to each utterance, it is enough to consider the position to which the user arrives after an instruction has been given, and before another one is requested. Nine destinations within the city of Edinburgh were selected to be the tasks to complete (the task is to arrive to each destination, from a common starting point, see Figure 2). Each pair of DG and DF had to complete all tasks and record their progress.

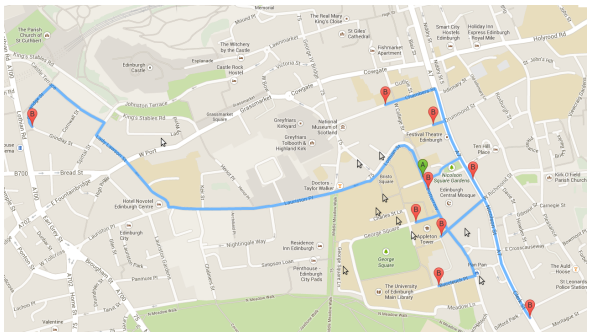

Figure 2: The 9 selected tasks .

For the creation of the corpus, a slightly modified version of the GRUVE wizards-desk was used. This tool is connected to the GRUVE web-client, and allows a human user to act as DF, generating instructions to assist the user in the completion of the task and monitoring his progression. Each instruction generated by a DG was numbered in order, in relation to each task. For example: if the fifth instruction given by the third DG, while performing the second task, was "Go forward and cross the square", then that instruction was numbered as follows:

$$5.3.2 - \text{"}Go\ forward\ and\ cross\ the\ square\text{"}.$$

This notation was included to maintain the generation order between instructions (as the tasks were given in an arbitrary specific order for each DG). With **last-generated**, we refer to the instructions that were generated in the last 3 runs of each DG. This notion is needed to evaluate the effect of the increasing knowledge of the city (this metric is explained in Section 4).

As discussed in (Benotti and Denis, 2011) misinterpreted instructions and corrections result in clearly inappropriate instruction-reaction associations. Since we want to avoid any manual annotation, but we also want to minimize the quantity of errors inside the corpus, we decided to create a first corpus in which the same person portraits the roles of DG and DF. This allows us to eliminate the ambiguity of the instruction interpretation on the DF side, and eliminates correction instructions (instructions that are of no use for guidance, but were made to correct a previous error from the DG, or a wrong action from the DF). Later on, each instruction in this corpus was performed upon the virtual world by various others users, their reactions compared to the original reaction, and scored. For each task, only the instructions whose score exceeded an acceptance threshold remained in the final corpus.

3.2 Instruction selection

The instruction selection algorithm, displayed in Algorithm 1 consists in finding in the corpus the set of candidate utterances C for the current task plan P, which is the sequence of actions that needs to be executed in the current state of the virtual world in order to complete the task. We use the planner included in GRUVE to create P. We define:

$$C = \{U \in Corpus \mid P\ starts\ with\ U.Reaction\}$$

In other words, an utterance U belongs to C if the first action of the current plan P exactly matches the reaction associated to the utterance U. Whenever the plan P changes, as a result of the actions of the DF, we call the selection algorithm in order to regenerate the set of candidate utterances C.

Algorithm 1 Selection Algorithm

$C \leftarrow \emptyset$
$action \leftarrow nextAction(currentObjective)$
for all $Utterance\ U \in Corpus$ **do**
  if $action = U.Reaction$ **then**
    $C \leftarrow C \cup U$
  end if
end for

All the utterances that pass this test are considered paraphrases and hence suitable in the current context. Given a set of candidate paraphrases, one has to consider two cases: the most frequent case when there are several candidates and the possible case when there is no candidate.

- No candidate available: If no instruction is selected because the current plan cannot be matched with any existing reaction, a default, neutral, instruction "go" is uttered.
- Multiple candidates available: When multiple paraphrases are available, the agent must select one to transmit to the user. In this case, the algorithm selects one from the set of the last-generated instructions for the task (see Section 3.1).

4 Evaluation and expected results

Is this section we present the metrics and evaluation process that will be performed to test the virtual instructor presented in Section 3, which was generated using the dialogue model algorithm introduced in Section 3.2.

4.1 Objective metrics

The objective metrics are summarized below:

- Task success: successful runs.
- Canceled: runs not finished.
- Lost: runs finished but failed.
- Time (sec): average for successful runs.
- Utterances: average per successful run.

With this metrics, we will compare 3 systems: agents A, B and C.

Agent A is the GRUVE buddy system, which is provider by the GRUVE Challenge organizers as a baseline. Agent B consists of our virtual instructor, configured to select a random instruction when presented with multiple candidates (see Section 3.1). Agent C is also our virtual instructor, but when presented with several candidates, C selects a candidate who is also part of the last-generated set. As each task was completed in different order by each DG when the corpus was created, it is expected that in every set of candidates, the most late-generated instructions were created with greater knowledge of the city.

4.2 Subjective metrics

The subjective measures will be obtained from responses to a questionnaire given to each user at the end of the evaluation, based partially on the GIVE-2 Challenge questionnaire (Koller et al., 2010). It ask users to rate different statements about the system using a $0\ to\ 10$ scale.

The questionnaire will include 19 subjective metrics presented below:

Q1: *The system used words and phrases that were easy to understand.*
Q2: *I had to re-read instructions to understand what I needed to do.*
Q3: *The system gave me useful feedback about my progress.*
Q4: *I was confused about what to do next.*
Q5: *I was confused about which direction to go in.*
Q6: *I had no difficulty with identifying the objects the system described for me.*
Q7: *The system gave me a lot of unnecessary Information.*
Q8: *The system gave me too much information all at once.*
Q9: *The system immediately offered help when I was in trouble.*
Q10: *The system sent instructions too late.*
Q11: *The systems instructions were delivered too early.*
Q12: *The systems instructions were clearly worded.*
Q13: *The systems instructions sounded robotic.*
Q14: *The systems instructions were repetitive.*
Q15: *I lost track of time while solving the overall task.*
Q16: *I enjoyed solving the overall task.*
Q17: *Interacting with the system was really annoying.*
Q18: *The system was very friendly.*
Q19: *I felt I could trust the systems instructions.*

Metrics Q1 to Q12 assess the effectiveness and reliability of instructions, while metrics Q13 to Q19 are intended to assess the naturalness of the instructions, as well as the immersion and engagement of the interaction.

4.3 Expected results

Based on the results obtained by (Benotti and Denis, 2011) in the GIVE-2 Challenge, we expect a good rate of successful runs for the agent. Furthermore, the most interesting part of the evaluation resides in the comparison between agents B and C. We expect that the different selection methods of this agents, when presented with multiple instruction candidates, can provide information about the form in which the level of knowledge of the virtual world or environment modifies the capacity of a Direction Giver to create correct, and useful, instructions.

References

Luciana Benotti and Alexandre Denis. 2011. Giving instructions in virtual environments by corpus based selection. In *Proceedings of the SIGDIAL 2011 Conference*, SIGDIAL '11, pages 68–77. Association for Computational Linguistics.

D. Byron, A. Koller, J. Oberlander, L. Stoia, and K. Striegnitz. 2007. Generating instructions in virtual environments (give): A challenge and evaluation testbed for nlg. In *Proceedings of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*.

Frank Dignum. 2012. Agents for games and simulations. *Autonomous Agents and Multi-Agent Systems*, 24(2):217–220, March.

S. Gandhe and D. Traum. 2007. First steps toward dialogue modelling from an un-annotated human-human corpus. In *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systemss*.

Dusan Jan, Antonio Roque, Anton Leuski, Jacki Morie, and David Traum. 2009. A virtual tour guide for virtual worlds. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 372–378, Berlin, Heidelberg. Springer-Verlag.

Srinivasan Janarthanam, Oliver Lemon, and Xingkun Liu. 2012. A web-based evaluation framework for spatial instruction-giving systems. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 49–54. Association for Computational Linguistics.

Patrick Kenny, Thomas D. Parsons, Jonathan Gratch, Anton Leuski, and Albert A. Rizzo. 2007. Virtual patients for clinical therapist skills training. In *Proceedings of the 7th International Conference on Intelligent Virtual Agents*, IVA '07, pages 197–210, Berlin, Heidelberg. Springer-Verlag.

Julia M. Kim, Randall W. Hill, Jr., Paula J. Durlach, H. Chad Lane, Eric Forbell, Mark Core, Stacy Marsella, David Pynadath, and John Hart. 2009. Bilat: A game-based environment for practicing negotiation in a cultural context. *Int. J. Artif. Intell. Ed.*, 19(3):289–308, August.

A. Koller, J. Moore, B. Eugenio, J. Lester, L. Stoia, D. Byron, J. Oberlander, and K. Striegnitz. 2007. Shared task proposal: Instruction giving in virtual worlds. In *In Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation.*

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the second nlg challenge on generating instructions in virtual environments (give-2). In *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, pages 243–250. Association for Computational Linguistics.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2006. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, SigDIAL '06, pages 18–27. Association for Computational Linguistics.

David Nunan. 2004. *Task-based language teaching*. University Press, Cambridge.

B.A. Shawar and E. Atwell. 2003. Using dialogue corpora to retrain a chatbot system. In *Proceedings of the Corpus Linguistics Conference*, pages 681–690.

B.A. Shawar and E. Atwell. 2005. Using corpora in machine-learning chatbot systems. *International Journal of Corpus Linguistics*, 10:489–516.

Bibliografía

J. Austin. *How to Do Things with Words*. Oxford, 1962.

S. Avalos and L. Benotti. A natural language instructor for pedestrian navigation based in generation by selection. In *Proceedings of the EACL 2014 Workshop on Dialogue in Motion*, pages 33–37, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

S. Bangalore and O. Rambow. Corpus-based lexical choice in natural language generation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 464–471, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

L. Benotti and A. Denis. Giving instructions in virtual environments by corpus based selection. In *Proceedings of the SIGDIAL 2011 Conference*, SIGDIAL '11, pages 68–77, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

H. M. Blalock. *Social Statistics*. McGraw-Hill, New York, USA, 2nd edition, 1979.

G. Burnett. "turn right at the traffic lights": The requirement for landmarks in vehicle navigation systems. *Journal of Navigation*, 53:499–510, 9 2000.

D. Byron, A. Koller, J. Oberlander, L. Stoia, , and K. Striegnitz. Generating instructions in virtual environments (give): A challenge and evaluation testbed for nlg. In *Proceedings of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*, 2007.

J. Chu-Carroll and B. Carpenter. Vector-based natural language call routing. *Comput. Linguist.*, 25(3):361–388, Sept. 1999.

R. Dale, S. Geldof, and J.-P. Prost. Coral: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian Computer Science Conference*, 2003.

N. Dethlefs, H. Cuayáhuitl, and J. Viethen. Optimising natural language generation decision making for situated dialogue. In *Proceedings of the SIGDIAL 2011 Conference*, SIGDIAL '11, pages 78–87, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

D. DeVault, D. Traum, and R. Artstein. Making grammar-based generation easier to deploy in dialogue systems. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, SIGdial '08, pages 198–207, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

F. Dignum. Agents for games and simulations. *Autonomous Agents and Multi-Agent Systems*, 24(2):217–220, Mar. 2012.

M. Dräger and A. Koller*. Generation of landmark-based navigation instructions from open-source data. In *Proceedings of the Thirteenth Conference of the European Chapter of the ACL (EACL)*, Avignon, 2012.

P. A. Duboue and K. R. McKeown. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 121–128, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

S. Gandhe and D. Traum. First steps toward dialogue modelling from an un-annotated human-human corpus. In *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systemss*, 2007.

C. Gardent and E. Kow. A symbolic approach to near-deterministic surface realisation using tree adjoining grammar. In J. A. Carroll, A. van den Bosch, and A. Zaenen, editors, *ACL*. The Association for Computational Linguistics, 2007.

A. Gargett, K. Garoufi, A. Koller, and K. Striegnitz. The give-2 corpus of giving instructions in virtual environments. In N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

K. Garoufi and A. Koller. Combining symbolic and corpus-based approaches for the generation of successful referring expressions. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 121–131, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *J. Artif. Int. Res.*, 14(1):253–302, May 2001.

C. Hsu and B. W. Wah. New features in sgplan for handling preferences and constraints in pddl3.0. In *In Proceedings of the Fifth International Planning Competition*, pages 39–42, 2006.

D. Jan, A. Roque, A. Leuski, J. Morie, and D. Traum. A virtual tour guide for virtual worlds. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 372–378, Berlin, Heidelberg, 2009. Springer-Verlag.

S. Janarthanam, O. Lemon, and X. Liu. A web-based evaluation framework for spatial instruction-giving systems. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 49–54, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

P. Kenny, T. D. Parsons, J. Gratch, A. Leuski, and A. A. Rizzo. Virtual patients for clinical therapist skills training. In *Proceedings of the 7th International Conference on Intelligent Virtual Agents*, IVA '07, pages 197–210, Berlin, Heidelberg, 2007. Springer-Verlag.

J. M. Kim, R. W. Hill, Jr., P. J. Durlach, H. C. Lane, E. Forbell, M. Core, S. Marsella, D. Pynadath, and J. Hart. Bilat: A game-based environment for practicing negotiation in a cultural context. *Int. J. Artif. Intell. Ed.*, 19(3):289–308, Aug. 2009.

A. Koller, J. Moore, B. Eugenio, J. Lester, L. Stoia, D. Byron, J. Oberlander, and K. Striegnitz. Shared task proposal: Instruction giving in virtual worlds. In *In Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation.*, 2007.

A. Koller, K. Striegnitz, A. Gargett, D. Byron, J. Cassell, R. Dale, J. Moore, and J. Oberlander. Report on the second nlg challenge on generating instructions in virtual environments (give-2). In *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, pages 243–250, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

E. Krahmer and K. van Deemter. Computational generation of referring expressions: A survey. *Comput. Linguist.*, 38(1):173–218, Mar. 2012.

M. Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 545–552, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

A. Leuski and D. Traum. NPCEditor: creating virtual human dialogue using information retrieval techniques. *AI Magazine*, 32(2):42–56, July 2011.

A. Leuski, R. Patel, D. Traum, and B. Kennedy. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, SigDIAL '06, pages 18–27, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

R. Malaka and E. Zipf. Deep map - challenging it research in the framework of a tourist information system. In *Information and Communication Technologies in Tourism 2000*, pages 15–27. Springer, 2000.

R. Malaka, J. Haeussler, and H. Aras. Smartkom mobile: Intelligent ubiquitous user interaction. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, pages 310–312, New York, NY, USA, 2004. ACM.

D. Nunan. *Task-based language teaching.* University Press, Cambridge, 2004.

J. Orkin and D. Roy. Automatic learning and generation of social behavior from collective human gameplay. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 385–392, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

A. Pauzié. Method to evaluate driver's workload in real road context. In *Proceedings of the 2Nd International Conference on Digital Human Modeling: Held As Part of HCI International 2009*, ICDHM '09, pages 453–462, Berlin, Heidelberg, 2009. Springer-Verlag.

M. Raubal and S. Winter. Enriching wayfinding instructions with local landmarks. In *Proceedings of the Second International Conference on Geographic Information Science*, GIScience '02, pages 243–259, London, UK, UK, 2002. Springer-Verlag.

E. Reiter and R. Dale. Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87, Mar. 1997.

E. Reiter and R. Dale. *Building Natural Language Generation Systems.* Cambridge University Press, New York, NY, USA, 2000.

E. Reiter, S. G. Sripada, and R. Robertson. Acquiring correct knowledge for natural language generation. *J. Artif. Int. Res.*, 18(1):491–516, June 2003.

J. R. Searle. *A taxonomy of illocutionary acts.* Linguistic Agency University of Trier, 1976.

B. Shawar and E. Atwell. Using dialogue corpora to retrain a chatbot system. In *Proceedings of the Corpus Linguistics Conference*, pages 681–690, 2003.

B. Shawar and E. Atwell. Using corpora in machine-learning chatbot systems. *International Journal of Corpus Linguistics*, 10:489–516, 2005.

R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.

D. Traum, M. Fleischman, and E. Hovy. NL generation for virtual humans in a complex social environment. In *AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, pages 151–158, Mar. 2003.

M. Walker, A. Stent, F. Mairesse, and R. Prasad. Individual and domain adaptation in sentence planning for dialogue. *J. Artif. Int. Res.*, 30(1):413–456, Nov. 2007.

P. A. Zandbergen and S. J. Barbeau. Positional accuracy of assisted gps data from high-sensitivity gps-enabled mobile phones. *Journal of Navigation*, 64:381–399, 7 2011.