Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía, Física y Computación

Generación de expresiones referenciales bajo incertidumbre con teoría de modelos

Tesis presentada el 29 de julio de 2016
para obtener el

Doctorado en Ciencias de la Computación

Ivana Romina Altamirano

Supervisora

Dra. Luciana Benotti	FaMAF, Universidad Nacional de Córdoba, Argentina

Jurado

Dr. Ivandrè Paraboni	EACH, Universidade de São Paulo (USP), Brasil
Dra. Laura Alonso i Alemany	FaMAF, Universidad Nacional de Córdoba, Argentina
Dr. Raul Alberto Fervari	FaMAF, Universidad Nacional de Córdoba, Argentina

Córdoba, Argentina, 2016

Agradecimientos

En este camino aprendí muchas cosas, conocí gente maravillosa, conocí lugares hermosos. Y quizás todo eso empezó aquel día en que la Dra. Laura Alonso i Alemany me dijo "Romina, querés ir a una conferencia en Los Angeles, EEUU?", y yo respondí "yo?, a Los Angeles?, no no", y ella dijo "Bueno, pensalo, después me decís". Luego de 2 días le dije que síií. Gracias Laura por abrirme las puertas del mundo, y permitirme pensar que yo también puedo hacer esas cosas. Y ahí conocí a la Dra. Luciana Benotti, nunca pensé que recorreríamos este largo camino juntas. También conocí a Jerry Hobbs y con él a mis primeras expresiones referenciales. O quizás empezó aquel día en que decidí anotarme en el doctorado con la Dra. Paula Estrella. Gracias Paula, lamento que no hayamos podido progresar en la traducción automática. Fui a una conferencia en Barcelona y ahí conocí a la Dra. Irene Castellón, y a su alumna de doctorado, divinas!, y hasta me saqué una foto con Alon Lavie un capo de la traducción automática. Después llegó Luciana de Europa y me dio una oportunidad con las expresiones referenciales!. Muchas gracias Lu, por dedicarme tanto tiempo!, por guiarme y conseguir nuevos desafíos para mí cada día. Muchas gracias Dr. Carlos Areces por ayudarnos con el marco teórico y hasta con el código. Gracias Luli!, por las correcciones! (solo para entendidos). Y las aventuras continuaron, con Luciana presentamos un paper y fui a la India en el 2012 ahí conocí a Dr. Manish Shrivastava. Gracias Manish por interesarte en mi trabajo, gracias a todos por tratarme como a una reina!, en la India hasta hablé en ruso, sí, ahí conocí a Sonun Karabeva y a Devendra Sing y con ellos hablabamos en ruso y fuimos al Taj Mahal y a Jaipur. En el 2013 fui a Francia, ahí conocí a la Dra. Craige Roberts, yo estaba embarazada en esa época, compartimos largas charlas en la cena de gala y camino de vuelta al hotel, ella me dijo refiriéndose a mi hija, "ojalá sea bella!, muy bella, porque sana seguro que va a ser" y así es. También conocí a Mattew Stone, que personaje!. Conocí al Dr. Ivandrè Paraboni y su alumno Thiago Ferreyra Castro, muy trabajadores!. Gracias Ivandrè disfruté mucho la estadía en Brasil, hubo mucho trabajo, pero también hubo playa!.

Gracias Laura, Ivandrè y Dr. Raul Fervari por aceptar ser los jurados de mi tesis. Para mi es un honor que sean ustedes.

A mis compañeros de trabajo en la SPGI que supieron tenerme muchos años trabajando en horario reducido. A mi jefe el Sr. Juan Montoya que sin conocerme, confió en mi desde el primer momento, por alentarme y decirme te quiero ver Doctora!. Al Dr. Sergio Obeide y a la SeCyT quienes consiguieron una excepción extraordinaria que me permitió hacer el doctorado siendo no-docente en la Secretaría de Planificación y Gestión Institucional de la UNC.

A mi hija que supo esperar el momento adecuado para llegar al mundo, y darme un recreo en la tesis. A Boris, a mi familia.

Y a todos los que olvidé mencionar!. Gracias a todos!.

Resumen

En esta tesis investigamos la generación automática de rankings de expresiones referenciales en contextos con incertidumbre. Las posibles aplicaciones de la generación de expresiones referenciales que deben referirse al mundo real (por ejemplo, software para robots, sistemas gps, etc.) sufren de incertidumbre por datos ruidosos de sensores y modelos incompletos de la realidad. Extendemos técnicas y algoritmos de teoría de modelos y simulaciones integrando una distribución finita de probabilidades que representa esta incertidumbre. El objetivo es generar un ranking de expresiones referenciales ordenado por la probabilidad de ser correctamente interpretada en el contexto.

En primer lugar, se desarrollaron técnicas y algoritmos de generación de expresiones referenciales que extienden algoritmos clásicos de minimización de autómatas. Los algoritmos de minimización se aplicaron a la caracterización de modelos de primer orden. Dichos algoritmos fueron extendidos usando probabilidades aprendidas de corpora con técnicas de aprendizaje automático. Los algoritmos resultantes fueron evaluados usando técnicas automáticas y evaluaciones de jueces humanos sobre datos de benchmarks del área. Finalmente se recolectó un nuevo corpus de expresiones referenciales de puntos de interés en mapas de ciudades con distintos niveles de zoom. Se evaluó el desempeño de nuestros algoritmos en este corpus relevante a aplicaciones sobre mapas del mundo real.

- **Clasificación de Biblioteca: CCS, Computing methodologies, Artificial intelligence, Natural language processing, Natural language generation**

- **Palabras Clave:**
  expresiones referenciales, aprendizaje automático, simulaciones, evaluación, corpus, teoría de modelos.

Abstract

In this thesis we investigate the automatic generation of referring expression rankings in contexts under uncertainty. The potential applications for the automatic generation of referring expressions that need to refer to the real world (e.g. robot software, gps systems, etc) suffer from uncertainty due to noisy sensor data and incomplete models. We extend tecniques and algorithms from model theory with finite probability distributions that represent these uncertainties. Our goal is to generate rankings of referring expressions ordered by the probability of being interpreted successfully.

First, we developed techniques and algorithms for generating referring expressions that extend classical algorithms for automata minimization applied to first order model characterizations. Such algorithms were extended using probabilities learned from corpora applying machine learning techniques. The resulting algorithms were evaluated using automatic metrics and human judgements with respect to benchmarks from the area. Finally, we collected a new corpus of referring expressions of interest points in city maps with different zoom levels. The algorithms were evaluated on this corpus which is relevant to applications that use maps of the real world.

- **Keywords:**
  referring expressions, machine learning, simulations, evaluation, corpora, model theory.

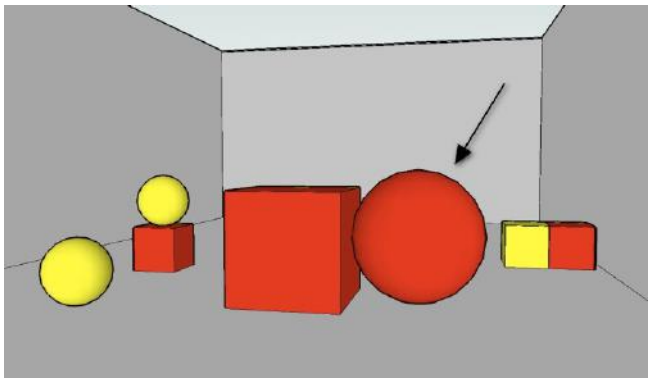
Contenidos

Capítulo 1

Introducción

La **generación de lenguaje natural (GLN)** es el proceso automático (o semi automático) de construcción de un texto en lenguaje natural para la comunicación con fines específicos. Este proceso que convierte información a texto en lenguaje natural es útil para aplicaciones prácticas en las que, por ejemplo, es necesario hacer accesible grandes volúmenes de información posiblemente técnica. La GLN se ha usado para generar recomendaciones de restaurantes personalizadas, para resumir información médica, para generar pronósticos del clima, entre otros [Reiter and Dale, 2000]. La generación de lenguaje natural, está dentro del área de procesamiento del lenguaje natural, que es una rama principal de la inteligencia artificial.

En el marco de esta tesis, una **expresión referencial (ER)** es un sintagma nominal que identifica unívocamente a un objeto en un contexto y para un interlocutor particular. Si quisiéramos referirnos al objeto señalado por la flecha en la Figura 1.1a, podríamos hacerlo con alguna de las expresiones referenciales que se muestran en la Figura 1.1b.



(a)

La esfera grande
La esfera roja que está al lado del cubo rojo
El objeto que está al lado del cubo grande
La bola roja
La pelota a la izquierda del cubo amarillo
La bola grande
La esfera que está a la derecha del cubo rojo y a la izquierda del cubo amarillo
La cosa que está a la derecha del cubo del medio
...

(b)

Figura 1.1: En (b) se listan expresiones referenciales que identifican unívocamente al objeto señalado por la flecha en la escena ilustrada en (a).

La **generación de expresiones referenciales (GER)** entre todas las subtareas de GLN, es una de las que ha recibido más atención. En la práctica, la mayoría de los sistemas de GLN, con independencia de su finalidad, contiene un módulo de GER de algún tipo [Mellish and Evans, 2004]. Esto no es sorprendente en vista del papel central que las expresiones referenciales tienen en la comunicación. Un sistema que proporciona consejos sobre los viajes aéreos [White et al., 2010] tiene que hacer referencia a los vuelos —*el vuelo más barato*, *un vuelo directo*—, un sistema de navegación para automóviles [Dräger and Koller, 2012] necesita generar descripciones espaciales —*tomar el puente junto a la iglesia, a la derecha*—, y un robot que ensambla piezas de juguetes junto con un usuario humano [Foster et al., 2009] debe hacer referencia a los componentes —*inserte el perno verde hasta el final en el cubo rojo*. Cuando

hablamos, nos referimos a cosas (tangibles como un puente, o intangibles como una fecha), es decir generamos expresiones referenciales. Un sistema que genera texto, también deberá generar expresiones referenciales. La generación automática de expresiones referenciales es el tema de esta tesis.

Un sistema de GLN incluye 3 etapas: **determinación de contenido** —qué decir— **lexicalización** —con qué palabras— y **realización lingüística** —cómo decirlo. La determinación de contenido, elige qué información incluir en la oración. La lexicalización elige qué lexemas usar para comunicar el contenido determinado por la etapa anterior. Y la realización arma la oración agregando los artículos, preposiciones, y demás palabras funcionales necesarias y ordenándolas de forma tal que la frase resultante sea gramaticalmente aceptable.

Un sistema de GER, tiene esas 3 etapas también: **determinación de contenido** —decide qué propiedades o relaciones del objeto a describir se incluirán en la expresión referencial— **lexicalización** —elige las palabras que se van a usar para nombrar las propiedades y relaciones— y **realización lingüística** —se encarga de armar el sintagma nominal para que sea gramaticalmente correcto.

Por ejemplo, la primer ER de la Figura 1.1 incluye las propiedades *tamaño* y *forma* del objeto señalado por la flecha, lexicalizadas como *grande* y *esfera* respectivamente y realizadas agregando el artículo *la* antes de *esfera*, e incluyendo el sustantivo *esfera* antes que el adjetivo *grande*; formando así el sintagma nominal *la esfera grande* que es correcto en español. Otra lexicalización y realización de las mismas propiedades —es decir, de la misma semántica— podría ser *la bola de gran tamaño*. A lo largo de esta tesis usaremos el término **expresión referencial (ER)** para nombrar la salida de cualquiera de las 3 etapas de la GER. Es decir, llamaremos expresión referencial al conjunto de propiedades y relaciones que refieren unívocamente a un objeto aunque no estén lexicalizadas o realizadas.

En esta tesis nos enfocamos en la parte de determinación de contenido, daremos como salida fórmulas de lógica que van a determinar el contenido de la ER. En algunos casos se muestran ejemplos que están lexicalizados y realizados, esto se hace a fin de hacer legible e intuitivo el seguimiento de la explicación.

En lo que sigue introduciremos terminología básica, relacionada con la GER que nos servirá a lo largo de toda la tesis.

El **dominio** de una ER define los tipos de entidades que están siendo contemplados. Por ejemplo el dominio de la Figura 1.1 son figuras geométricas en 3 dimensiones. En particular, el dominio incluye cubos y esferas de colores, algunas grandes y otras pequeñas, situadas en un entorno con perspectiva.

El **contexto** de una ER contiene un subconjunto de las entidades del dominio. La Figura 1.1 muestra un contexto con 7 entidades del dominio, 4 rojas y 3 amarillas. Otros ejemplos de contextos son los puntos de referencia visibles en un cierto momento en un camino para el que estamos dando direcciones, un subconjunto de las fotografías utilizadas en una configuración experimental, o los ingredientes de cocina que ya se han mencionado en una receta. En un dominio visual, el contexto, incluyendo las propiedades de sus objetos y su configuración espacial, se puede llamar **escena**.

Cada **entidad** del contexto (también conocido como objeto o elemento) tiene un tipo —*esfera*— ciertas propiedades o características —*color*—, valores de esas propiedades —*rojo*—, y puede tener relaciones con otros objetos —*a la derecha de*. Una **propiedad** (unaria) es una característica de una entidad particular. Por ejemplo, la raza de un perro, el tener o no tener bigotes para un hombre, o el color para un objeto. Cada entidad puede tener muchas propiedades, y puede tener **relaciones** (también llamadas propiedades binarias), por ejemplo con respecto a la posición física, como estar situado al lado de otro objeto.

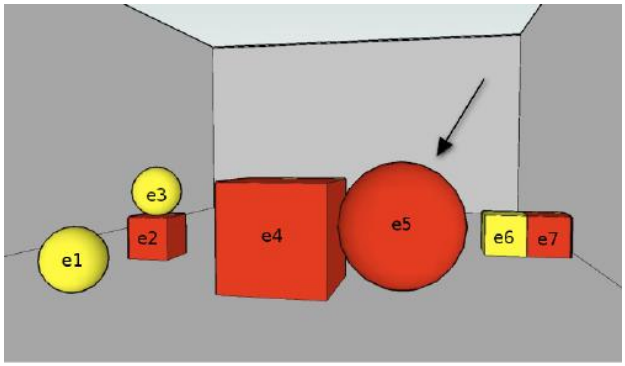
El **target** (u objetivo), es el subconjunto de objetos de un contexto a los cuales queremos referirnos. En la escena del ejemplo de la Figura 1.1, el target es el objeto señalado por la

flecha. En este caso, el target es un conjunto singleton, es decir tiene un sólo elemento. Si el target tiene más de un elemento, las ERs que lo identifican son plurales.

Dado un contexto, un target y una descripción parcial del target (es decir, una descripción que no lo identifica unívocamente), los **distractores** son otros elementos que se encuentran en el contexto, y que también cumplen con la descripción parcial. Por ejemplo, si la descripción parcial es *esfera* las esferas que no son el target de la Figura 1.1 son distractores, y por ello es necesario seguir agregando propiedades o relaciones para identificar unívocamente al target.

Un **algoritmo** para GER, es un procedimiento automático que toma, al menos, algún tipo de representación de un contexto y un target, y da como resultado una (o más) expresión(es) referencial(es) para el target considerado, si puede identificarlo unívocamente en el contexto.

Por ejemplo, una computadora que se enfrenta a la tarea de generar automáticamente expresiones referenciales para la Figura 1.1 necesitará una representación de todos los objetos de la figura y las propiedades de cada uno de ellos. En la Figura 1.2 se muestra una posible forma de representar los objetos, sus propiedades y relaciones: una base de datos que contiene todas las propiedades relevantes de los objetos de la escena. Entonces, la tarea de GER para el objeto e_5 involucra encontrar alguna combinación de valores de propiedades y relaciones con otros objetos, que aplique únicamente a e_5, y no a los otros objetos. Como dijimos, esta tarea de encontrar las propiedades y relaciones que aplican a un target y no a los distractores, se llama selección de contenido para la generación de una expresión referencial. Mirando la Tabla 1.2b podemos decir que *red ball*, *large ball* y *large red ball* son algunas ERs del objeto e_5, cuyas realizaciones en español podrían ser: *la bola roja*, *la esfera grande* y *la esfera grande y roja*.



(a)

id	type	color	size	right of	left of	ontop	below
e_1	ball	yellow	small	-	-	-	-
e_2	cube	red	small	-	-	-	e_3
e_3	ball	yellow	small	-	-	e_2	-
e_4	cube	red	large	-	e_5	-	-
e_5	ball	red	large	e_4	-	-	-
e_6	cube	yellow	small	-	e_7	-	-
e_7	cube	red	small	e_6	-	-	-

(b)

Figura 1.2: Formalización de las propiedades de la escena en una tabla de doble entrada.

En la próxima sección veremos porqué la tarea de GER es más compleja de lo que parece a primera vista y en la siguiente introduciremos cómo herramientas de teoría de modelos nos pueden ayudar.

1.1 Expresiones referenciales bajo incertidumbre

Cuando la generación de expresiones referenciales ocurre en la vida real, en lugar de ocurrir en un experimento controlado, las fuentes de **incertidumbre** que afectan al proceso se multiplican. En trabajo reciente en el área de GER [Turner et al., 2009; van Deemter, 2016] se discute que, al intentar usar algoritmos de GER en dominios grandes y realistas (como por ejemplo la descripción de regiones en mapas) la identificación óptima del target es una tarea que puede ser aproximada pero raramente lograda. Los autores argumentan que esto se debe, en parte, a que las representaciones geográficas son necesariamente **incompletas**. Esta falta de información introduce incertidumbre. Por ejemplo, el restaurante señalado por la flecha en la Figura 1.3, ¿es realmente el único restaurante de la calle de Cádiz o ¿es el único que aparece en el mapa?.

Figura 1.3: Fragmento de mapa de la ciudad de Madrid obtenido de OpenStreetMap.

Cuando la información del contexto proviene de datos de sensores, las entradas del algoritmo de GER son inevitablemente ruidosas. Es decir, contienen información no sólo incompleta sino también posiblemente **incorrecta**. Incluso en contextos tan simples como el de la Figura 1.1 hay incertidumbre: ¿la Tabla 1.2(b) representa toda la información del contexto?. Algunos podemos opinar que sí, otros que no. En efecto, la persona que generó la ER *la pelota a la izquierda del cubo amarillo* opina que no: la relación *a la izquierda de* entre el target y el cubo amarillo no está representada en la tabla. Un algoritmo de GER con esta tabla como input no puede generar esta expresión referencial. La información disponible no sólo impide la generación de ERs válidas, sino que también permite la generación de ERs claramente incorrectas como *la esfera roja que no tiene nada a la derecha.*

La tabla puede completarse para que sea posible generar la expresión referencial. La Figura 1.4 ilustra una posible forma de completar la información de la tabla representada en (a) como un grafo, en el grafo (b). Sin embargo, este nuevo grafo tampoco es completo, de hecho, *la cosa que está a la derecha del cubo del medio* no se podría generar con este grafo. Toda representación de un contexto será necesariamente incompleta.

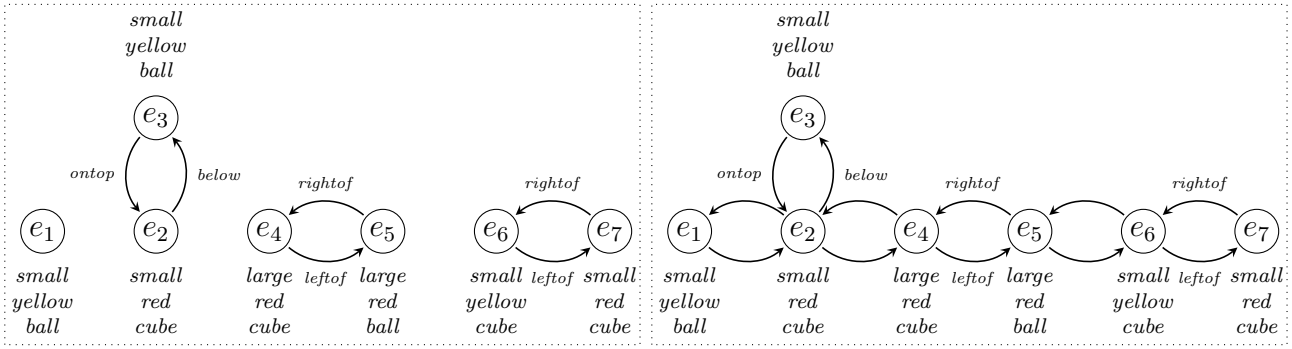


Figura 1.4: Formalización de las propiedades de la Figura 1.2 como un grafo etiquetado en (a) y representación de contexto más completo en (b).

Aunque no hablemos ni de incompletitud ni de incorrectitud también hay incertidumbre en **cuál** de todas las posibles ERs un sistema de GER debe elegir. Por ejemplo seguramente hay personas que preferirán ERs que usen la palabra *esfera*, antes que *bola*. Y quizás no falte quien diga que *la* mejor expresión referencial para el target ni si quiera aparece en la Figura 1.1. Sin embargo, habrá una tendencia como se muestra en estudios previos [Viethen, 2011]: Nadie preferirá una ER que use 21 propiedades y 6 relaciones de la tabla para describir al target de la Figura 1.1.

El largo de una ER afecta su calidad. Se podría pensar, por ejemplo, que las referencias son óptimas cuando son **mínimas** en longitud, es decir, cuando contienen sólo información suficiente para identificar el objeto target y nada más. Pero la generación de referencias mínimas no es lo que las personas hacen ni lo que es más útil para los oyentes, como se muestra en trabajo previo [Luccioni et al., 2015; Paraboni et al., 2007]. Y ni siquiera nos resuelve completamente el problema de elegir una ER: *la bola grande* y *la esfera roja* son dos ERs de la Figura 1.1 y tienen exactamente la misma longitud.

Dado que la incertidumbre es inherente a la tarea de GER, lo mejor que se puede hacer para ayudar a los sistemas interactivos de GLN es que un algoritmo de GER devuelva un **ranking** de ERs, o lo que es lo mismo, un conjunto de ERs en el que cada ER tiene un valor asociado que intenta reflejar cuán buena es esa ER para el input dado. Si un sistema interactivo tiene varias ERs para un objeto, puede dar la siguiente frase simplemente seleccionando otra ER, cuando el usuario le indica que no ha entiende la frase.

¿Cómo podemos generar un ranking de ERs?. Un algoritmo simplista generaría primero todas las ERs posibles y luego las ordenaría con algún criterio. Esto es posible porque la cantidad de ERs es finita dado un modelo finito, pero esta cantidad puede ser muy grande, exponencial en el tamaño del modelo y la gran mayoría de estas ERs no se llegarían a usar en una aplicación real. Una aplicación sólo usará las mejores ERs de un target considerado. Una forma más eficiente de generar un ranking de ERs es diseñar un algoritmo no determinístico que intente generar primero las mejores ERs.

Un algoritmo para GER es **no-determinístico** si puede dar diferentes ERs para el mismo contexto y target en sucesivas ejecuciones. Para producir un ranking de las mejores ERs para un target, el no determinismo debe estar guiado de alguna forma que le permita preferir ERs mejores y generarlas con una mayor probabilidad. Llamaremos a esta guía: **probabilidad de uso** de una propiedad o relación. Esta probabilidad se puede ver afectada por características de la propiedad o relación (algunas propiedades pueden ser más fáciles de interpretar que otras) o por características de quién interpretará la ER (por ejemplo, su conocimiento del idioma).

Hay diferentes **tipos de propiedades**, por ejemplo taxonómicas (las intrínsecas al objeto target, como *esfera*, o *roja*), relacionales (las que necesitan describir a otro objeto, por ejemplo *estar al lado de*), vagas (por ejemplo, *chico*, *grande*, necesitan ser interpretadas con respecto a un contexto, ¿con respecto a qué algo es chico o grande?). Ciertos valores de propiedades pueden ser más fáciles de identificar que otros, por ejemplo cierto color verde podría ser más complicado de identificar que el tamaño grande de un objeto. Notar que cuando decimos el tamaño, tenemos como marco de referencia a los objetos de la escena, en ese contexto un objeto es *grande*.

La selección de la ER más apropiada también debe tener en cuenta al **interlocutor**, ya que es natural que los humanos demos distintas ERs a distintos interlocutores. La selección de qué propiedades y/o relaciones con otros objetos incluir en una expresión referencial dependerá del propósito que tengamos para dicha expresión referencial. Una expresión referencial será muy distinta si nuestro objetivo es dar la mínima información que distinga al objeto, que si nuestro objetivo es ayudar al interlocutor a que identifique el objeto.

La generación de expresiones referenciales en el mundo real sufre de diversas fuentes de incertidumbre. En esta tesis proponemos formas para modelar esta incertidumbre partiendo de

técnicas de representación de conocimiento basadas en teoría de modelos y proponiendo como extenderlas usando distribuciones finitas de probabilidades.

1.2 Expresiones referenciales usando teoría de modelos

Los **modelos relacionales**, también conocidos como **modelos de Kripke** y **modelos de primer orden**, son muy utilizados para la representación de situaciones o escenas. Los modelos relacionales son grafos etiquetados y pueden verse también como autómatas finitos. Estas estructuras matemáticas han sido muy estudiadas y tienen muchas propiedades bien conocidas [Areces and ten Cate, 2006].

En esta tesis se adaptan algoritmos clásicos de minimización de autómatas aplicados a la caracterización de modelos relacionales. Particularmente extendimos técnicas y algoritmos de teoría de modelos y simulaciones integrando una distribución finita de probabilidades que representa esta incertidumbre. Nuestro objetivo es generar un ranking de las expresiones referenciales ordenado por la probabilidad de ser correctamente interpretada en el contexto.

Para un **vocabulario** de símbolos relacionales r, un modelo relacional $\mathcal{M}$ es una tupla $\langle \Delta, \| \cdot \| \rangle$ donde:

- Δ es un conjunto no vacío de objetos llamado el dominio
- $\| \cdot \|$ es una función de interpretación, esto es, $\|r\| \subseteq \Delta^n$ para todo símbolo de relación n-ario r que está en el vocabulario.

El **tamaño** de un modelo $\mathcal{M}$ es la suma $(\#\Delta + \#\| \cdot \|)$, donde $\#\Delta$ es la cardinalidad de Δ y $\#\| \cdot \|$ es la suma de todas las aridades de las relaciones en $\| \cdot \|$. Si asumimos un vocabulario finito de símbolos de relación n-arios, entonces $\mathcal{M}$ es *finito*.

A continuación vamos a presentar 2 ejemplos de modelos relacionales, uno en el que tenemos entidades **agentivas** (es decir, agentes animados que pueden realizar acciones) como perros y gatos, y otro en el que las entidades son **no agentivas**, es decir, son objetos estáticos.

El primer ejemplo se muestra en la Figura 1.5 donde damos una posible representación de la escena de la Figura 1.1 como un modelo relacional. El tamaño de este modelo es 21, ya que $\#\Delta$=7 y $+\#\| \cdot \|$=14.

$$
\begin{aligned}
\Delta &= \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\} \\
\|red\| &= \{e_2, e_4, e_5, e_7\} \\
\|yellow\| &= \{e_1, e_3, e_6\} \\
\|ball\| &= \{e_1, e_3, e_5\} \\
\|cube\| &= \{e_2, e_4, e_6, e_7\} \\
\|small\| &= \{e_1, e_2, e_3, e_6, e_7\} \\
\|large\| &= \{e_4, e_5\} \\
\|leftof\| &= \{(e_4, e_5), (e_6, e_7)\} \\
\|rightof\| &= \{(e_5, e_4), (e_7, e_6)\} \\
\|ontop\| &= \{(e_3, e_2)\} \\
\|below\| &= \{(e_2, e_3)\}
\end{aligned}
$$

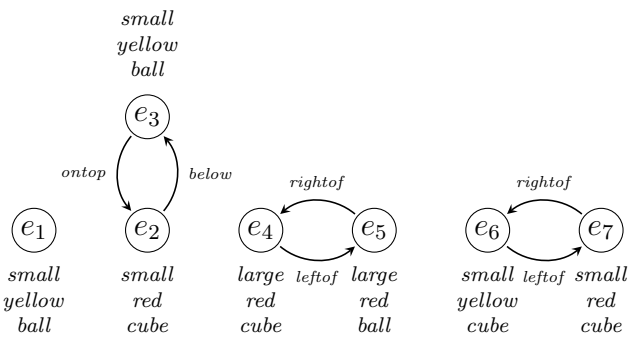


Figura 1.5: Representación de la escena de la Figura 1.1 como un modelo relacional e interpretación de sus propiedades y relaciones. Ejemplo con entidades no agentivas.

El dominio es $\Delta = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$. La interpretación de *ball* es e_1, e_3 y e_5, ya que estos objetos son esferas, la de *cube* es $\|cube\| = \{e_2, e_4, e_6, e_7\}$, ya que esos objetos son cubos. Tenemos las relaciones *rightof*, *leftof*, *below* y *ontop*, cuyas interpretaciones se muestran en

la figura. Como discutimos en la sección anterior, se puede argumentar que este modelo está incompleto, pero es suficiente para los objetivos ilustrativos de esta sección.

El segundo ejemplo se muestra en la Figura 1.6 en la cual hemos representado un contexto agentivo como un modelo relacional. En el modelo, a, b y d son perros, mientras que c y e son gatos; d es un pequeño beagle; b y c también son pequeños. Leeremos $\mathit{sniffs}(d, e)$ como "*d huele a e*". La interpretación del símbolo relacional *dog* es $\|\mathit{dog}\| = \{a, b, d\}$ ya que a, b y d son los únicos perros del contexto. La interpretación de *sniffs* es el conjunto de pares de elementos que cumplen *sniffs*. Por ejemplo (a, a) pertenece al conjunto dado que a es un perro que se huele a sí mismo en el modelo. El tamaño de este modelo es 11, ya que $\#\Delta{=}5$ y $+\#\|\cdot\|{=}6$.

$$
\begin{aligned}
\Delta &= \{a, b, c, d, e\} \\
\|\mathit{dog}\| &= \{a, b, d\} \\
\|\mathit{cat}\| &= \{c, e\} \\
\|\mathit{beagle}\| &= \{d\} \\
\|\mathit{small}\| &= \{b, c, d\} \\
\|\mathit{sniffs}\| &= \{(a, a), (b, a), (c, b), (d, e), (e, d)\}
\end{aligned}
$$

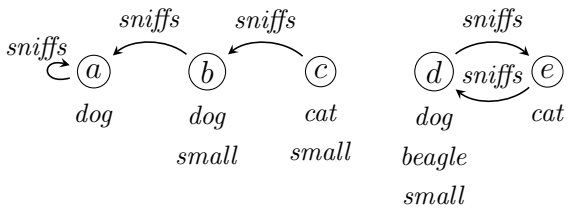


Figura 1.6: Representación de un contexto agentivo como un modelo relacional e interpretación de sus propiedades y relaciones.

Ahora nos enfocaremos en conseguir ERs para identificar a targets dados usando los modelos que acabamos de describir. Veremos lenguajes de distintas lógicas y cómo las fórmulas lógicas pueden representar a las ERs. A continuación definimos el lenguaje clásico de la lógica de primer orden (con desigualdad), $\mathcal{FO}$. Dicho lenguaje se define inductivamente como sigue. Una fórmula φ es una fórmula de $\mathcal{FO}$ si es de alguna de las siguientes formas:

1. $\top$ es como decir *cosa* o *ese*, todo elemento del modelo satisface $\top$.
2. $x_i \not\approx x_j$ La desigualdad entre dos elementos x_i y x_j.
3. $r(\bar{x})$ Relaciones de tuplas.
4. $\neg\gamma$ Negación de fórmulas.
5. $\gamma \wedge \gamma'$ Conjunción de fórmulas.
6. $\exists x_i.\gamma$ Existencial de una variable ligada en la fórmula γ.

donde $\gamma, \gamma' \in \mathcal{FO}$, r es un símbolo de relación n-aria y $\bar{x}$ es una tupla de n variables. Como es usual, $\gamma \vee \gamma'$ y $\forall x.\gamma$ son las versiones cortas de $\neg(\neg\gamma \wedge \neg\gamma')$ y $\neg\exists x.\neg\gamma$, respectivamente. Fórmulas de la forma $\top$, $x_i \not\approx x_j$ y $r(\bar{x})$ son llamadas *átomos*.

Dado un modelo relacional $\mathcal{M} = \langle \Delta, \|\cdot\| \rangle$ y una fórmula γ con variables libres[1] $x_1 \ldots x_n$, inductivamente definimos la **extensión** o **interpretación** de γ como el conjunto de n-tuplas $\|\gamma\|^n \subseteq \Delta^n$ que satisface:

1. $\|\top\|^n = \Delta^n$
2. $\|x_i \not\approx x_j\|^n = \{\bar{a} \mid \bar{a} \in \Delta^n, a_i \neq a_j\}$
3. $\|\neg\delta\|^n = \Delta^n \setminus \|\delta\|^n$
4. $\|r(x_{i_1} \ldots x_{i_k})\|^n = \{\bar{a} \mid \bar{a} \in \Delta^n, (a_{i_1} \ldots a_{i_k}) \in \|r\|\}$

[1]Asumimos que cada variable no puede aparecer libre y ligada a la vez, que una variable no está ligada 2 veces, y que el índice de las variables crece en la fórmula de izquierda a derecha.

5. $\|\delta \wedge \theta\|^n = \|\delta\|^n \cap \|\theta\|^n$

6. $\|\exists x_l.\delta\|^n = \{\bar{a} \mid \bar{a}e \in \|\delta'\|^{n+1}$ para algún elemento $e\}$

donde $1 \leq i, j, i_1, \ldots, i_k \leq n$, $\bar{a} = (a_1 \ldots a_n)$, $\bar{a}e = (a_1 \ldots a_n, e)$ y δ' son obtenidos reemplazando todas las ocurrencias de x_l en δ por x_{n+1}.

Con una sintaxis y semántica de un lenguaje en mente, podemos formalmente definir el problema de **GER en $\mathcal{L}$** ($\mathcal{L}$-GER) para un conjunto target T de elementos. $\mathcal{L}$ es el lenguaje de la lógica elegida, en este caso hemos explicado la lógica de primer orden $\mathcal{FO}$, y en el Capítulo 3 restringiremos esa lógica para obtener ERs más cercanas a las que aparecen en corpora y para conseguir algoritmos computacionalmente más eficientes.

Problema $\mathcal{L}$-GER	
Entrada:	un modelo $\mathcal{M} = \langle \Delta, \|\cdot\| \rangle$ y un conjunto target no vacío $T \subseteq \Delta$.
Salida:	una fórmula $\varphi \in \mathcal{L}$ tal que $\|\varphi\| = T$ si existe, y $\perp$ caso contrario.

La salida del problema $\mathcal{L}$-GER es una fórmula de $\mathcal{L}$ cuya interpretación en el modelo de input $\mathcal{M}$ es el conjunto target T, si esa fórmula existe. Cuando la salida no es $\perp$, decimos que φ es una **expresión referencial en $\mathcal{L}$ (ER-$\mathcal{L}$)** para T en $\mathcal{M}$, y cuando es $\perp$ puede ser que la lógica elegida no sea lo suficientemente expresiva para identificar al target, o que el modelo dado no tenga suficiente detalle.

Consideramos sólo modelos relacionales con símbolos de relaciones unarias y binarias, usaremos p para las proposiciones (propiedades) y r para los símbolos de relación binarias. Como dijimos anteriormente, dado un modelo $\mathcal{M}$, podría haber un número muy grande de fórmulas que de forma unívoca describan al target (incluso fórmulas que no son lógicamente equivalentes podrían tener la misma interpretación una vez que el modelo este fijo). Por ejemplo, en el modelo $\mathcal{M}$ de la Figura 1.5 las fórmulas $\mathit{large} \wedge \mathit{ball}$ y $\mathit{red} \wedge \mathit{ball}$ no son lógicamente equivalentes pero tienen la misma interpretación en $\mathcal{M}$ ya que $\|\mathit{large} \wedge \mathit{ball}\| = \{e_5\}$ y $\|\mathit{red} \wedge \mathit{ball}\| = \{e_5\}$. Diferentes ERs del mismo target podrían ser más o menos apropiadas en el contexto dado. Otra cosa que es importante tener en cuenta, es que la determinación de contenido usando lenguajes con diferente poder expresivo, puede tener un impacto en la complejidad computacional de la GER y en la etapa posterior de realización sintáctica. Para ilustrar eso, veamos las fórmulas que identifican al elemento target b en la Figura 1.6.

$$\varphi_1 : \mathit{dog}(x) \wedge \mathit{small}(x) \wedge \exists y.(\mathit{sniffs}(x, y) \wedge \mathit{dog}(y))$$
$$\varphi_2 : \mathit{dog}(x) \wedge \mathit{small}(x) \wedge \forall y.(\neg \mathit{cat}(y) \vee \neg \mathit{sniffs}(x, y))$$
$$\varphi_3 : \mathit{dog}(x) \wedge \exists y.(x \not\approx y \wedge \mathit{dog}(y) \wedge \mathit{sniffs}(x, y))$$
$$\varphi_4 : \mathit{dog}(x) \wedge \exists y.(\mathit{cat}(y) \wedge \mathit{small}(y) \wedge \mathit{sniffs}(y, x))$$

Notar que las fórmulas mostradas, hacen uso de diferentes operadores de la lógica $\mathcal{FO}$: hay negación de átomos y de relaciones, hay cuantificación existencial, y universal, conjunción, disyunción y desigualdad. La realización sintáctica de algunas de éstas fórmulas pueden involucrar estructuras gramaticales complejas. Por ejemplo la fórmula φ_4 requiere el uso de voz pasiva y podría realizarse como *el perro que es olido por un gato pequeño*. La fórmula $\mathit{dog}(x) \wedge \exists y.(x \not\approx y \wedge \mathit{dog}(y) \wedge \mathit{sniffs}(x, y))$ que representa al target b de la Figura 1.6 requiere el uso de lexemas como otro y puede realizarse como *el perro que huele a otro perro*. La fórmula φ_2 requiere el uso de negación del predicado y podría realizarse como *el perro pequeño que no huele gatos*.

Como nuestra meta es hacer un ranking de las mejores expresiones referenciales, una manera de acercarnos a esa meta es restringiendo la lógica. Si restringimos a $\mathcal{FO}^-$ que no tiene negación, aseguramos que fórmulas como φ_2 no se generarán. Si sacáramos el operador distinto

($\not\approx$) del lenguaje, φ_3 también queda excluida. A continuación presentamos fragmentos de $\mathcal{FO}$ conocidos como lógicas de descripción. Por ejemplo, el lenguaje de la lógica de descripción $\mathcal{ALC}$ de [Areces and ten Cate, 2006], se define sintácticamente como el conjunto de fórmulas que se pueden generar recursivamente como sigue

$$\top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \exists r.\varphi$$

donde p es un símbolo proposicional, r es un símbolo relacional binario, y φ, φ' son fórmulas de $\mathcal{ALC}$.

Mediante la restricción a fórmulas de $\mathcal{ALC}$ evitaríamos fórmulas como φ_3 (es decir con desigualdad). También evitaríamos fórmulas como φ_4 (*el perro que es olido por un pequeño gato*) en la cual tenemos que realizar la parte del existencial como una voz pasiva al aparecer la variable x en segundo lugar.

Cada lenguaje lógico puede ser visto como un compromiso entre la expresividad, realizabilidad y complejidad computacional. Para la tarea de GER el uso de un lenguaje u otro debe depender del contexto particular considerado y del target a describir.

Para el segundo ejemplo, supongamos que queremos identificar al objeto e_5 de la Figura 1.5. Algunas fórmulas posibles que identifican unívocamente a e_5 en el modelo se muestran a continuación.

φ_1 : $red(x) \wedge ball(x)$

φ_2 : $large(x) \wedge ball(x)$

φ_3 : $large(x) \wedge red(x) \wedge ball(x)$

φ_4 : $large(x) \wedge red(x) \wedge ball(x) \wedge \exists y.(rightof(x,y) \wedge large(y) \wedge red(y) \wedge cube(y))$

φ_5 : $large(x) \wedge red(x) \wedge ball(x) \wedge \forall y.(\neg ball(y) \vee \neg rightof(x,y))$

φ_6 : $large(x) \wedge red(x) \wedge ball(x) \wedge \exists y.(x \not\approx y \wedge cube(y) \wedge rightof(x,y))$

φ_7 : $large(x) \wedge red(x) \wedge ball(x) \wedge \exists y.(cube(y) \wedge red(y) \wedge leftof(y,x))$

Las fórmulas que no contienen negación se pueden representar como subgrafos como se muestran en la Figura 1.7. El de la izquierda representa la fórmula φ_1 y el de la derecha a la fórmula φ_4.

Figura 1.7: Subgrafos que representan las fórmulas φ_1 y φ_4.

Las fórmulas φ_1 a φ_7 pueden realizarse como sigue.

φ_1: *La esfera roja*

φ_2: *La esfera grande*

φ_3: *La esfera grande y roja*

φ_4: *La esfera grande y roja que está a la derecha de un cubo grande y rojo*

φ_5: *La esfera grande y roja, que no tiene ninguna esfera a la derecha*

φ_6: *La esfera grande y roja que está a la derecha de un cubo*

φ_7: *La esfera grande y roja que tiene un cubo rojo a la izquierda*

Notar que φ_1 y φ_2 son mínimas, es decir no se puede dar una fórmula más corta que esas que identifique al target. Las fórmulas φ_1, φ_2, φ_3, φ_4 y φ_6 son caracterizadas como fórmulas positivas, conjuntivas y existenciales (no contienen negación y sólo tienen conjunciones, cuantificadores existenciales y desigualdad), este tipo de fórmulas son las que más se encuentran

en corpora de ERs [Viethen and Dale, 2006; van Deemter et al., 2006; Dale and Viethen, 2009]. Luego como dijimos recién, podemos elegir otra lógica cuyo lenguaje esté restringido a las ERs que queremos generar. Restringiendo la determinación de contenido a $\mathcal{FO}^-$, aseguramos que fórmulas como φ_5 no se generarán. Si prohibimos el distinto del lenguaje, φ_6 también queda excluida. La lógica de menor complejidad computacional que se corresponde con todas esas restricciones es $\mathcal{EL}$, la cual está dada por:

$$\top \mid p \mid \gamma \wedge \gamma' \mid \exists r.\gamma$$

donde p es un símbolo proposicional, r un símbolo de relación binaria, y $\gamma, \gamma' \in \mathcal{EL}$, es decir son fórmulas de $\mathcal{EL}$. $\mathcal{EL}$ se corresponde con el fragmento de $\mathcal{ALC}$ sin negación.

En el Capítulo 3 describimos cómo los operadores lógicos afectan la expresividad y la complejidad computacional de la tarea de GER, así como las ERs generadas. En el Capítulo 4 proponemos como generar no una sino un ranking de ERs y agregamos una distribución finita de probabilidades asociada a los símbolos de relación del modelo.

1.3 Mapa de la tesis

En esta sección contamos como está organizada la tesis. En lo que sigue daremos el resumen de la temática de cada capítulo.

1.3.1 Capítulo 1: "Introducción"

Situamos a la generación automática de expresiones referenciales dentro de la inteligencia artificial y de la generación de lenguaje natural. Dijimos que un sistema de generación de lenguaje natural tiene tres etapas: la determinación de contenido, la lexicalización y la realización sintáctica. La generación de expresiones referenciales es una subárea de la generación de lenguaje natural y por lo tanto, incluye las mismas tres etapas. La determinación de contenido, que es la selección de las propiedades y/o relaciones con otros objetos que vamos a elegir para identificar unívocamente al target, es el tema de la tesis. Explicamos conceptos básicos que usamos a lo largo de la tesis, como *dominio*, *contexto*, *propiedad*, *target*, *distractor* y *algoritmo*. Explicamos porqué la tarea de GER es compleja, ya que hay incertidumbre al representar el contexto en situaciones de la vida real. Los modelos son indefectiblemente incompletos o incorrectos. Dimos una introducción a la variedad de expresiones referenciales que se puede tener para el mismo objeto target en el mismo contexto. Propusimos dar no una ER sino un ranking de ERs para un input dado. Introdujimos el concepto de teoría de modelos, describimos diversos lenguajes formales (es decir, lógicos) y su impacto al ser usados en el problema de generación automática de expresiones referenciales. Partimos desde la lógica de primer orden ($\mathcal{FO}$), la más expresiva, explicamos el lenguaje asociado, para darnos una idea de cuáles son las fórmulas que el lenguaje puede generar, vimos una serie de ejemplos y propusimos restricciones al lenguaje que permiten generar estructuras que se ven en corpora.

1.3.2 Capítulo 2: "Generación de expresiones referenciales"

Este capítulo está dividido en 4 secciones. En la primer sección definimos los diferentes tipos de expresiones referenciales: proposicionales, relacionales, minimales, sobreespecificadas, subespecificadas. Luego comentamos en qué se basan la teoría introducida en [Clark, 1992; Clark, 1996; Clark and Marshall, 1981] y cómo son refutadas por algunas investigaciones. Mostramos información básica de corpora existente en el área, varios de los cuales usaremos a lo largo de la tesis. Viendo la simplicidad de estos corpora, por un lado podemos ver la complejidad de

la tarea, ya que si para corpora tan simples, es tan difícil conseguir algoritmos que hagan lo que hacen los humanos, podríamos imaginarnos que es realmente una tarea mucho más difícil en contextos que la gente usa en la vida diaria. Por otro lado, motivamos la obtención de un nuevo corpus, el cual describimos en el Capítulo 6, que usa imágenes de mapas de ciudades como contextos. En la segunda sección describimos diferentes tipos de algoritmos para la tarea de generación automática de expresiones referenciales: determinísticos, no-determinísticos, que generan sobreespecificación, plurales, sólo singulares, relacionales o proposicionales. Se estudió el avance en el área de la generación automática de expresiones referenciales, los algoritmos existentes, se compararon esos algoritmos en cuanto al tipo de algoritmo y salida que producen. Se explica el algoritmo Graph en más detalle que los demás porque es el más cercano a los tipos de algoritmos que se extienden en esta tesis. Luego en la tercer sección damos una introducción a las métricas de evaluación de la tarea de GER. Algunas métricas usan corpus para comparar las salidas de los algoritmos con las ERs dadas por humanos. Hay métricas automáticas y otras manuales. Para finalizar el capítulo damos un resumen y explicamos cómo se linkea con los demás capítulos.

1.3.3 Capítulo 3: "Usando teoría de modelos"

Damos definiciones básicas de modelo, interpretación y fórmula. Explicamos de la noción de similaridad de 2 elementos u y v del modelo la cual dice que son similares en $\mathcal{L}$, cuando para toda fórmula $\varphi \in \mathcal{L}$, tenemos que $\{u, v\} \subseteq \|\varphi\|$. Se dice que u y v son indistinguibles en el lenguaje lógico $\mathcal{L}$, ya que no hay una fórmula que satisfaga uno y no el otro. El concepto de *similaridad* puede ser usado para definir la tarea GER pero implica generar posiblemente infinitas fórmulas. Veremos que para modelos finitos la $\mathcal{L}$-similaridad puede definirse en términos de $\mathcal{L}$-simulaciones lo cual nos permitirá saber si hay una ER para el target sin checkear infinitas fórmulas. Se adaptan algoritmos conocidos de simulación para varias lógicas tales como $\mathcal{ALC}$ y $\mathcal{EL}$ entre otras para la GER. Explicamos cómo computamos las ERs para los distintos lenguajes lógicos $\mathcal{FO}^-$, $\mathcal{ALC}$ y $\mathcal{EL}$. Clasificamos los algoritmos vistos según la complejidad computacional de los mismos.

1.3.4 Capítulo 4: "Modelando la incertidumbre"

En este capítulo explicamos nuestra propuesta de agregar probabilidades de uso a las palabras de la signatura del modelo para representar la incertidumbre del contexto, adaptando algoritmos introducidos en el Capítulo 3. Mostramos que los algoritmos permiten sobreespecificación, pero aseguran terminación y agregamos un componente aleatorio para conseguir no-determinismo y así generar un ranking de ERs. Mostramos en detalle el input que toma el nuevo algoritmo. Explicamos cómo obtener las probabilidades de uso que toma como input el algoritmo a partir de corpus usando aprendizaje automático. Para la tarea de aprendizaje automático usamos características independientes del dominio. Mostramos la salida del algoritmo, explicamos cómo conseguimos no-determinismo en las distintas ejecuciones del algoritmo y cómo aseguramos terminación. Además mostramos completitud, es decir que siempre conseguimos una expresión referencial si existe. Explicamos cómo agregamos sobreespecificación a las expresiones referenciales. Finalmente mostramos un ejemplo de ejecución.

1.3.5 Capítulo 5: "Evaluación de rankings sobre benchmarks"

En el Capítulo 2 se describieron métricas de evaluación tanto automáticas como manuales. Evaluamos los algoritmos presentados en el Capítulo 4, teniendo en cuenta ambos tipos de métricas. La evaluación está dividida en 2 partes. Una parte en la cual comparamos la salida del algoritmo

para modelos de 2 corpora descriptos en el Capítulo 2, con expresiones referenciales dadas por los humanos, tomando como input probabilidades de uso obtenidas con aprendizaje automático a partir de cada corpora. Vemos cómo las ERs generadas con probabilidades obtenidas del corpus con aprendizaje automático se acercan mucho más a las humanas, que 2 baselines (random y uniforme). En la otra parte de este capítulo presentamos una evaluación manual en la cual 2 jueces humanos decidieron qué ER era mejor entre la del corpus y la generada por el algoritmo. Es interesante notar cómo muchas veces las ERs generadas por el algoritmo fueron juzgadas por los jueces como mejores que las humanas. El algoritmo tiene la ventaja de que siempre da una ER si existe, en cambio los humanos muchas veces dan expresiones ambiguas que no son referenciales.

1.3.6 Capítulo 6: "Recolección y análisis del corpus ZOOM"

Introducimos un nuevo corpus, el ZOOM corpus, el cual fue recolectado en un trabajo conjunto con la Universidad de São Paulo (Brasil), para tener un corpus de un dominio más natural de expresiones referenciales. El corpus ZOOM contiene expresiones referenciales de puntos de interés en mapas. Los mapas son fragmentos de las ciudades de Madrid y Lisboa. Este corpus fue recolectado en 2 idiomas, español y portugués. Contiene todos los tipos de ERs presentados en el Capítulo 2. Se explica el método de recolección del corpus, se dan estadísticas de las personas que completaron el experimento y se explica la manera en que se anotó el corpus. Se describe brevemente y se evalúa sobre este corpus un método puramente basado en aprendizaje automático que usa *support vector machines*, para decidir si incluir o no una propiedad en la ER de salida. Además se evalúa el desempeño de nuestros algoritmos sobre un fragmento del corpus ZOOM.

1.3.7 Capítulo 7: "Conclusiones y trabajo futuro"

En este capítulo se da un resumen de lo estudiado, se explican los avances realizados en esta tesis, como la incorporación de distribuciones finitas de probabilidades a algoritmos determinísticos que usan simulaciones para la GER. Esta distribución de probabilidades la aprendemos desde corpora con funciones de regresión lineal. Los algoritmos están inspirados en el modelo psicolingüístico de GER que determina la forma de sobreespecificar. Evaluamos los rankings obtenidos con una serie de benchmarks del área usando métricas automáticas y humanas. Creamos un nuevo corpus de ERs, el ZOOM corpus, que es más complejo y cercano a aplicaciones de la vida real que corpora existente. Damos una serie de propuestas para trabajo futuro en las que se aprovechan las potencialidades de los algoritmos basados en lógicas formales y teoría de modelos para dominios complejos. Por otro lado proponemos extensiones a los algoritmos para tratar casos con target plurales que fueron los menos explorados en esta tesis.

Capítulo 2

Generación de expresiones referenciales

En este capítulo presentamos definiciones básicas y un resumen del estado del arte del área de generación de expresiones referenciales. Nos enfocamos en el trabajo previo más relevante para esta tesis. En la Sección 2.1 damos definiciones básicas que nos servirán a lo largo de la tesis. Primero, explicamos *qué* expresiones referenciales pueden generar las personas. Describimos los diferentes tipos de expresiones referenciales. Las ERs se diferencian según el tipo de propiedades que contengan, pueden contener propiedades atómicas, o relaciones con otros objetos, o ambas. Las ERs también se diferencian por la cantidad de información que contienen, pudiendo ser minimales o sobreespecificadas. Definimos ER plural y parcial. En segundo lugar, describimos trabajo previo que estudia *cómo* los humanos usamos el lenguaje para transmitir intenciones, y cómo esas intenciones son interpretadas por los oyentes, describimos trabajos que estudian el rol de la sobreespecificación y damos ejemplos de uso. Finalmente, introducimos corpora de ERs existente. Luego, en la Sección 2.2 presentamos una clasificación de los algoritmos del área según los tipos de ERs que pueden generar. Pueden ser determinísticos o no-determinísticos. Generar o no sobreespecificación, plurales o sólo singulares, relacionales o proposicionales. Además describimos los algoritmos más conocidos del área. En la Sección 2.3, damos una introducción a las métricas de evaluación en el área. Algunas de ellas usan corpus para comparar automáticamente salidas de algoritmos con ERs dadas por personas. Otras necesitan jueces humanos para evaluar la calidad de las ERs generadas por el algoritmo. Para finalizar, en la Sección 2.4, damos el resumen del capítulo y explicamos su relación con los demás capítulos.

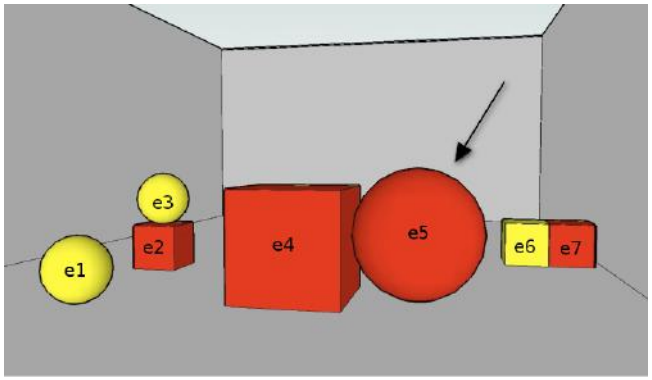
2.1 Generación *humana* de expresiones referenciales

Esta sección está dividida en 3 partes, en la primera describimos diferentes tipos de ERs. En la segunda describimos trabajo previo que estudia cómo las personas generan ERs. Veremos que la sobreespecificación es algo común en la mayoría de los contextos, que no está mal vista por parte de los oyentes y que, de hecho, puede ser útil en diversos dominios. En la tercera presentamos corpora existente en el área y mostramos una comparación entre ellos.

2.1.1 ¿*Qué* tipos de expresiones referenciales existen?

Recordemos que una **expresión referencial** (ER), es un sintagma nominal que identifica a un target unívocamente en un contexto dado para un interlocutor particular. En una ER, una **propiedad** es una característica propia de un objeto. Por ejemplo, en la Figura 2.1, el objeto

señalado con la flecha tiene la propiedad *forma* con valor *esfera*.[1] Una **relación** caracteriza a un objeto describiendo propiedades de otro u otros objetos. Por ejemplo, en la ER *La pelota que está al lado del cubo grande* la relación *al lado de* necesita describir propiedades de otro objeto, en este caso del *cubo grande*.



(a)

La esfera grande
La esfera roja que está al lado del cubo rojo
El objeto que está al lado del cubo grande
La bola roja
La pelota a la izquierda del cubo amarillo
La bola grande
La esfera que está a la derecha del cubo rojo y a la izquierda del cubo amarillo
La cosa que está a la derecha del cubo del medio
...

(b)

Figura 2.1: Expresiones referenciales para el contexto de la figura extraídas del GRE3D7 corpus [Viethen and Dale, 2011]. En la figura los elementos se muestran etiquetados.

De acuerdo a las propiedades o relaciones que una ER incluya, se clasifica en distintos tipos: relacional o no relacional (también llamada proposicional), minimal, sobreespecificada, subespecificada o parcial, plural o singular. A continuación describimos cada uno de estos tipos.

- Una ER **proposicional** o no relacional, incluye sólo propiedades intrínsecas del objeto target. Por ejemplo, en la Figura 2.1: *La esfera roja*. Las características de ser *esfera* y ser *roja* son propias del objeto identificado y no de otros objetos de su contexto.

- Una ER es **relacional**, cuando contiene relaciones con otros objetos. En este caso la ER incluye descripciones de los otros objetos. Por ejemplo, en la Figura 2.1: *La esfera que está a la derecha del cubo*. *A la derecha de* lexicaliza una relación que necesita describir otro objeto además del target, en este caso un *cubo*.

- Se dice que una ER es **minimal** cuando incluye la mínima cantidad de propiedades y/o relaciones con otros objetos, con las cuales el target puede ser distinguido en el contexto dado. Notar que puede haber muchas expresiones minimales. Por ejemplo, en la Figura 2.1: *La esfera roja* es una ER minimal, como así también lo es *La esfera grande*, y *La esfera a la derecha del cubo*. Para verificar que estas ERs son minimales, intentemos sacar alguna propiedad, si de *La esfera roja* sacamos *esfera*, ya no podemos identificar al target, si sacamos *roja*, tampoco. Lo mismo pasa para *La esfera grande*. Veamos qué pasa con *La esfera a la derecha del cubo*, si sacamos *esfera* tenemos otro objeto, e_7, además del target que también tiene a la izquierda un cubo. Si sacamos *a la derecha de* no es posible construir una frase nominal aceptable.

- Cuando la ER contiene más información de la mínima necesaria para distinguir al target de los demás objetos en el contexto dado, se dice que la ER es **sobreespecificada**. Por ejemplo: *La esfera roja grande* en la Figura 2.1 es sobreespecificada porque se le puede sacar *grande* o *roja* y sigue siendo una ER correcta.

[1]Por simplicidad, en el resto de la tesis vamos a decir directamente "tiene la propiedad *esfera*", cuando queremos decir que el objeto "tiene la propiedad *forma* con valor *esfera*".

- Una expresión es **subespecificada**[2] cuando no puede distinguir al target de los otros objetos en el contexto. Estos otros objetos son llamados **distractores**. Por ejemplo: *La esfera* en la Figura 2.1 no alcanza a identificar al objeto e_5 apuntado por la flecha, ya que hay otras 2 esferas que son distractores, e_1 y e_3. Una expresión subespecificada identifica a un conjunto de objetos y ese conjunto incluye otros objetos además del target.

- Una ER es **plural** cuando el target es un conjunto no singleton. Por ejemplo *Las esferas* es una ER para las 3 esferas de la Figura 2.1. *Las esferas* es una ER **colectiva** que con una sola propiedad describe 3 objetos. Otra ER para el mismo target podría ser *La esfera que está sola, y la que está arriba del cubo y la que está al lado del cubo rojo*, la cual es una expresión **distributiva**.

- Una ER es **singular** cuando el target es singleton, es decir es un sólo objeto. Las ERs de la Figura 2.1 son todos ejemplos de ERs singulares ya que el target (el objeto apuntado por la flecha) es uno solo.

- Una expresión es **parcial** cuando es una ER plural que representa a un subconjunto estricto del target. Por ejemplo, en la Figura 2.1, si quisiéramos identificar las 3 esferas y damos la expresión *Las esferas pequeñas*, sólo identificamos a 2 de las 3 esferas requeridas.

2.1.2 ¿*Cómo* generamos expresiones referenciales?

¿Cómo usamos el lenguaje para transmitir y entender expresiones referenciales? En la teoría propuesta por [Clark, 1992; Clark, 1996] se asume que el hablante usa un principio de diseño óptimo, es decir, los hablantes diseñan sus oraciones de tal manera que sus interlocutores tengan suficiente información para entenderles, pero no más de la necesaria. Para hacer esto modelan la información que es parte del conocimiento mutuo, creencias mutuas y suposiciones mutuas.

En la investigación de [Keysar et al., 1998] se discute que bajo ciertas condiciones los interlocutores dejan de usar el principio de diseño óptimo. Keysar et al. muestran mediante experimentos cuidadosamente diseñados que, al producir expresiones referenciales, las personas no consideran el punto de vista del receptor desde el principio, sino que lo consideran en una etapa de revisión. Es decir, los adultos producen sus ERs egocéntricamente, como hacen los niños, pero las corrigen antes de decirlas para que el receptor sea capaz de identificar el target unívocamente. En este trabajo se argumenta que esta primer etapa egocéntrica es un proceso heurístico que se basa en una representación de la *saliencia* de las características del contexto. Keysar et al. discute que una posible razón para este proceso de generación heurística y posterior ajuste son las limitaciones cognitivas de la mente humana. Estas limitaciones se hacen particularmente evidentes bajo restricciones de tiempo como muestran los experimentos de Keysar et al. *Cómo* se generan las ERs, tiene un impacto en *qué* ERs se generan. Keysar et al. analizan que el modelo propuesto por ellos de generación-y-ajuste podría explicar en parte el tipo de sobreespecificación que se observa en las ERs generadas por personas. Durante el proceso heurístico inicial, la persona incluye más propiedades y/o relaciones que las necesarias para identificar al target, guiada por una representación de la saliencia en el contexto dado.

Como dijimos, normalmente las personas cuando hablan, dan más información de la mínima necesaria para identificar al target, es decir sobreespecifican. Como este es un fenómeno tan frecuente, hay mucho trabajo relacionado. En esta sección describimos el trabajo que fue relevante para el diseño de nuestros algoritmos que se proponen en el Capítulo 4. En la Sección 4.5 explicamos de qué manera este trabajo influenció nuestro diseño.

[2]Estrictamente hablando, una expresión subespecificada no es una ER dado que no identifica unívocamente al target. Sin embargo, en esta tesis hablaremos en ocasiones de ERs subespecificadas para referirnos a éstas expresiones que no alcanzan a ser referenciales.

En [Arts et al., 2011] se estudió cómo la sobreespecificación afecta a las personas cuando interpretan ERs. Hicieron experimentos sobreespecificando ERs ya sea con información de las características intrínsecas de los objetos (tamaño, color, forma) o de su ubicación (en el eje vertical u horizontal). Los resultados del experimento proporcionaron información sobre el efecto de la sobreespecificación en el tiempo de la identificación del target. Concluyen que las ERs sobreespecificadas conducen a una identificación más rápida del target en contextos complejos, cuando permitieron que el lector tenga una imagen mental completa de la entidad. Por otro lado que delimitan la búsqueda a una parte más específica del contexto. La información adicional sobre la ubicación vertical (arriba, abajo) ha demostrado ayudar más a la velocidad de reconocimiento que la información adicional sobre el eje horizontal (izquierda, derecha).

En [Engelhardt et al., 2006] también se estudia la sobreespecificación. En un experimento mostraron que al menos la tercera parte de los hablantes sobreespecificó sus ERs. Un segundo experimento mostró que los oyentes no juzgan las descripciones sobreespecificadas como peores que las expresiones concisas. Y un tercer experimento, reveló que en contextos simples, las descripciones sobreespecificadas desencadenan movimientos oculares que pueden interpretarse como una indicación de confusión.

En [Luccioni et al., 2015] se investigó el rol que la cantidad de información juega en la coordinación léxica en un entorno virtual interactivo. Experimentaron con 2 conjuntos de personas, a un conjunto se les dieron ERs sobreespecificadas, y al otro minimales. Concluyeron que las ERs sobreespecificadas ayudaron más que las minimales a aprender palabras nuevas en una segunda lengua.

Paraboni et al. en su paper [Paraboni et al., 2015] estudian la sobreespecificación de las ERs, en particular de las ERs relacionales. La hipótesis que evalúan es la siguiente:

> *h1: Dado el objetivo de sobreespecificar una descripción relacional usando una propiedad p extra para el landmark, p debería corresponder a la propiedad más sobresaliente que está disponible en el contexto.*

Un **landmark** de un target en un contexto dado, es un objeto del contexto que está relacionado con el target a través de una relación particularmente saliente. Por ejemplo en la Figura 2.2 el cubo rojo grande que está abajo del target, es el landmark. A fin de verificar la hipótesis, en las imágenes hay propiedades sobre-salientes que tienen una alta discriminatoriedad. Una propiedad o relación es más **discriminatoria** que otra cuando, al ser agregada a una ER, elimina más distractores. Por ejemplo, en la Figura 2.2, el tamaño del landmark es más discriminatoria que su color: agregar *grande* a la expresión *cubo* elimina 3 distractores, mientras que agregar *rojo* sólo elimina 2.

De acuerdo a los resultados de trabajo previo (ver [Pechman, 1989]) se asume que se prefiere incluir el color en una ER antes de incluir el tamaño de un objeto. Además, siguiendo el principio de diseño óptimo de [Clark, 1992] se supone que propiedades unarias como color o tamaño se preferirán frente a relaciones que resultan en descripciones más complejas. En [Paraboni et al., 2015] se argumenta en contra de este trabajo previo mostrando que dependiendo solamente de su poder discriminativo, se puede preferir tamaño antes que color y se pueden preferir expresiones relacionales largas en vez de expresiones proposicionales más cortas. Ilustramos la preferencia por propiedades más discriminatorias evaluando la hipótesis *h1* usando la Figura 2.2. De acuerdo a [Pechman, 1989] se supone que si se decide sobreespecificar el landmark en la ER *la esfera sobre el cubo*, se agregaría el color: *la esfera sobre el cubo rojo*. Si *h1* es cierta, sin embargo, se preferirá generar *la esfera sobre el cubo grande*, porque *grande* es más discriminatoria que *rojo* en el ejemplo. Los experimentos presentados en [Paraboni et al., 2015] muestran que la hipótesis es cierta en la mayor parte de los datos analizados. Este es un resultado interesante para nuestro trabajo dado que refina el trabajo de [Keysar et al., 1998] previamente descripto identificando el poder discriminativo de una propiedad como parte

de su definición de saliencia. Esto se correlaciona con nuestros resultados y el rol del poder discriminatorio en la tarea de aprendizaje automático que describimos en el Capítulo 4.

La discriminación de una propiedad juega un rol central en la tarea de desambiguación. En el artículo buscan probar que la discriminación también es importante cuando no hay nada para desambiguar, como es el caso de la sobreespecificación.

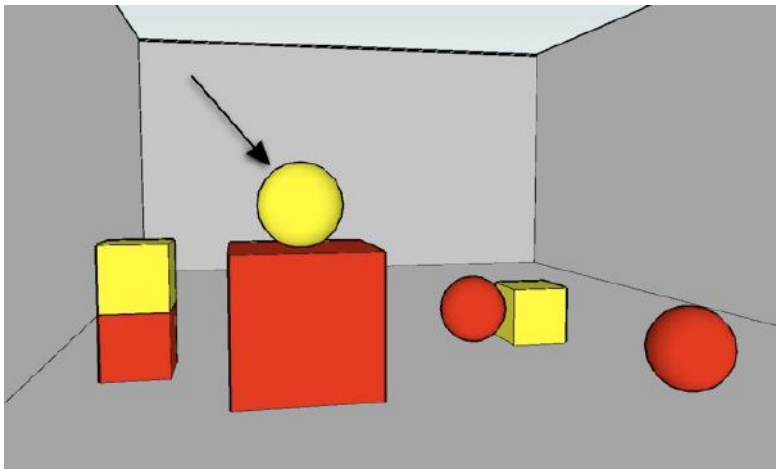


Figura 2.2: Ejemplo de contexto del corpus GRE3D7 en el que el landmark (el cubo grande), tiene una propiedad altamente discriminatoria: su tamaño. El landmark es el único objeto grande de la escena.

2.1.3 Corpora de expresiones referenciales

TUNA [Gatt et al., 2007] fue el primer corpus prominente de ERs disponible públicamente con fines de investigación. El corpus fue desarrollado en una serie de experimentos controlados de propósito general, contiene 2280 descripciones producidas por 60 personas en dos dominios (1200 expresiones referenciales de imágenes de muebles y 1080 expresiones referenciales de fotografías de personas situadas en una grilla). Se muestran ejemplos de imágenes en la Figura 2.3.

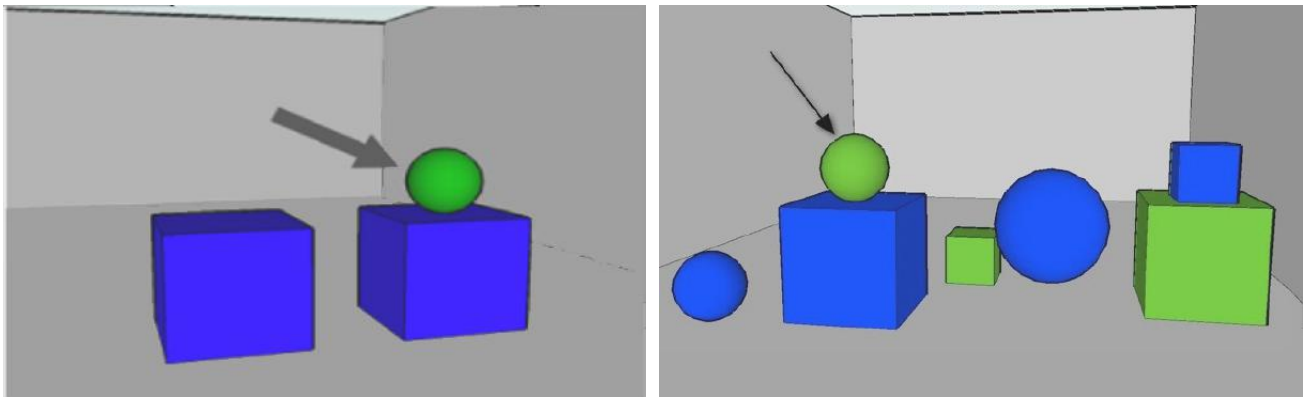
(a) (b)

Figura 2.3: Imágenes del TUNA corpus.

El corpus TUNA no contiene descripciones relacionales. Este corpus se ha utilizado ampliamente en una serie de desafíos [Gatt et al., 2009]. En el Capítulo 6 de esta tesis se usa para evaluar nuestros algoritmos y así comparamos con el desempeño de otros algoritmos del área que participaron en estos desafíos. El corpus contiene sólo una ER por escena, por lo cual no

es útil para comparar con rankings de ERs, y la evaluación de nuestro algoritmo con respecto a este corpus es limitada.

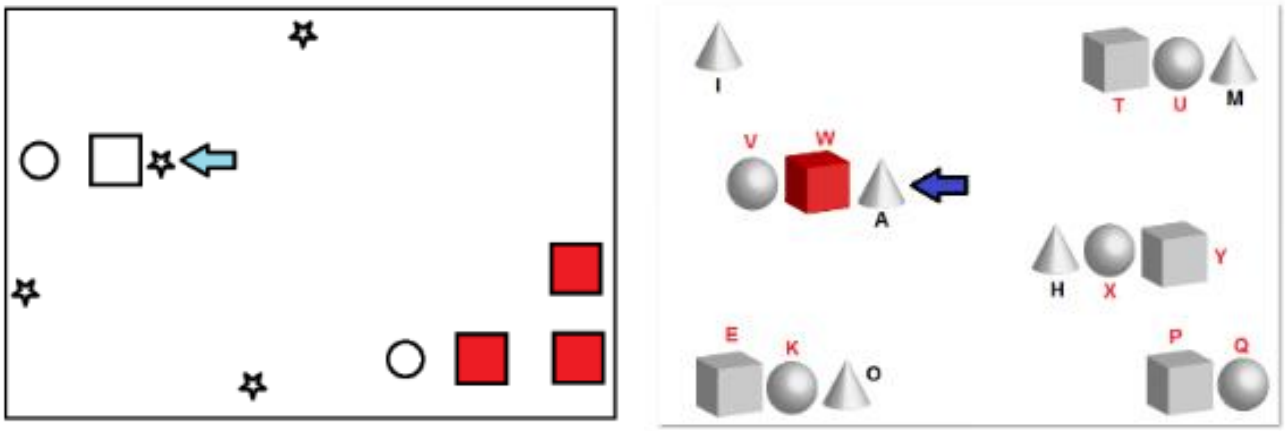
GRE3D3 y su extensión **GRE3D7** [Dale and Viethen, 2009; Viethen and Dale, 2011] se desarrollaron en una serie de experimentos basados en la web, y se centraron principalmente en el estudio de las descripciones relacionales. GRE3D3 contiene 630 descripciones producidas por 63 personas y GRE3D7 contiene 4480 descripciones producidas por 287 personas, y es el corpus más grande del área hasta la fecha. El dominio de estos corpora constan de escenas visuales simples que contienen sólo dos tipos de objetos (cubos y esferas) con variación limitada en color y tamaño. En cada escena, hay una relación espacial entre el target y el landmark más cercano. Ambos corpus contienen descripciones proposicionales y relacionales. Ejemplo de imágenes del GRE3D3 y GRE3D7 se muestran en la Figura 2.4. Las escenas del GRE3D3 contienen sólo 3 objetos mientras que las escenas del GRE3D7 contienen 7 objetos. El GRE3D7 contiene más de 100 expresiones referenciales por escena. Permitiendo una evaluación detallada de los rankings generados por nuestros algoritmos como se describe en el Capítulo 5.



(a) (b)

Figura 2.4: Imágenes del GRE3D3 corpus (a) y del GRE3D7 corpus (b).

Stars y su extensión **Stars2** [Paraboni et al., 2016] se recolectaron para el estudio de la sobreespecificación (particularmente en el caso de las descripciones relacionales).



(a) (b)

Figura 2.5: Imágenes del corpus Stars (a) y del corpus Stars2 (b).

Stars se desarrolló en un experimento piloto basado en la web, contiene 704 descripciones producidas por 64 personas. El conjunto de datos Stars2 se obtuvo de situaciones de diálogo que implicaban a dos personas, contiene 884 descripciones producidas por 56 participantes.

Ambos dominios hacen uso de escenas visuales simples que contienen tres tipos de objetos (por ejemplo para Stars, estrellas, cuadrados y círculos y para Stars2 cubos, conos y esferas) con variación limitada en color y tamaño. A diferencia de otros corpus para GER, Stars/2 incluyen un número considerable de situaciones complejas de referencia en que participan hasta tres objetos, como en *el cubo cerca de la esfera, al lado del cono*. Ejemplos de imágenes se muestran en la Figura 2.5. Este corpus fue recientemente liberado y planeamos usarlo en trabajo futuro.

ZOOM [Altamirano et al., 2015] se desarrolló durante nuestra colaboración con la Universidad de São Paulo. Contiene ERs para 20 mapas de las ciudades de Lisboa y Madrid, fue realizado en la web, y contiene ERs de targets singulares y plurales, con y sin zoom, en 2 idiomas: español y portugués. El corpus portugués contiene 100 ER por mapa, es decir 2000 ERs en total, y el español 80 por mapa, sumando un total de 1600 ERs. El corpus fue recolectado a fin de poder realizar experimentos con situaciones más cercanas a aplicaciones del mundo real, que los corpus existentes hasta el momento. Ejemplos de mapas se muestran en la Figura 2.6.

(a) (b)

Figura 2.6: Imágenes del ZOOM corpus, target plural sin zoom (a) y con zoom (b).

En la Tabla 2.1 se comparan los diferentes corpus presentados anteriormente[3]. El número de propiedades diferentes que tiene en cuenta cada corpus se muestra en columna Propiedades. El número de posibles landmarks que la descripción puede incluir se muestra en la columna Landmarks, el largo promedio de la ER, que es la suma de las propiedades y relaciones de todas las ERs dividido la cantidad total de ERs, se ve en la columna Largo promedio. Y la proporción de propiedades usadas, se muestra en la columna Uso. Esta es la proporción de propiedades que aparecen en la descripción sobre el número total de posibles propiedades y landmarks. Mayores largos de descripción y menor promedio de uso se ven en situaciones complejas de referencia. Como se puede observar, el corpus ZOOM es más complejo en este sentido y es el que usamos para nuestros casos de estudio en el Capítulo 6.

2.2 Generación *automática* de expresiones referenciales.

Un algoritmo para la generación automática de expresiones referenciales es un programa que dado al menos un contexto y un target genera una ER para el target.

Los algoritmos pueden ser de distintos tipos, según los tipos de ERs que sean capaces de generar. Por ejemplo, un algoritmo puede ser: determinístico o no-determinístico, relacional o proposicional, incluir negaciones o no, generar plurales, singulares o ambos, generar ERs sobreespecificadas o minimales. A continuación se describen cada uno de esos tipos.

[3]En el caso del TUNA y el ZOOM se usó para la comparación sólo la parte singular.

Corpus	Propiedades	Landmarks	Largo promedio	Uso
TUNA-Muebles	4	0	3.1	0.8
TUNA-Personas	10	0	3.1	0.3
GRE3D3	9	1	3.4	0.3
GRE3D7	6	1	3.0	0.4
Stars	8	2	4.4	0.4
Stars2	9	2	3.3	0.3
Zoom-Portugués	19	4	6.7	0.3
Zoom-Español	19	4	7.2	0.3

Tabla 2.1: Comparación de corpora de ERs existente.

Un algoritmo es **determinístico** si dado un input (un contexto y un target), da siempre la misma ER de salida. En cambio, un algoritmo es **no-determinístico** si es posible que dé distintas salidas para el mismo input, en distintas ejecuciones. En general las personas generan expresiones referenciales de forma no determinística, en este sentido los algoritmos no-determinísticos simulan el comportamiento de las personas.

Por ejemplo, un algoritmo no-determinístico podría generar para la Figura 2.1, una vez *La esfera roja* y otra vez *La esfera grande*. En la Tabla 2.2 se muestran las distintas ERs dadas por las personas para la Figura 2.1 del corpus GRE3D7 [Viethen and Dale, 2011]. La tabla también muestra cuántas personas generaron cada ER para el target de la figura. Por ejemplo *large red ball* ocurrió 71 veces en el corpus, es decir más de la mitad de las personas decidieron incluir el tipo, el tamaño y el color en su ER, generando así una ER sobreespecificada.

Si el largo de la ER es finito, dado un contexto finito, existen finitas ERs a generar, así un algoritmo no-determinístico nos permitiría explorar el espacio de ERs posibles para un input dado. También nos permite generar un ranking de ERs como se discute en el Capítulo 1.

Orden	ER	Cantidad
1	large red ball	71
2	red ball	56
3	large red ball next-to large red cube	5
4	large ball	2
5	large red ball next-to red cube	2
6	large red ball right-of large red cube	1
7	large red ball next-to large red ball	1
8	large red ball next-to cube	1
9	red ball next-to large red cube	1
Total		140

Tabla 2.2: ERs dadas por las personas para la Figura 2.1 del corpus GRE3D7.

Un algoritmo es **proposicional**, cuando las ERs que genera contienen sólo atributos del target, es decir no contienen relaciones con otros objetos ni propiedades de otros objetos. Por ejemplo, para la Figura 2.1, sólo podría generar las ERs 1, 2 y 4 de la Tabla 2.2. Es decir, genera solamente ERs proposicionales.

Un algoritmo es **relacional** si además de generar ERs proposicionales genera ERs relacionales, en cuyo caso además de generar las relaciones correspondientes deberá generar expresiones para el o los objetos relacionados. Un algoritmo relacional podría generar todas las ERs de la Tabla 2.2. Notar que no todas las descripciones de los objetos involucrados en una ER relacional, son ERs que identifican unívocamente al objeto. Por ejemplo, consideremos la ER 8, que se podría realizar como *La gran esfera roja que está al lado del cubo*. En este caso, *el cubo* es una expresión que se tuvo que dar como consecuencia de incluir la relación *al lado de*.

La expresión *el cubo* no es una ER del landmark si se considera aislada porque hay otros cubos en la escena.

En algunos contextos, por ejemplo cuando el target es el único que no tiene una propiedad o relación, sería útil un algoritmo que incluya **negaciones**. Por ejemplo, para la Figura 2.1, la ER *La esfera que está sola* podría ser una forma efectiva de referirse al objeto e_1 dado que es la única esfera que no está tocado un cubo.

Algunos algoritmos pueden generar ERs para un conjunto de objetos en el contexto considerado. Por ejemplo, para la Figura 2.1 la ER *Los cubos* se refiere al conjunto de objetos $\{e_2, e_4, e_6, e_7\}$. Estos algoritmos generan ERs **plurales**. En caso de sólo poder generar ERs para un target singleton se dice que el algoritmo genera sólo **singulares**.

Un algoritmo que genera ER **minimales** (introducidas en la Sección 2.1.1), es un algoritmo que da ERs que contienen la mínima cantidad de propiedades o relaciones que se necesitan para distinguir al target. Por ejemplo, en la Tabla 2.2 las ER 2 y 4 son minimales. Si el algoritmo es minimal y determinístico, normalmente toma un orden de preferencias de las propiedades como input, que hace que el algoritmo pueda decidir cuál ER dar en caso de tener varias minimales.

Un algoritmo que **sobreespecifique** tiene la característica de poder dar más propiedades o relaciones con otros objetos que las mínimas necesarias para identificar al target. Por ejemplo, para la Figura 2.1 la ER *La esfera roja, grande que está a la derecha del cubo rojo grande* es una ER sobreespecificada porque (como ya mencionamos en la Sección 2.1.1) podríamos sacarle algunas propiedades o la relación y seguiría siendo una ER. Sacando *roja* de las propiedades del target, la expresión *La esfera grande que está a la derecha del cubo rojo grande* sigue siendo una ER, también podríamos sacar la relación con *el cubo rojo grande*, quedando *La esfera roja, grande* y sigue siendo una ER. Viendo la Tabla 2.2 podemos notar que la ER de mayor frecuencia, es *large red ball* y es una ER sobreespecificada, ya que *red ball* o *large ball*, son ERs minimales para el target de la Figura 2.1. En la tabla hay 58 ER minimales y 82 sobreespecificadas. Es decir, el 59% son sobreespecificadas y el 41% minimales.

Hay una forma trivial de que un algoritmo sobreespecifique: agregar todas las propiedades y relaciones del target con otros objetos. Por ejemplo, para la Figura 2.1 la ER sería *Large red ball left-of (large red cube right-of large red ball)*. Pero, como vemos en la Tabla 2.2, esto no es lo que hace la gente. En esta tesis, uno de los temas que investigamos es cómo hacer para el que un algoritmo sobreespecifique imitando el comportamiento humano tanto como sea posible.

Los algoritmos no generan expresiones parciales ni subespecificadas, ya que no sirven para identificar al target. La identificación del target es el objetivo de la GER.

2.2.1 Primeros algoritmos

En esta sección vamos a hablar de una selección de algoritmos de generación automática de expresiones referenciales del área que son particularmente relevantes para la tesis. Para un estado del arte del área más detallado ver [Krahmer and van Deemter, 2012].

Full brevity: El algoritmo **Full Brevity** [Dale, 1989] genera la descripción más corta que identifica al target. Es decir, genera ERs minimales. Para hacerlo, busca si hay una propiedad del target que no sea propiedad de ningún distractor. Si no hay chequea todas las posibles combinaciones de 2 propiedades, si no la hay, busca de a 3 y así sucesivamente. Por ejemplo para la Figura 2.1 se fijaría en las propiedades del target *red, ball, large*, como ninguna de ellas aisladamente identifica sólo al target, probaría con combinaciones de 2 propiedades *red ball* si identifica al target, devuelve *red ball* y finaliza.

Heurística Greedy: El algoritmo de **Heurística Greedy** [Dale, 2002] iterativamente selecciona la propiedad que elimina más distractores, y argumentan que la propiedad seleccionada

tiene el más alto poder discriminativo en esa etapa. Como resultado no siempre genera expresiones referenciales mínimas. Por ejemplo para la Figura 2.1 se fijará que *ball* elimina 2 distractores e_1, e_3, *red* elimina 3 distractores e_2, e_4, e_7 y *large* elimina 1 distractor e_4, entonces elegirá *red*, pero como no alcanza para ser ER, seguirá con *ball* que es la que elimina más distractores luego de *red*, finalizará porque *red ball* identifica al target. Supongamos que hubiera otro objeto con propiedad *large*, entonces *large* eliminaría también 2 distractores, entonces el algoritmo devolvería *large red ball*, y esa ER no es minimal, es sobreespecificada. Este algoritmo es más eficiente que el Full Brevity porque encontrar la descripción más corta es computacionalmente caro. Aunque el algoritmo de heurística Greedy es más eficiente que el Full Brevity, pronto fue superado por el algoritmo *Incremental* y sus sucesores [Reiter and Dale, 1992; Dale and Reiter, 1995]. El algoritmo Incremental fue y sigue siendo uno de los algoritmos más importantes del área, lo explicamos a continuación.

Incremental: El input del **Algoritmo Incremental**, es el target r, que queremos identificar, D es el contexto, y *Pref* una lista de propiedades ordenada según la preferencia. El algoritmo se muestra en la Figura 2.7.

```
1.  IncrementalAlgorithm ({r}, D, Pref){
2.  L ← ∅
3.  C ← D − {r}
4.  for each A_i in list Pref do
5.      V = Value(r, A_i)
6.      if C ∩ RulesOut(⟨A_i, V⟩) ≠ ∅
7.      then L ← L ∪ {⟨A_i, V⟩}
8.           C ← C − RulesOut(⟨A_i, V⟩)
9.      endif
10.     if C = ∅
11.     then return L
12.     endif
13. return failure }
```

Figura 2.7: Algoritmo Incremental, Figura 2 de [Krahmer and van Deemter, 2012].

Vamos a ejemplificar la corrida del algoritmo con el ejemplo de la Figura 2.1, supongamos que el target es e_5 y la lista ordenada de propiedades es ésta [tipo, color, tamaño]. D inicialmente es el conjunto de todos los objetos del contexto: $\{e_1,e_2,e_3,e_4,e_5,e_6,e_7\}$. En *Paso 2* se asigna a L la descripción vacía, al finalizar la ejecución, L tendrá el conjunto de propiedades con los cuales identificaremos a e_5, es decir la ER. En el *Paso 3* se inicializa C con el conjunto de distractores del target r, y e_5 en nuestro ejemplo $\{e_1,e_2,e_3,e_4,e_6,e_7\}$. La idea del algoritmo es ir eliminando distractores usando las propiedades A_i del target en el orden de preferencia *Pref*, por eso, en el *Paso 4* recorre las propiedades A_i. En *Paso 5* le asigna a V el valor que tiene la propiedad A_i para el target r, en nuestro ejemplo e_5. $RulesOut(A_i, V)$ es el conjunto de objetos que tienen diferente valor para la propiedad A_i que el que tiene el target, la función se fija si el valor de esa propiedad elimina distractores. La primer propiedad del target e_5 a considerar según el orden de preferencias *Pref* es el tipo, el valor del target para tipo es *esfera*. En el *Paso 6*, identifica los objetos en C que tengan tipo con valor distinto de esfera, ellos son $\{e_2,e_4,e_6,e_7\}$, y los elimina de C, el cual queda sólo con $\{e_1,e_3\}$. En *Paso 10* pregunta si C es vacío, es decir si ya se eliminaron todos los distractores, pero no lo es, por lo tanto continúa con la siguiente propiedad, en este caso con el color, el valor de color para el target es *rojo*, agrega *rojo* a L, y actualiza C con $\emptyset$ porque tanto e_1 como e_3 son amarillos, es decir no comparten el color con el

target, en *Paso 10* pregunta si C es vacío, y si lo es, por lo tanto devuelve {*esfera*, *rojo*}. Lo cual se podría realizar como *La esfera roja*, y sería una ER para el target considerado.

Luego se propusieron extensiones del algoritmo Incremental, por ejemplo, una extensión que permite la generación de referencia teniendo en cuenta la prominencia discurso del target se muestran en [Krahmer and Theune, 2010; Krahmer and Theune, 2002].

Algoritmos relacionales: Los algoritmos que describimos hasta ahora son proposicionales. Varios investigadores han intentado ampliar el algoritmo incremental permitiendo descripciones relacionales [Horacek, 1997; Krahmer and Theune, 2002; Kelleher and Kruijff, 2006], se basan a menudo en el supuesto de que las propiedades relacionales (como "x está en y") son menos preferidas que las no relacionales (como "x es blanca"), o sea que los algoritmos sólo las generan como última opción. Si se requiere una relación para distinguir al target x, en el cual se deba nombrar al objeto y, se aplica el algoritmo básico iterativamente a y, pero como dijimos antes, esto no siempre es lo que las personas hacen, muchas veces la descripción del landmark no es una ER. Además, no está claro que las propiedades proposicionales sean siempre preferidas antes que las relacionales. En [Viethen, 2011] se sugieren que, incluso en escenas simples, donde los objetos pueden fácilmente ser distinguidos sin relaciones, las personas también utilizan con frecuencia las relaciones (en aproximadamente un tercio de las ERs que dan).

En caso de generar ERs relacionales con esta estrategia hay que asegurarse que el algoritmo termina, es decir que no entra en ciclos infinitos tratando de identificar objetos. En [Dale and Haddock, 1991b] se estudia cómo abordar el problema de regresión infinita, en el cual el algoritmo trata de describir al landmark haciendo referencia al target, y al target haciendo referencia al landmark infinitamente, como en *el libro en la mesa la cual soporta un libro en la mesa...*

2.2.2 Algoritmo de búsqueda en grafos

El **Algoritmo Graph** de [Krahmer et al., 2003] propone tratar la generación de expresiones referenciales como un problema de grafos.

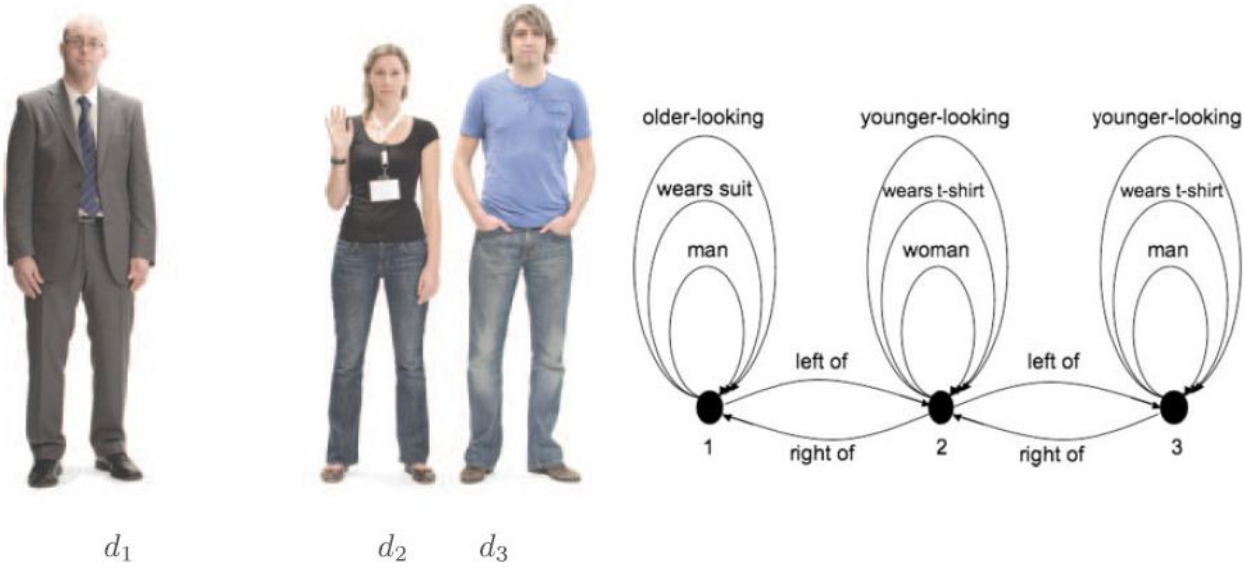


Figura 2.8: Ejemplo de contexto y grafo extraído de [Krahmer and van Deemter, 2012].

El contexto que incluye al target y los distractores es representado como un grafo, por ejemplo en la Figura 2.8 se muestra el contexto y el grafo correspondiente. Cada objeto de la escena (personas en este caso) se modela como un vértice en el grafo. Las propiedades atómicas como `younger-looking`, `older-looking`, `wears suit`, `wears t-shirt`, `woman`, `man`, se representan como un bucle en el correspondiente nodo. Las relaciones binarias entre objetos, por ejemplo `left-of` o `right-of` se modelan como aristas entre los nodos correspondientes. Dado un objeto target, conseguir una ER que distinga al objeto target de los demás objetos del contexto, es equivalente a encontrar un subgrafo del grafo original que únicamente caracterice al target. Intuitivamente este subgrafo se puede poner sobre el target y no sobre ningún otro objeto del dominio considerado.

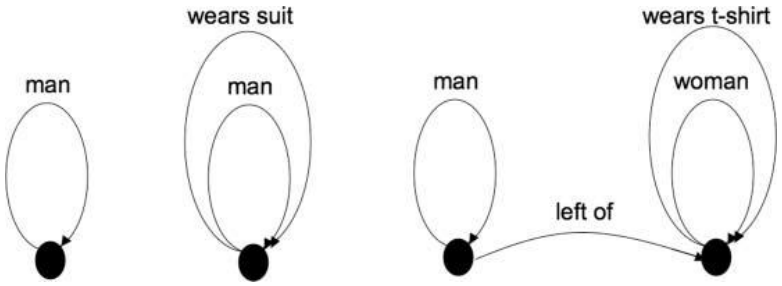


Figura 2.9: Subgrafos del grafo de la Figura 2.8.

Comenzando con el subgrafo que contiene un solo vértice, que representa al target, según una heurística basada en costos (costo de incluir propiedades, relaciones) empieza a agregar propiedades o relaciones, del target o nodos que ya hayan sido agregados. Cada vez que agrega algo, chequea si hay algún otro nodo el cual el grafo pueda distinguir, si lo hay quiere decir que es un distractor, cuando no hay distractores, el grafo distingue unívocamente al target. Por ejemplo en la Figura 2.9 se ve el primer grafo *man*, el cual puede ser puesto sobre los nodos 1 y 3 de la Figura 2.8, es decir no identifica sólo al target, el segundo grafo *man, wears suit* solamente puede ser puesto sobre el nodo 1, por lo tanto es un grafo que distingue al target d_1.

El algoritmo sigue explorando grafos y siempre se queda con el de menor costo.

La función de costo está definida sobre las aristas y vértices del grafo dominio. El costo de un subgrafo se define como la suma sobre todas las aristas y vértices que contiene el grafo. El algoritmo de búsqueda garantiza encontrar el subgrafo de menor costo que representa al target.

La función de costo es usada para podar las ramas del árbol de búsqueda cuando estas se hacen más costosas que el grafo de menor costo encontrado hasta el momento. Esta función hace que se prefieran propiedades sobre otras que tienen mayor costo.

En la Tabla 2.3 se muestra una clasificación de los algoritmos nombrados, según los tipos de algoritmos que vimos en la sección anterior, considerando si es determinístico o no-determinístico, proposicional o relacional, si genera negaciones, plurales, si genera sobreespecificación. Notar que si no generan sobreespecificación, generan ERs minimales.

Algoritmos	Determinístico	No-Determinístico	Proposicional	Relacional	Negaciones	Plurales	Sobreespecificado
Full Brevity	Si	No	Si	No	No	No	No
Greedy	Si	No	Si	No	No	No	Si
Incremental	Si	No	Si	No	No	No	No
GRAPH	Si	No	Si	Si	No	Si	Si
relacional	Si	No	Si	Si	No	No	No

Tabla 2.3: Clasificación de algoritmos según el tipo de ER que generan.

En lo que sigue veremos cómo evaluar a los algoritmos de GER.

2.3 Evaluación y comparación con corpus

Esta sección está dividida en 3 partes, en la primera se muestra trabajo previo usando corpora relevante a nuestro estudio, en la segunda se explican algunas métricas automáticas usadas en el área y en la tercera se da una introducción a las métricas manuales.

2.3.1 Trabajo previo en evaluación usando corpora

Los temas generales que con la evaluación basada en corpus de la experiencia de [Viethen, 2011] quedaron al descubierto fueron (1) la interdependencia estrecha entre algoritmos y la representación subyacente de conocimiento que utilizan, (2) el no-determinismo de la generación del lenguaje natural, (3) la cuestión de cómo comparar la salida de los algoritmos con los gold-standard, y (4) la independencia del dominio específico deseada para los algoritmos de GER.

La discusión de estos temas ha dado lugar a la siguiente lista de evaluación de GER basada en corpus:

- Si el corpus está destinado para ser reutilizado para la evaluación comparativa de diferentes algoritmos, una representación subyacente del dominio, es decir de la semántica, debe ser proporcionada para ser usada por todos los algoritmos.

- El corpus debe contener tantos casos como sea posible de tantos diferentes hablantes como se pueda conseguir para cada escenario referencial para identificar las potencialidades del contexto.

- Si la probabilidad de un algoritmo de ser un modelo de la conducta humana de referencia se evalúa, deben utilizarse métricas basadas en cobertura y exactitud. Es decir, el conjunto completo de las descripciones que el algoritmo proporciona para cada escenario de referencia en virtud de cualquier ajuste de parámetro debe ser comparado con el conjunto de las descripciones contenidas en el corpus para el mismo escenario referencial.

- Algoritmos que son juzgados en un dominio específico, no se debe asumir como fácilmente adaptables a otros dominios. Los algoritmos deben ser evaluados en distintos dominios.

Investigación en GER pre-2000 dio poca o ninguna atención a la evaluación empírica de los algoritmos. Más recientemente, sin embargo, los estudios de evaluación GER han comenzado a usar corpora para evaluar.

En esta sección, se discutió cuáles son los criterios que se deben cumplir para evaluar adecuadamente los algoritmos GER.

Para medir la performance de los algoritmos podemos usar métricas automáticas o métricas manuales, las métricas automáticas son aquellas que se calculan mediante un algoritmo y las manuales en las cuales les requerimos a personas que evalúen las expresiones referenciales como se describe en las siguientes secciones.

2.3.2 Métricas automáticas

Una de las formas más usadas de evaluación son las siguientes métricas que se pueden calcular automáticamente con respecto a corpora. Una métrica automática de evaluación implica comparar la ER dada por el sistema para un target en un contexto dado con la ER (gold-standard) dada por una persona para el mismo target y contexto. Esta comparación de ER puede estar dada a distintos niveles, podemos comparar si son iguales, si sólo difieren en el orden de las palabras, si difieren en las palabras pero no en la cantidad de palabras que contienen, etc. En lo que sigue nombramos las métricas de evaluación automática más usadas.

La **exactitud** (*accuracy*) se define como el porcentaje de coincidencias exactas entre cada ER producida por un ser humano y la producida por el sistema para la misma escena y target. Se considera que es una métrica demasiado estricta.

El coeficiente **Dice** es una métrica de comparación de conjuntos, el valor va entre 0 y 1, 1 indica un matcheo perfecto entre los conjuntos. Para dos ERs A y B, Dice se calcula como sigue:

$$Dice(A, B) = \frac{2 \times |A \cap B|}{|A| + |B|}$$

MASI de [Passonneau, 2006] es un coeficiente que varía en favor de la similaridad cuando un conjunto es un subconjunto de otro, como Dice varía entre 0 y 1, 1 indica matcheo perfecto. Se calcula como sigue:

$$\text{MASI}(A, B) = \delta \times \frac{|A \cap B|}{|A \cup B|}$$

donde δ es un coeficiente definido como sigue:

$$\delta = \begin{cases} 0 & if A \cap B = \emptyset \\ 1 & if A = B \\ \frac{2}{3} & if A \subset B \text{ or } B \subset A \\ \frac{1}{3} & otro\ caso \end{cases} \tag{2.1}$$

Intuitivamente significa que se prefieren aquellas descripciones producidas por el sistema las cuales no incluyen atributos que los humanos no incluyeron.

Notar que estas métricas no cumplen con el punto (iii) de la Sección 2.3.1 y no nos permiten comparar rankings de ERs, pero igualmente las usamos en el Capítulo 5 para comparar la salida de nuestro algoritmo con trabajo previo.

2.3.3 Métricas manuales

Las personas pueden actuar como jueces humanos, a los cuales les podemos hacer preguntas para evaluar distintas cosas, por ejemplo:

- Adecuación - mostrándole a un juez humano la misma imagen que vio la persona que generó la ER y preguntándole cosas como ¿Qué tan clara es la descripción?.
- Fluidez - Preguntando ¿Considera que es fluida esta descripción?.
- Claridad - Preguntando ¿Es buena y clara en el idioma considerado?.

Otra manera de evaluar manualmente es viendo si las ERs son útiles para dar instrucciones. Por ejemplo, [Belz and Gatt, 2008], mostraron a los participantes una descripción generada para un ensayo. Después que los participantes leían esta descripción, una escena aparecía y se pidió a los participantes hacer clic en el objetivo previsto. Esto les permitió calcular tres métricas de evaluación:

- Tiempo de lectura.
- Tiempo de identificación.
- Tasa de error (que se define como el número de objetivos identificados incorrectamente).

Eye-tracking otra forma de evaluar los ítems mencionados anteriormente, viendo la cantidad de movimientos que realizan los ojos antes de encontrar el objetivo.

Hicimos una evaluación manual en la que pedimos a 2 jueces que elijan entre dos ERs la más natural, se les mostró el contexto, y se les presentó 2 ERs una dada por el sistema y otra proveniente del corpus dada por un humano. También se le podría mostrar la misma imagen pero sin la flecha que señala al target, darle la ER que dio alguna persona, y pedirle que señale el objeto al cual la ER refiere. De esta manera se puede evaluar facilidad de encontrar objetivo con la expresión dada.

2.4 Notas finales y linkeo del capítulo

Este capítulo lo dividimos en 3 secciones, en la primer sección dimos definiciones básicas que usamos a lo largo de toda la tesis, como son los tipos de ERs de acuerdo a las propiedades o relaciones que incluyan, también de acuerdo a la cantidad de información que tengan, luego presentamos las premisas en las que se basa la teoría de [Clark, 1992; Clark, 1996; Clark and Marshall, 1981], vimos experimentos que muestran que dadas ciertas condiciones las personas no siguen esas premisas, [Keysar et al., 1998]. También vimos que algo común es que las personas den ERs sobreespecificadas, [Arts et al., 2011; Engelhardt et al., 2006], los experimentos indican que por lo menos la tercera parte de las expresiones son sobreespecificadas, y que los oyentes no juzgan estas ERs como peores que la minimales. Otras conclusiones encontradas son que la sobreespecificación permite la identificación más rápida del target y que la información adicional (arriba, abajo) es más útil que la (izquierda, derecha). En [Luccioni et al., 2015] se muestra un experimento de aprendizaje de palabras en una lengua extranjera, y concluyen que las personas que tuvieron a su disposición ERs sobreespecificadas, aprendieron más que aquellas a las que se les dio ERs minimales. Dimos una introducción a corpora existente en el área como son el TUNA, el GRE3D3/7, Stars/2 y el ZOOM corpus, éste último fue creado en el desarrollo de esta tesis, por lo cual lo explicaremos con más detalle en el Capítulo 6. En la segunda sección explicamos los diferentes tipos de algoritmos los cuales se diferencian por los tipos de ERs que pueden generar, dimos los algoritmos más importantes del área como son incremental, graph, y relacional entre otros, vimos sus diferencias y pudimos compararlos entre ellos. Entre los algoritmos del área no incluimos el de bisimulación del cual hablaremos en el Capítulo 4 por ser éste el algoritmo base de los aportes de esta tesis. En la tercer sección de este capítulo dimos una introducción a las métricas de evaluación tanto automáticas como manuales, algunas de las cuales usaremos para evaluar nuestro algoritmo como se muestra en el Capítulo 5.

Capítulo 3

Usando teoría de modelos

En el Capítulo 1 (Sección 1.2) introdujimos terminología y conceptos básicos de lógica y de teoría de modelos y motivamos su uso para la GER. De acuerdo a la complejidad de las estructuras gramaticales que queramos permitir en las ERs, tendremos un lenguaje asociado, el cual pertenecerá a alguna lógica. En este capítulo se explica cómo partiendo de fórmulas de primer orden ($\mathcal{FO}$) podemos ir restringiendo la lógica y con ello el lenguaje, y así conseguir una menor complejidad computacional. Es interesante evaluar la complejidad dependiendo de la lógica seleccionada, ya que, por ejemplo, sistemas de tiempo real necesitan conseguir ERs rápidamente. Daremos la complejidad de las distintas lógicas, y veremos qué lógica aplica a cada algoritmo del estado del arte del área introducidos en el Capítulo 2.

Este capítulo está dividido en 5 secciones. En la Sección 3.1 daremos nociones de similaridad entre objetos del modelo en un lenguaje particular y explicamos cómo se relacionan con la GER. Pero como la similaridad involucra la verificación de infinitas fórmulas, veremos simulaciones que es un concepto matemático que nos va a ayudar a resolver el problema de GER sin tener que verificar infinitas fórmulas. Luego en la Sección 3.2 veremos cómo computar ERs en $\mathcal{EL}$ usando simulaciones. En la Sección 3.3 veremos cómo computar ERs en $\mathcal{ALC}$, y en la Sección 3.4 en $\mathcal{FO}^-$. Para finalizar en la Sección 3.5 daremos un resumen del capítulo y contaremos como se linkea esta parte de la tesis con los demás capítulos. Parte de los resultados mostrados en este capítulo fueron publicados en [Areces et al., 2008; Areces et al., 2011].

3.1 Objetos indistinguibles en un lenguaje lógico $\mathcal{L}$

En esta sección veremos dos conceptos importantes, similaridad en $\mathcal{L}$, y simulación en $\mathcal{L}$. Recordemos que en la Sección 1.2 definimos la salida del problema GER con el lenguaje $\mathcal{L}$ (es decir $\mathcal{L}$-GER) como una fórmula φ en $\mathcal{L}$ cuya interpretación en el modelo de entrada $\mathcal{M}$ es el conjunto target T, si esa fórmula existe. Formalmente, $\mathcal{L}$-GER($\mathcal{M}$, T)=$\{\varphi$ en $\mathcal{L} \mid \|\varphi\| = \mathrm{T}\}$ $\vee \perp$ en caso de no existir ninguna fórmula.

3.1.1 Los objetos indistinguibles en $\mathcal{L}$ son $\mathcal{L}$-*similares*

Considere la Figura 3.1, asumamos que el lenguaje en el que podemos generar ERs, es decir, nuestro $\mathcal{L}$, es $\mathcal{C}$. $\mathcal{C}$ es el lenguaje lógico que sólo puede tener conjunciones de propiedades proposicionales, sin negación y sin relaciones. Sea REL = $\{$*large*, *small*, *red*, *ball*, *cube*, *yellow*$\}$ nuestro vocabulario, es decir, el conjunto de relaciones (unarias) que podemos usar en las ERs.

Ahora si buscamos una ER para el target T = $\{e_5\}$ en la Figura 3.1 usando el lenguaje $\mathcal{C}$, veremos que no existe una ER en $\mathcal{C}$ para e_5. Observando el modelo de la figura vemos que las fórmulas que son ciertas para e_5 de $\mathcal{C}$ y sus interpretaciones son las siguientes:

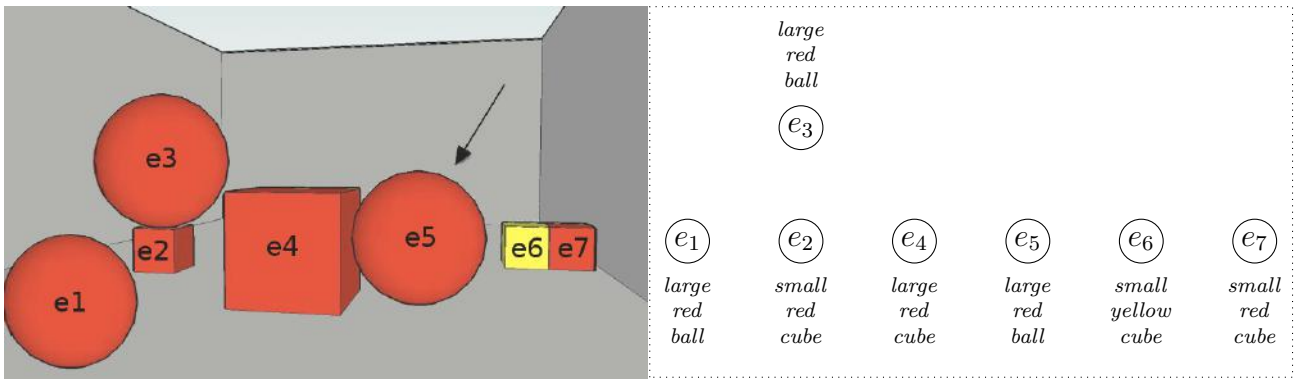


Figura 3.1: Contexto y modelo sólo con propiedades proposicionales.

- Con 1 proposición: $\|red\|=\{e_1,e_2,e_3,e_4,e_5,e_7\}$, $\|large\|=\{e_1,e_3,e_4,e_5\}$, $\|ball\|=\{e_1,e_3,e_5\}$
- Con 2 proposiciones: $\|red \wedge ball\|=\{e_1,e_3,e_5\}$, $\|ball \wedge large\|=\{e_1,e_3,e_5\}$, $\|large \wedge red\|=\{e_1,e_3,e_4,e_5\}$
- Con 3 proposiciones: $\|large \wedge red \wedge ball\|=\{e_1,e_3,e_5\}$

Concluimos que e_5 no puede distinguirse de e_1 y e_3 usando sólo propiedades proposicionales y conjunción que es lo que $\mathcal{C}$ nos permite. Decimos que e_5 es $\mathcal{C}$-similar a e_1 y a e_3 y por lo tanto no es posible identificar unívocamente a e_5 usando el lenguaje $\mathcal{C}$.

Definición 1. *Dado un lenguaje lógico $\mathcal{L}$ decimos que un objeto u en un modelo $\mathcal{M}$ es* **$\mathcal{L}$-similar** *a otro objeto v del modelo si para toda fórmula en $\mathcal{L}$ satisfecha por u también es satisfecha por v.*

Formalmente sea $\mathcal{M} = \langle \Delta, \| \cdot \| \rangle$ un modelo relacional con u, $v \in \Delta$ $u \overset{\mathcal{L}}{\rightsquigarrow} v$ (u $\mathcal{L}$-similar v) si para toda fórmula φ en $\mathcal{L}$, $u \in \|\varphi\| \Rightarrow v \in \|\varphi\|$. La $\mathcal{L}$-similaridad captura la noción de "capacidad de identificar en $\mathcal{L}$".

La noción de $\mathcal{L}$-similaridad entonces, nos permite identificar aquellos elementos del modelo que pueden o no ser referenciados usando un lenguaje dado $\mathcal{L}$. En la Figura 3.1 sólo e_4 (*el cubo grande*) y e_6 (*el cubo amarillo*) pueden ser referenciados en $\mathcal{C}$. Y por lo tanto, nos permitiría identificar qué lenguaje es necesario para poder generar ERs para los elementos de un modelo dado. En nuestro ejemplo tenemos que $e_5 \overset{\mathcal{C}}{\rightsquigarrow} e_1$ y $e_5 \overset{\mathcal{C}}{\rightsquigarrow} e_3$. Por lo que $\mathcal{C}$-GER no tiene solución, para este ejemplo es $\bot$. Si usamos un lenguaje más expresivo como $\mathcal{EL}$ que permite usar relaciones binarias, la fórmula $ball \wedge \exists rightof.cube$ es una ER para el target del ejemplo.

Sin embargo, no es posible computar $\mathcal{EL}$-GER para nuestro ejemplo dado que, como está planteada, la noción de $\mathcal{L}$-similaridad requiere computar infinitas fórmulas. Las fórmulas en $\mathcal{EL}$ que describen al target e_5 son infinitas porque hay un ciclo en el modelo.

La solución para no tener que considerar infinitas fórmulas es reformular la noción de $\mathcal{L}$-similaridad de una manera estructural usando la noción de teoría de modelos conocida como **$\mathcal{L}$-simulación**. A diferencia de la $\mathcal{L}$-similaridad que se define sobre infinitas fórmulas, las $\mathcal{L}$-simulaciones se definen sobre el modelo, el cual para la GER, es finito.

3.1.2 Los objetos indistinguibles en $\mathcal{L}$ son $\mathcal{L}$-*simulables*

La relación entre $\mathcal{L}$-similaridad y $\mathcal{L}$-simulación ha sido ampliamente estudiada y el siguiente es un resultado fundamental de teoría de modelos:

Teorema 1. *Si $\mathcal{M} = \langle \Delta, \| \cdot \| \rangle$ es un modelo finito, u y $v \in \Delta$, entonces $u \overset{\mathcal{L}}{\rightsquigarrow} v$ (u $\mathcal{L}$-similar v) si y sólo si $u \overset{\mathcal{L}}{\rightarrow\!\!\!\!\rightarrow} v$ (u $\mathcal{L}$-simula v) (para $\mathcal{L} \in \{\mathcal{FO}, \mathcal{FO}^-, \mathcal{ALC}, \mathcal{EL}, \mathcal{EL}^+\}$).*

$\mathcal{L}$	Descripción	ATOM_L	ATOM_R	REL_L	REL_R	INJ_L	INJ_R
$\mathcal{FO}$	lógica de primer orden	×	×	×	×	×	×
$\mathcal{FO}^-$	$\mathcal{FO}$ sin negación	×		×		×	
$\mathcal{ALC}$	$\mathcal{FO}$ sin desigualdad	×	×	×	×		
$\mathcal{ELU}_{(\neg)}$	$\mathcal{EL}$ más disyunción y negación proposicional	×	×	×			
$\mathcal{EL}^+$	$\mathcal{EL}$ con negación proposicional	×	×	×			
$\mathcal{EL}$	$\mathcal{ALC}$ sin negación	×		×			
$\mathcal{P}$	lógica proposicional	×	×				
$\mathcal{C}$	sólo conjunciones	×					

Tabla 3.1: $\mathcal{L}$-simulaciones para varios lenguajes lógicos.

Demostración tomada de [Areces et al., 2011] página 34: $\overset{\mathcal{FO}}{\rightarrow}$ es isomorfismo en grafos etiquetados [Ebbinghaus et al., 1996]; $\overset{\mathcal{ALC}}{\rightarrow}$ corresponde a la noción de bisimulación [Blackburn et al., 2001, Def. 2.16]; $\overset{\mathcal{EL}}{\rightarrow}$ es una simulación definida en [Blackburn et al., 2001, Def. 2.77]. Los restantes son variaciones simples de estos.

El Teorema 1 nos dice que, en modelos finitos, las simulaciones capturan exactamente el concepto de similaridad. El teorema no se mantiene en general sobre modelos infinitos.

También nos dice que si 2 elementos distintos u y v en Δ son tales que $u \overset{\mathcal{L}}{\rightarrow} v$ entonces para toda $\mathcal{L}$-fórmula que es verdadera para u es también verdadera para v. En ese caso no hay fórmula en $\mathcal{L}$ que pueda identificar unívocamente a u. Desde esta perspectiva, conociendo cuándo el modelo contiene un elemento que es $\mathcal{L}$-similar pero distinto de u es equivalente a decidir si hay una expresión referencial en $\mathcal{L}$ para u.

Formalmente, para un modelo relacional $\langle \Delta, \| \cdot \| \rangle$, diremos que $u \overset{\mathcal{L}}{\rightarrow} v$ (v $\mathcal{L}$-simula u) si **existe** una relación no vacía $\text{Q} \subseteq \Delta \times \Delta$ que satisface **todas** las propiedades indicadas para $\mathcal{L}$ en la Tabla 3.1. Por ejemplo, la Tabla 3.1 muestra que para $\mathcal{EL}$, la relación Q debe satisfacer ATOM_L y REL_L. A continuación se definen las propiedades de Q.

ATOM_L : Si u Q v, entonces $u \in \|p\| \Rightarrow v \in \|p\|$
ATOM_R : Si u Q v, entonces $v \in \|p\| \Rightarrow u \in \|p\|$
REL_L : Si u Q v y $(u, w) \in \|r\|$, entonces $\exists x$ tal que w Q x y $(v, x) \in \|r\|$
REL_R : Si u Q v y $(v, x) \in \|r\|$, entonces $\exists w$ tal que u Q w y $(u, w) \in \|r\|$
INJ_L : Q es una función inyectiva (cuando es restringida a su dominio)
INJ_R : Q^{-1} es una función inyectiva (cuando es restringida a su dominio)

Por ejemplo, podemos probar que $e_5 \overset{\mathcal{C}}{\rightarrow} e_1$ en el modelo de la Figura 3.1, sin tener que generar todas las fórmulas φ en $\mathcal{C}$ que son ciertas de e_5, ya que la relación $\text{Q} = \{(e_5, e_1)\}$ satisface ATOM_L. Usando el Teorema 1 concluimos que no hay $\mathcal{C}$-descripción para e_5. Como ya dijimos, si se opta por un lenguaje más rico como $\mathcal{EL}$, se puede describir a e_5. De nuevo, esto se puede hacer sin tener que verificar infinitas fórmulas.

Como veremos en la siguiente subsección, la simulación nos da un enfoque eficiente, computacionalmente factible para atacar el problema de $\mathcal{L}$-GER. Algoritmos para calcular muchos tipos de $\mathcal{L}$-simulaciones son bien conocidos (vea, [Hopcroft, 1971; Henzinger et al., 1995; Dovier et al., 2004; Areces et al., 2008]), y para muchos lenguajes (por ejemplo, $\mathcal{ALC}$, $\mathcal{EL}^+$ y $\mathcal{EL}$) se ejecutan en tiempo polinomial. No se conoce algoritmo polinomial para simulación con $\mathcal{FO}$ o $\mathcal{FO}^-$; la complejidad exacta del problema en estos casos está abierta [Garey and Johnson, 1979]. En la Tabla 3.2 se muestra una comparación entre los algoritmos más conocidos del área, que lógica de descripción (DL) usan y la complejidad.

DL	Algoritmo	Complejidad
$\mathcal{FO}$		NP
$\mathcal{FO}^-$	Algoritmo Graph [Krahmer et al., 2003]	NP
$\mathcal{ALC}$	Algoritmo DL [Areces et al., 2008]	P
$\mathcal{ELU}_{(\neg)}$	Algoritmo plural [Gardent, 2002]	P
$\mathcal{EL}^+$		P
$\mathcal{EL}$	Algoritmo relacional [Horacek, 1997] entre otros	P
$\mathcal{P}$	Extensiones a IA [van Deemter, 2002] entre otros	P
$\mathcal{C}$	Algoritmo incremental y predecesores [Dale and Reiter, 1995]	P

Tabla 3.2: Complejidad de los algoritmos según la lógica que usan.

3.2 Computando expresiones referenciales en $\mathcal{EL}$

Las $\mathcal{L}$-simulaciones nos dan un enfoque eficiente, computacionalmente factible para GER usando un lenguaje $\mathcal{L}$, como veremos a continuación. Si tenemos un lenguaje $\mathcal{L}$ y un modelo $\mathcal{M}$ y queremos referirnos a un elemento u del modelo $\mathcal{M}$, podemos computar el **conjunto de simulación** de u definido como $sim^{\mathcal{M}}_{\mathcal{L}}(u) = \{v \in \Delta \mid u \overset{\mathcal{L}}{\rightrightarrows} v\}$. Si $sim^{\mathcal{M}}_{\mathcal{L}}(u)$ no contiene sólo a u, quiere decir que el modelo $\mathcal{M}$ tiene elementos que son indistinguibles de u y no existen ERs para u en ese lenguaje $\mathcal{L}$. Primero explicamos cómo computar $sim^{\mathcal{M}}_{\mathcal{EL}}(u)$, es decir, los conjuntos de objetos indistinguibles en $\mathcal{EL}$ y luego usamos este cómputo para la GER.

3.2.1 Computando conjuntos de objetos indistinguibles en $\mathcal{EL}$

Un algoritmo para computar $sim^{\mathcal{M}}_{\mathcal{EL}^+}(v)$ es dado en [Henzinger et al., 1995]. Para todos los elementos v de un modelo dado finito $\mathcal{M} = \langle \Delta, \|\cdot\| \rangle$ computa $sim^{\mathcal{M}}_{\mathcal{EL}^+}(v)$ en tiempo $O(\#\Delta \times \#\|\cdot\|)$. Intuitivamente, este algoritmo define $S(v)$ como un conjunto de candidatos para simular v y sucesivamente refina este conjunto borrando aquellos elementos que fallan en simular v. Al final, $S(v) = sim^{\mathcal{M}}_{\mathcal{EL}^+}(v)$. El algoritmo puede ser adaptado para computar $sim_{\mathcal{L}}$ para cualquier otro lenguaje $\mathcal{L}$. En particular, lo podemos usar para computar $sim^{\mathcal{M}}_{\mathcal{EL}}$ en tiempo polinomial y nos dará un límite superior a la complejidad del problema $\mathcal{EL}$-GER. El código es mostrado en el Algoritmo 1, el cual usa la siguiente notación. $\mathcal{P}$ es un conjunto fijo de símbolos de relaciones unarias, para $v \in \Delta$, sea $P(v) = \{p \in \mathcal{P} \mid v \in \|p\|\}$ y sea también $suc_r(v) = \{u \in \Delta \mid (v, u) \in \|r\|\}$ para r un símbolo de relación binaria.

Entrada : un modelo finito $\mathcal{M} = \langle \Delta, \|\cdot\| \rangle$
Salida : $\forall v \in \Delta$, el conjunto de simulaciones $sim^{\mathcal{M}}_{\mathcal{EL}}(v) = S(v)$

foreach $v \in \Delta$ **do**
  $S(v) := \{u \in \Delta \mid P(v) \subseteq P(u)\}$
end
while $\exists r, u, v, w : v \in suc_r(u), w \in S(u), suc_r(w) \cap S(v) = \emptyset$ **do**
  $S(u) := S(u) - \{w\}$
end

Algoritmo 1: Computando $\mathcal{EL}$-similaridad.

Inicializamos $S(v)$ con el conjunto de todos los elementos $u \in \Delta$ tal que $P(v) \subseteq P(u)$, es decir, el conjunto de todos los elementos que cumplan al menos las mismas relaciones unarias que v (esto garantiza que la propiedad ATOM_L se mantiene). En cada paso, si hay tres elementos u, v y w tales que por alguna relación r, $(u, v) \in \|r\|$, $w \in S(u)$ (Es decir, w es un candidato para simular u) pero $suc_r(w) \cap S(v) = \emptyset$ (No hay ningún elemento $w' \in S(v)$) tal que $(w, w') \in \|r\|$

y $w' \in S(v)$) entonces es claro que la condición REL_L no se cumple $S(u)$ es 'demasiado grande' porque w no puede simular u. Por lo tanto w se borra de $S(u)$.

El Algoritmo 1 nos dice si hay o no una expresión referencial en $\mathcal{EL}$ para algún elemento u (esto es, si $sim_{\mathcal{EL}}(u) = \{u\}$ o no). No computa ninguna ER que identifique unívocamente a v. Pero podemos adaptarlo para obtener tal fórmula en $\mathcal{EL}$. La estrategia principal del Algoritmo 1 para computar simulaciones es refinar sucesivamente una aproximación al conjunto de simulación. La razón detrás de cada refinamiento puede ser codificada usando una $\mathcal{EL}$-fórmula.

3.2.2 Computando expresiones referenciales en $\mathcal{EL}$

El Algoritmo 2 muestra una versión transformada del Algoritmo 1. La idea es que cada nodo $v \in \Delta$ es ahora etiquetado con una fórmula $F(v)$ de $\mathcal{EL}$. Las fórmulas $F(v)$ son actualizadas a través de la ejecución del ciclo cuyo invariante asegura que $v \in \|F(v)\|$ y $\|F(u)\| \subseteq S(u)$ se mantienen para $u, v \in \Delta$.

Entrada: un modelo finito $\mathcal{M} = \langle \Delta, \|\cdot\| \rangle$
Salida : $\forall v \in \Delta$, una fórmula $F(v) \in \mathcal{EL}$, y el conjunto de simulaciones de $S(v)$ tal que $\|F(v)\| = S(v) = sim_{\mathcal{EL}}(v)$

foreach $v \in \Delta$ **do**
  $S(v) := \{u \in \Delta \mid P(v) \subseteq P(u)\}$
  $F(v) := \bigwedge P(v)$
end
while $\exists r, u, v, w : v \in suc_r(u), w \in S(u), suc_r(w) \cap S(v) = \emptyset$ **do**
  invariante $\forall u, v : \|F(u)\| \subseteq S(u) \wedge v \in \|F(v)\|$
  $S(u) := S(u) - \{w\}$
  if $\exists r.F(v)$ *no es conjunción de* $F(u)$ **then**
    $F(u) := F(u) \wedge \exists r.F(v)$
  end
end

Algoritmo 2: Computando $\mathcal{EL}$-similaridad y generando ERs para $\mathcal{EL}$.

Inicialmente $F(v)$ es la conjunción de todas las relaciones unarias que satisfacen v (si no hay ninguna, entonces $F(v) = \top$). El algoritmo busca elementos r, u, v, w tales que $(u, v) \in \|r\|$, $w \in S(u)$ y $suc_r(w) \cap S(v) = \emptyset$, actualiza $F(u)$ a $F(u) \wedge \exists r.F(v)$. Esta nueva fórmula φ está en $\mathcal{EL}$ y se puede mostrar que $v \in \|\varphi\|$ y $w \notin \|\varphi\|$, por lo tanto $v \overset{\mathcal{EL}}{\rightsquigarrow} w$ es falso, w no simula a u y se elimina de $S(u)$.

El Algoritmo 2 se puede ejecutar en un subconjunto arbitrario del dominio de $\langle \Delta, \|\cdot\| \rangle$ en $O(\#\Delta \times \#\|\cdot\|)$ pasos como se muestra en [Henzinger et al., 1995]. Por lo tanto el problema $\mathcal{EL}$-GER se puede resolver en tiempo polinomial. Nótese, sin embargo, que este resultado supone una representación conveniente de fórmulas como, por ejemplo, grafos acíclicos dirigidos, para asegurar que cada paso de la construcción de la fórmula puede hacerse en $O(1)$.

El Algoritmo 2 se obtuvo mediante la adición de anotaciones de fórmulas al algoritmo estándar de 'minimización de $\mathcal{EL}$-simulaciones' presentado en la Sección 3.2.1. Dada una minimización de $\mathcal{L}$-Simulación, podemos adaptarlo de manera similar para obtener un algoritmo $\mathcal{L}$-GER. El algoritmo calcula las $\mathcal{L}$-ERs para *todos* los elementos del dominio simultáneamente. Esto hará que sea especialmente adecuado para aplicaciones con dominios que requieren referencias a muchos objetos.

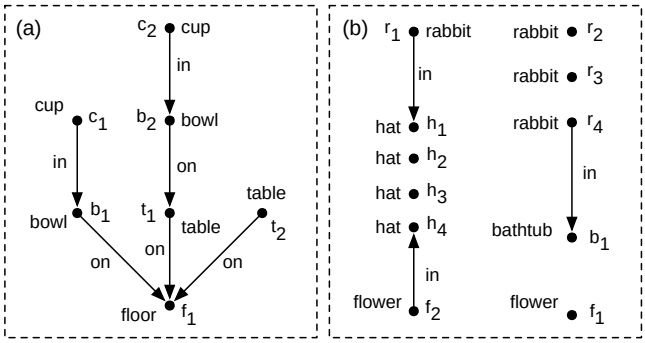


Figura 3.2: Modelo de ejemplo tomado de [Dale and Haddock, 1991a].

3.3 Computando expresiones referenciales en $\mathcal{ALC}$

A continuación daremos una introducción a la lógica de descripción $\mathcal{ALC}$ y la comparamos con $\mathcal{EL}$. Las fórmulas φ de $\mathcal{ALC}$ son generadas por la siguiente gramática:

$$\varphi, \varphi' ::= \top \mid p \mid \neg\varphi \mid \varphi \sqcap \varphi' \mid \exists R.\varphi$$

donde $p \in \mathsf{prop}$, es decir, es el conjunto de los símbolos proposicionales y $R \in \mathsf{rel}$ es el conjunto de los símbolos relacionales. $\mathcal{EL}$ puede definirse como $\mathcal{ALC}$ sin negación. $\mathcal{ALC}$ permite negación tanto atómica (por ejemplo, $\neg\mathsf{flower}$) como sobre relaciones (por ejemplo, $\neg\exists\mathsf{in}.\mathsf{hat}$).

Las fórmulas de $\mathcal{ALC}$ (como las de $\mathcal{EL}$) son interpretadas en modelos relacionales de primer orden $\mathcal{M} = (\Delta, \|\cdot\|)$ donde Δ es un conjunto no vacío y $\|\cdot\|$ es una función de interpretación tal que:

$$\begin{array}{rcl} \|p\| & \subseteq & \Delta \text{ para } p \in \mathsf{prop} \\ \|R\| & \subseteq & \Delta \times \Delta \text{ para } R \in \mathsf{rel} \\ \|\neg\varphi\| & = & \Delta - \|\varphi\| \\ \|\varphi \sqcap \varphi'\| & = & \|\varphi\| \cap \|\varphi'\| \\ \|\exists R.\varphi\| & = & \{i \mid \text{para algún } i', (i, i') \in \|R\| \text{ e } i' \in \|\varphi\|\}. \end{array}$$

La interpretación de cada fórmula lógica denota un conjunto de objetos del dominio; por lo tanto podemos usar fórmulas para describir conjuntos. Por ejemplo en el modelo de la Figura 3.2(b), la fórmula flower denota el conjunto $\{f_1, f_2\}$; La fórmula $\mathsf{flower} \sqcap \exists\mathsf{in}.\mathsf{hat}$ denota $\{f_2\}$; y la fórmula $\mathsf{flower} \sqcap \neg\exists\mathsf{in}.\mathsf{hat}$ denota $\{f_1\}$.

En $\mathcal{EL}$ la similaridad no es una relación simétrica. Por ejemplo, f_1 es $\mathcal{EL}$-similar a f_2 en Figura 3.2 (si f_1 satisface una fórmula, entonces también la satisface f_2), pero f_2 no es $\mathcal{EL}$-similar a f_1 (f_2 satisface la fórmula $\exists\mathsf{in}.\mathsf{hat}$ y f_1 no la satisface). En $\mathcal{ALC}$ la similaridad es una relación simétrica porque el lenguaje contiene negación. En la figura, f_1 no es $\mathcal{ALC}$-similar a f_2 porque f_2 no satisface $\neg\exists\mathsf{in}.\mathsf{hat}$. $\mathcal{ALC}$ es más expresivo que $\mathcal{EL}$, esto es, para algún elemento a es posible ser $\mathcal{EL}$-similar pero no $\mathcal{ALC}$-similar a algún elemento b, pero no viceversa.

Como mostramos en la Sección 3.1, se puede demostrar que, para $\mathcal{ALC}$, los conjuntos de similaridad de un modelo finito coinciden exactamente con las *clases de simulación* del modelo. Las clases de simulación se han estudiado ampliamente en la literatura (por ejemplo, [Blackburn et al., 2001]; [Kurtonina and de Rijke, 1998]), y hay varios algoritmos eficientes para el cálculo de las clases de $\mathcal{ALC}$-simulación [Hopcroft, 1971; Paige and Tarjan, 1987; Dovier et al., 2004]. Sin embargo, estos algoritmos sólo calculan las clases de simulación para $\mathcal{ALC}$ y no obtienen ERs como sucedía para $\mathcal{EL}$ con los algoritmos que describimos en la Sección 3.2.1. Para poder generar ERs en $\mathcal{ALC}$ es necesario ampliar el algoritmo [Hopcroft, 1971] que calcule junto con cada conjunto, también una fórmula que denote exactamente este conjunto.

El pseudocódigo del algoritmo de $\mathcal{ALC}$ que genera ERs se muestra como Algoritmo 3 (con $\mathcal{L} = \mathcal{ALC}$) y Algoritmo 4. Dado un modelo de $\mathcal{M} = (\Delta, \|\cdot\|)$, el algoritmo calcula un conjunto RE de $\mathcal{ALC}$, que contiene las fórmulas tales que $\{\|\varphi\| \mid \varphi \in \mathsf{RE}\}$ es el conjunto de conjuntos $\mathcal{ALC}$-similitud de $\mathcal{M}$. El algoritmo comienza con $\mathsf{RE} = \{\top\}$ (donde $\|\top\| = \Delta$), y sucesivamente refina RE haciendo que sus elementos denoten conjuntos más y más pequeños. Eso mantiene el invariante que, al inicio y al final de cada iteración, $\{\|\varphi\| \mid \varphi \in \mathsf{RE}\}$ es siempre una partición de Δ. Los algoritmos iteran sobre todos los símbolos relacionales proposicionales en prop y rel para la construcción de nuevas fórmulas hasta que todas las fórmulas de RE denoten elementos únicos (es decir, que sólo un individuo las satisfaga), o no se ha hecho ningún cambio desde la iteración anterior. En cada iteración, se llama al procedimiento $\text{add}_{\mathcal{ALC}}(\varphi, \mathsf{RE})$, que agrega φ a toda fórmula $\psi \in \mathsf{RE}$, si $\|\psi \sqcap \varphi\| \neq \emptyset$ y $\|\psi \sqcap \neg\varphi\| \neq \emptyset$.

Entrada: Un modelo $\mathcal{M} = (\Delta, \|\cdot\|)$
Salida : Un conjunto RE de fórmulas tales que $\{\|\varphi\| \mid \varphi \in \mathsf{RE}\}$ es el conjunto de $\mathcal{L}$-similaridad de $\mathcal{M}$.

```
RE ← {⊤}
for p ∈ prop do
 | add_L(p, RE)
end
while exista alguna φ ∈ RE, |||φ|||^M > 1 do
 | for φ ∈ RE, R ∈ rel do
 |  | add_L(∃R.φ, RE)
 | end
 | if no se hicieron cambios a RE then
 |  | exit
 | end
end
```

Algoritmo 3: Computando los conjuntos de $\mathcal{L}$-similaridad.

El algoritmo $\mathcal{ALC}$ calcula los conjuntos $\mathcal{ALC}$-similaridad del modelo en tiempo $O(n^3)$, donde n es el número de individuos en el dominio. Por lo tanto su complejidad computacional es polinomial. En trabajo previo [Areces et al., 2008] se ha mostrado que este algoritmo se ejecuta más rápidamente que el algoritmo para $\mathcal{EL}$ (ambos en tiempo polinomial). El problema de $\mathcal{ALC}$ para la GER es que introducirá negaciones libremente en las distinciones de casos, esto puede hacer que las fórmulas resultantes sean difíciles de entender. Aquí también presentamos un algoritmo para los conjuntos $\mathcal{EL}$-similitud simplemente sustituyendo de la llamada a $\text{add}_{\mathcal{ALC}}$ en Algoritmo 3 por una llamada a $\text{add}_{\mathcal{EL}}$, que es definido en el Algoritmo 5. Como antes, el algoritmo mantiene un conjunto $\mathsf{RE} = \{\varphi_1, \ldots, \varphi_n\}$ de fórmulas (esta vez de $\mathcal{EL}$) tal que $\|\varphi_1\| \cup \ldots \cup \|\varphi_n\| = \Delta$, y que se refina de manera iterativa. El algoritmo $\text{add}_{\mathcal{ALC}}$ asegura que $\|\varphi_1\|, \ldots, \|\varphi_n\|$ es una partición de Δ, pero $\text{add}_{\mathcal{EL}}$ no lo asegura.

Debido a que mantiene un invariante más débil, el conjunto RE puede contener más fórmulas en el $\mathcal{EL}$-algoritmo que en el $\mathcal{ALC}$-algoritmo. Como Δ tiene un número exponencial de subconjuntos, hay un riesgo de que el $\mathcal{EL}$-algoritmo tenga peor tiempo de ejecución, como se muestra en [Areces et al., 2008].

```
for ψ ∈ RE with |||ψ||| > 1 do
 | if ||ψ ⊓ φ|| ≠ ∅ and ||ψ ⊓ ¬φ|| ≠ ∅ then
 |  | add ψ ⊓ φ and ψ ⊓ ¬φ to RE;
 |  | borrar ψ de RE;
 | end
end
```

Algoritmo 4: $\text{add}_{\mathcal{ALC}}(\varphi, \mathsf{RE})$.

for $\psi \in RE$ *con* $\|\psi\| > 1$ **do**
  if $\psi \sqcap \varphi$ *no es redundante en* RE **and** $\|\psi \sqcap \varphi\| \neq \emptyset$ **and** $\|\psi \sqcap \varphi\| \neq \|\psi\|$ **then**
    add $\psi \sqcap \varphi$ to RE
    borrar fórmulas redundantes de RE
  end
end

Algoritmo 5: $\text{add}_{\mathcal{EL}}(\varphi, \mathsf{RE})$.

Vamos a probar estos algoritmos en algunos ejemplos. En primer lugar, corremos el $\mathcal{EL}$ algoritmo en el modelo que se muestra en la Figura 3.2 (a), que es tomado de [Dale and Haddock, 1991a]. El algoritmo comienza con $\mathsf{RE} = \{\top\}$. En el primer bucle, se añaden las fórmulas floor, bowl, cup, y table, y luego se quita $\top$ porque es redundante. No todas estas fórmulas denotan elementos únicos; por ejemplo, $\|\mathsf{cup}\|$ contiene dos elementos. Así que iteramos las relaciones para refinar nuestras fórmulas. Después de la primera iteración sobre las relaciones, tenemos $\mathsf{RE} = \{\mathsf{floor}, \mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{floor}, \mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{table}, \mathsf{cup}, \mathsf{table}\}$. Notar que bowl es redundante ya que están $\mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{floor}$ y $\mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{table}$. Estas dos fórmulas tienen interpretaciones singleton, es decir que son ERs de b_1 y b_2 respectivamente. Pero aún quedan sin distinguir tables y cups.

Ahora podemos usar la división entre los bowls para distinguir las cups en la segunda iteración. El resultado de esto es $\mathsf{RE} = \{\mathsf{floor}, \mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{floor}, \mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{table}, \mathsf{cup} \sqcap \exists \mathsf{in}.(\mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{floor}), \mathsf{cup} \sqcap \exists \mathsf{in}.(\mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{table}), \mathsf{table}\}$. En este punto, todas las fórmulas excepto table denotan elementos únicos, y más iteraciones no nos permiten refinar table; por lo que el algoritmo termina. Cada fórmula con una extensión singleton es una expresión referencial; por ejemplo, $\mathsf{cup} \sqcap \exists \mathsf{in}.(\mathsf{bowl} \sqcap \exists \mathsf{on}.\mathsf{table})$ sólo es satisfecha por c_2, por lo que puede realizarse como *la taza que está en el bowl, sobre la mesa.* Notar que el algoritmo no se centró en un elemento en particular (un target); sino que simultáneamente generó ERs para todos los elementos a excepción de las dos mesas (que son $\mathcal{EL}$-similares entre sí).

El $\mathcal{EL}$-algoritmo demora más que el de $\mathcal{ALC}$ con el ejemplo de la Figura 3.2(b) vea [Stone and Webber, 1998]. A pesar de ello identifica correctamente a r_1 como *el conejo en el sombrero* y a f_2 como *la flor en el sombrero.* No es capaz de calcular una ER de f_1 porque f_1 es $\mathcal{EL}$-similar a f_2. De hecho, el algoritmo termina con RE que contiene tanto a flower como a $\mathsf{flower} \sqcap \exists \mathsf{in}.\mathsf{hat}$. Esto es un patrón típico de los casos asimétricos de similitud en $\mathcal{EL}$: si hay dos fórmulas φ_1 y φ_2 en el conjunto de salida con $\|\varphi_1\| \subset \|\varphi_2\|$, entonces hay en general, un elemento $b \in \|\varphi_2\| - \|\varphi_1\|$ tal que todos los elementos en $\|\varphi_1\|$ son similares a b, pero no viceversa. Por el contrario, en el $\mathcal{ALC}$-algoritmo se puede explotar la mayor expresividad de $\mathcal{ALC}$ para dividir flower en 2 nuevas fórmulas $\mathsf{flower} \sqcap \exists \mathsf{in}.\mathsf{hat}$ y $\mathsf{flower} \sqcap \neg\exists \mathsf{in}.\mathsf{hat}$, generando una ER única para f_1 pero que contiene negación.

3.4 Computando expresiones referenciales en $\mathcal{FO}^-$

En esta sección comenzaremos explicando trabajo previo en la GER usando $\mathcal{FO}^-$. En particular describimos el trabajo de [Krahmer et al., 2003] que se introdujo brevemente en la Sección 2.2.2. Luego mostraremos cómo la GER en $\mathcal{FO}^-$ puede verse bajo la luz de las $\mathcal{FO}^-$-simulaciones. En [Krahmer et al., 2003] introducen un algoritmo para GER basado en computación de *isomorfismos de subgrafos.* Su algoritmo toma como entrada un grafo dirigido etiquetado G y un nodo e (el target) y devuelve, si es posible, un subgrafo conectado H de G, que contiene al target e y suficiente información conectada para distinguir a e de los otros nodos.

Para mantener la terminología de [Krahmer et al., 2003], en lo que sigue hablaremos alternativamente de **grafos etiquetados** en lugar de modelos relacionales. La diferencia es que los grafos etiquetados de [Krahmer et al., 2003] codifican las propiedades proposicionales utilizando loops de relaciones binarias (por ejemplo, escriben $dog(e, e)$ en vez de $dog(e)$).

La idea principal de la implementación de [Krahmer et al., 2003] de isomorfismos de subgrafos puede resumirse intuitivamente como sigue. Dados dos grafos etiquetados H y G, y los vértices v de H y w de G, decimos que el par (v, H) *refiere* al par (w, G) si y sólo si H está conectado y H puede ser colocado "sobre" G de tal manera que: 1) v se coloca sobre w; 2) cada nodo de H se coloca sobre un nodo de G con al menos los mismos predicados unarios (pero tal vez más); y 3) cada arista de H es colocada sobre una arista con la misma etiqueta. Por otra parte, (v, H) *refiere únicamente a* (w, G) si (v, H) refiere a (w, G) y no hay vértice $w' \neq w$ en G tal que (v, H) refiere a (w', G).

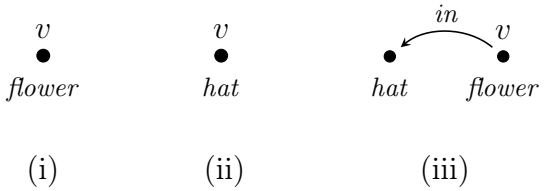


Figura 3.3: Algunos subgrafos conectados de la escena de la Figura 3.2(b), (i) puede realizarse como *flor*, (ii) como *sombrero* y (iii) *flor en un sombrero*.

Como ejemplo, consideremos el modelo relacional que se representa en la Figura 3.2(b) como un grafo etiquetado G, y consideremos los pares de nodos conectados y subgrafos (V, H) que se muestran en la Figura 3.3. Está claro que (i) se refiere a cualquier nodo $w \in \{f_1, f_2\}$; (ii) se refiere a $w \in \{h_1, h_2, h_3, h_4\}$ y en (iii) hat refiere únicamente a h_4, y flower a f_2; El algoritmo propuesto en [Krahmer et al., 2003] es una implementación directa del algoritmo de ***branch & bound*** guiado por costo que trata sistemáticamente todos los subgrafos pertinentes H de G comenzando con el subgrafo que contiene sólo el vértice target v y ampliando de forma recursiva, tratando de agregar arcos de G que son adyacentes al subgrafo H construido hasta el momento. En la terminología de [Krahmer et al., 2003] un *distractor* es un nodo de G diferente de v que también es referenciado por H. El algoritmo asegura que un subgrafo se refiere únicamente al target v cuando no hay distractores. Llegado este punto, existe una solución, pero puede haber otra solución de menor costo por lo que el proceso de búsqueda continúa hasta que se detecta la solución de menor costo. La relación entre el método basado en grafos de [Krahmer et al., 2003] y nuestra perspectiva de teoría de modelos es: en modelos relacionales finitos, el isomorfismo de subgrafos corresponde a las $\mathcal{FO}^-$-simulaciones. Dados nodos u, v de G, existe un subgrafo isomorfo a G a través de una función f, que contiene u y v, y tal que $f(u) = v$ si y sólo si $u \overset{\mathcal{FO}^-}{\rightarrow} v$. Una implementación usando $\mathcal{FO}^-$-simulaciones se muestra en el Algoritmo 6.

El Algoritmo 6 transforma el *grafo* calculado, que se refiere únicamente al target v en una $\mathcal{FO}^-$-ER *fórmula para* v. El Algoritmo 7 nos dice cómo extender H en cada paso del bucle principal. Hay que tener en cuenta que, a diferencia de la presentación de [Krahmer et al., 2003], $\texttt{makeRE}_{\mathcal{FO}^-}$ computa no sólo un grafo H, sino también una $\mathcal{L}$-simulación de f.

Entrada: $G = \langle \Delta_G, \| \cdot \| \rangle$ y $v \in \Delta_G$
Salida : $\mathcal{FO}^-$-ER para v en G si hay, o $\perp$

$H := \langle \{v\}, \emptyset \rangle$;
$f := \{v \mapsto v\}$;
$H' := \texttt{find}_{\mathcal{FO}^-}\ (v, \perp, H, f)$;
let $H' = \langle \{a_1 \ldots a_n\}, \| \cdot \| \rangle$,$v = a_1$;
$\gamma := \bigwedge_{a_i \neq a_j} (x_i \not\approx x_j) \wedge \bigwedge_{(a_i, a_j) \in \|r\|} r(x_i, x_j) \wedge \bigwedge_{a_i \in \|p\|} p(x_i)$

return $\exists x_2 \ldots \exists x_n . \gamma$;

Algoritmo 6: $\texttt{makeRE}_{\mathcal{FO}^-}(v)$.

if $best \neq \bot \wedge \texttt{cost}(best) \leq \texttt{cost}(H)$ **then**
  return $best$;
end
$distractors := \{n \mid n \in \Delta_G, n \neq v, v \overset{\mathcal{FO}^-}{\Rightarrow} n\}$;
if $distractors = \emptyset$ **then**
  return H;
end
$A := \{H{+}p(u) \mid u \in \Delta_H, u \in \|p\|_G \setminus \|p\|_H\}$;
$B := \{H{+}r(u,v) \mid u \in \Delta_H, \{(u,v),(v,u)\} \cap \|r\|_G \setminus \|r\|_H \neq \emptyset\}$;
$EXT := (A \cup B) \times \{id\}$;
foreach $\langle H', f' \rangle \in EXT$ **do**
  $I := \texttt{find}_{\mathcal{FO}^-}(v, best, H', f')$;
  if $best = \bot \vee \texttt{cost}(I) \leq \texttt{cost}(best)$ **then**
    $best := I$
  end
end
return $best$;

Algoritmo 7: $\texttt{find}_{\mathcal{FO}^-}(v, best, H, f)$.

Como nota final sobre la complejidad, la computación $\mathcal{FO}^-$-distractores parece requerir necesariamente una solución al problema de isomorfismos de subgrafos, que es NP-completo. Mientras que en las Secciones 3.2 y 3.3 presentamos algoritmos que pueden encontrar ERs en tiempo polinomial.

3.5 Notas finales y linkeo del capítulo

En este capítulo dimos definiciones básicas de modelo, interpretación, fórmula y de otros conceptos básicos de lógica y teoría de modelos. Explicamos e ilustramos la noción de similaridad de 2 elementos u y v de un lenguaje lógico $\mathcal{L}$ ($u \overset{\mathcal{L}}{\rightsquigarrow} v$ en el lenguaje $\mathcal{L}$). Cuando para toda fórmula $\varphi \in \mathcal{L}$, tenemos que $\{u, v\} \subseteq \|\varphi\|$, se dice que u y v son indistinguibles en el lenguaje lógico $\mathcal{L}$. Como la similaridad está definida para infinitas fórmulas, también vimos un teorema que cuando el modelo es finito acota esa búsqueda infinita a una finita, teniendo en cuenta propiedades de las lógicas usadas. Explicamos algoritmos eficientes que computan ERs usando simulaciones en $\mathcal{FO}^-$, $\mathcal{ALC}$ y $\mathcal{EL}$. También se mostró una clasificación de los algoritmos vistos según la complejidad computacional de los mismos.

En el Capítulo 4 extenderemos estos algoritmos integrando una distribución finita de probabilidades que representa la incertidumbre presente en posibles aplicaciones de la generación de expresiones referenciales que deben referirse al mundo real y tienen que lidiar con datos ruidosos de sensores y modelos incompletos de la realidad. El objetivo es generar un ranking de las expresiones referenciales ordenado por la probabilidad de ser correctamente interpretada en el contexto. En el Capítulo 5 evaluamos estos algoritmos usando técnicas automáticas y evaluaciones de jueces humanos sobre datos de benchmarks del área. En el Capítulo 6 se realiza una evaluación con respecto a un nuevo corpus de expresiones referenciales de puntos de interés en mapas de ciudades con distintos niveles de zoom.

Capítulo 4

Modelando la incertidumbre

Las posibles aplicaciones de la generación de expresiones referenciales que deben referirse al mundo real sufren de **incertidumbre** por datos ruidosos de sensores y modelos incompletos de la realidad como discutimos en el Capítulo 1. En este capítulo se presentan nuevos algoritmos basados en teoría de modelos y $\mathcal{L}$-simulaciones como los descriptos en el Capítulo 3. Pero que a diferencia de estos usan una distribución finita de probabilidades que representa esta incertidumbre. Estos algoritmos generan no **una** ER, sino un **ranking** de ERs. Este ranking de expresiones referenciales está ordenado por la probabilidad de ser correctamente interpretada en el contexto. Conseguimos así tres características nuevas de estos algoritmos:

- **No determinismo**: dos ejecuciones del algoritmo con la misma entrada podrían resultar en diferentes ERs para los objetos en la escena. Podemos generar entonces como salida un ranking de las ERs generadas por el algoritmo si lo ejecutamos muchas veces con el mismo input.

- Conseguimos mayor control sobre la **sobreespecificación** que genera el algoritmo, simulando el tipo y la cantidad de sobreespecificación que encontramos en corpora.

- Dado un corpus de ERs para un dominio determinado, es posible usar **aprendizaje automático** para aproximar una distribución de probabilidad adecuada para las propiedades y relaciones de una escena no vista antes. Esto hace que el ranking de las ERs generadas para el modelo simule el que se observa en el corpus usado para el aprendizaje automático.

Este capítulo está dividido en cinco secciones. En la primer sección explicamos la entrada y salida del algoritmo. En la Sección 4.2 mostramos como obtener la distribución de probabilidad finita que toma como entrada el algoritmo, usando aprendizaje automático cuando hay corpus disponible para el dominio. Luego en la Sección 4.3 mostramos el algoritmo en detalle y definimos los conceptos y técnicas de teoría de modelos necesarios para entenderlo. En la Sección 4.4 mostramos un ejemplo de ejecución paso a paso. Explicamos cómo conseguimos asegurar terminación, cómo se generan las ERs no-determinísticamente y cómo conseguimos mayor control sobre la sobreespecificación. Finalmente en la Sección 4.5 damos un resumen y comentamos cómo se linkea el capítulo con el resto de la tesis. Parte de los resultados mostrados en este capítulo fueron publicados en [Altamirano et al., 2012].

4.1 Entrada y salida del algoritmo probabilístico

Como todo algoritmo de GER, nuestros algoritmos toman como entrada un contexto representado por un **modelo** $\mathcal{M}$. Por ejemplo en la Figura 4.1 a la derecha se muestra el modelo de

la escena de la izquierda. Sin embargo a diferencia de otros algoritmos, nuestros algoritmos no toman como entrada el **target**. Nuestros algoritmos generan rankings de ERs para todos los elementos del modelo paralelamente. En particular, el algoritmo generará ERs para el conjunto singleton $\{e_5\}$. En la Sección 4.4.3 de este capítulo se muestra una sencilla extensión de los algoritmos que permite generar ERs plurales.

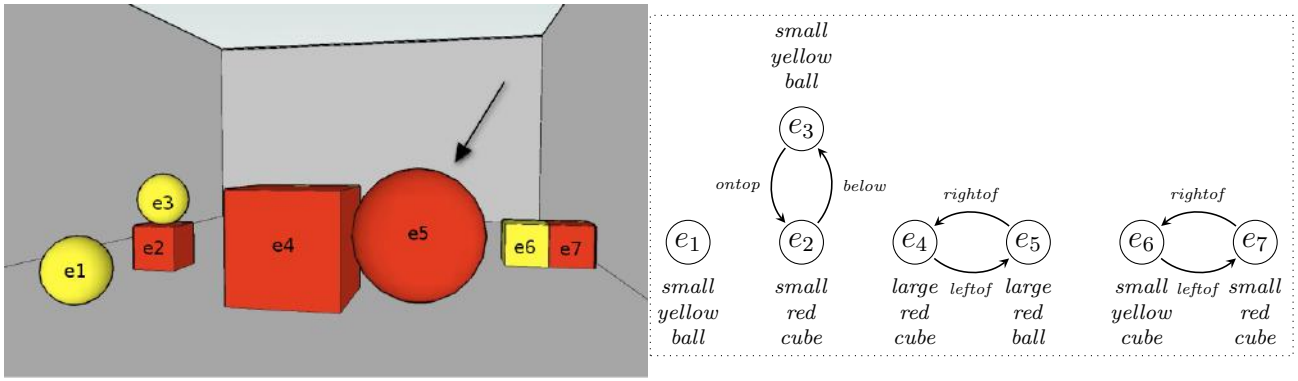


Figura 4.1: Modelo que representa a la figura.

El sistema también toma como entrada una **distribución de probabilidad finita** de las propiedades y relaciones de la signatura del modelo[1]. Una distribución de probabilidades Rs, es una lista de pares (R, R.p_{use}) que vincula a cada relación R a una cierta probabilidad de uso R.p_{use} ordenada de mayor a menor por p_{use}. La distribución de probabilidad para el modelo del ejemplo se ilustra en la Tabla 4.1.

R	*ball*	*cube*	*red*	*large*	*ontop*	*yellow*	*small*	*rightof*	*leftof*	*below*
R.p_{use}	1.0	1.0	0.978	0.257	0.178	0.15	0.107	0.007	0	0

Tabla 4.1: Distribución de probabilidad de las propiedades y relaciones de la Figura 4.1.

Sea REL el conjunto de todos los símbolos de relación en el modelo (es decir, la *signatura* del modelo), entonces podemos decir que Rs $\in (\mathsf{REL} \times [0,1])^*$. En la próxima sección explicamos como obtener estas probabilidades que el algoritmo toma como input. Los algoritmos descriptos en el Capítulo 2 y en el Capítulo 3 procesan las relaciones unarias y las relaciones binarias de forma diferente. Además, en general prefieren usar primero las relaciones unarias y sólo recurrir a las binarias en caso de que sea necesario. Como se ha observado previamente en trabajo empírico [Viethen, 2011], hay escenas en las cuales las personas usan relaciones binarias antes que unarias y aunque no sea necesario usarlas. Por lo tanto para uniformizar el tratamiento de todas las relaciones sin importar su aridad nuestro algoritmo modifica el modelo $\mathcal{M}$ antes de comenzar transformando todas las relaciones a binarias. Esto se hace agregando un elemento extra *dummy* al modelo que no representa ningún objeto de la escena y que se relaciona con todos los objetos que tenían relaciones unarias. Por ejemplo, en la Figura 4.1 se codifica el hecho de que e_1 es amarillo diciendo que está relacionado con el elemento *dummy* por la relación binaria *yellow*.

Como explicamos en el Capítulo 3, la salida el algoritmo es lo que se llaman las $\mathcal{L}$-clases (clases de la lógica $\mathcal{L}$) de similaridad del modelo de entrada $\mathcal{M}$. Intuitivamente, si dos elementos en el modelo pertenecen a la misma $\mathcal{L}$-clase de semejanza, entonces el lenguaje $\mathcal{L}$ no es lo suficientemente expresivo para diferenciarlos (es decir, no hay una fórmula en $\mathcal{L}$ que pueda distinguirlos). Cada $\mathcal{L}$-clase va acompañada de un conjunto de fórmulas cuyas interpretaciones en el modelo de input coinciden con la $\mathcal{L}$-clase. Estas fórmulas son las expresiones referenciales que referencian a los elementos de la $\mathcal{L}$-clase. Si las $\mathcal{L}$-clases son conjuntos singleton, el algoritmo llegó a un punto fijo y termina. En otra etapa de postprocesamiento se identifica la fórmula cuya

[1]En lo que sigue nos referiremos a las propiedades y relaciones de un modelo simplemente como relaciones, distinguiendo entre relaciones unarias (propiedades) o binarias (relaciones) cuando sea necesario.

extensión es igual al target como una ER del target. Como el algoritmo es no-determinístico, es decir genera diferentes ERs en distintas ejecuciones, el ranking que ERs de la salida se genera ejecutando el algoritmo un número N de veces. Cada ER generada tiene asociada una probabilidad calculada como una combinación de los p_{use} de las relaciones que contiene la ER. El ranking de ERs se genera ordenando las ERs de acuerdo a la frecuencia con la cual el algoritmo las generó, la cual se correlaciona con la probabilidad antes mencionada.

4.2 Probabilidades de uso

En la sección anterior dijimos que el algoritmo asume que para cada relación R del modelo se tiene una probabilidad de uso R.p_{use}. Como dijimos, nuestro algoritmo es no-determinista: en 2 ejecuciones podría dar ERs diferentes para el mismo target y la misma lista de probabilidades de uso. Las probabilidades de uso son el motor que guía este no-determinismo, como explicaremos en la Sección 4.3. En esta sección describimos cómo obtener la distribución finita de probabilidades de las relaciones de la signatura del modelo de entrada. Primero explicamos cómo calcularlas en el caso de tener corpus disponible para la escena. Luego generalizamos nuestro enfoque al caso de tener que generar ERs para una escena no vista antes usando técnicas de aprendizaje automático.

4.2.1 Calculando p_{use} cuando hay disponible un corpus para la escena

Supongamos que queremos generar automáticamente una ER para el target t en una determinada escena, y que tenemos disponible un corpus C de ERs de t en esa escena.

Antes de dar la metodología para calcular las probabilidades de uso, explicamos qué hacer con un corpus de ERs para tener unificado el vocabulario, y poder así definir el conjunto de relaciones a usar en las ERs. A continuación describimos los pasos:

1. **Tokenizar** las expresiones referenciales y llamar al conjunto de palabras distintas Pal.

2. **Eliminar hiperónimos** de Pal. Por ejemplo, si ambos *cubo* y *cosa* aparecen en Pal para nombrar lo mismo, eliminar *cosa*, ya que *cubo* es más específico.

3. **Normalizar sinónimos** si el conjunto de palabras obtenidas en los pasos anteriores contiene sinónimos hay que normalizarlos con un representante de la clase. Por ejemplo, las palabras *chico* y *pequeño* son ambas representadas por la semántica *small*.

4. **Llamar REL** al conjunto resultante, de la etapa anterior; que será la signatura del modelo $\mathcal{M}$ que utilizará el algoritmo.

5. **Definir** $\mathcal{M}$ para cada escena, tal que la interpretación $\|\cdot\|$ asegure de que todas las ERs encontradas en el corpus sean ERs en el modelo. Por ejemplo, las ERs de la Figura 4.2 deben denotar el target señalado en la figura, siendo el modelo $\mathcal{M}$ considerado el representado en la parte derecha de la Figura 4.2.

6. **Calcular R.p_{use}** para cada R $\in$ REL utilizando la siguiente fórmula: si hay muchas ERs para cada escena (como es el caso del corpus GRE3D7) $R.\mathsf{p}_{use}$ $= \#ERs$ que tienen R $/\#ERs$ en el corpus o asignamos 1 a R.p_{use} si R está en la ER, asignamos 0 en caso contrario (como es el caso del corpus TUNA).

En el caso del corpus GRE3D7 el vocabulario unificado es REL $= \{ball, cube, large, small, green, red, yellow, blue, right, left, top, center, rightof, leftof, ontop, below\}$.

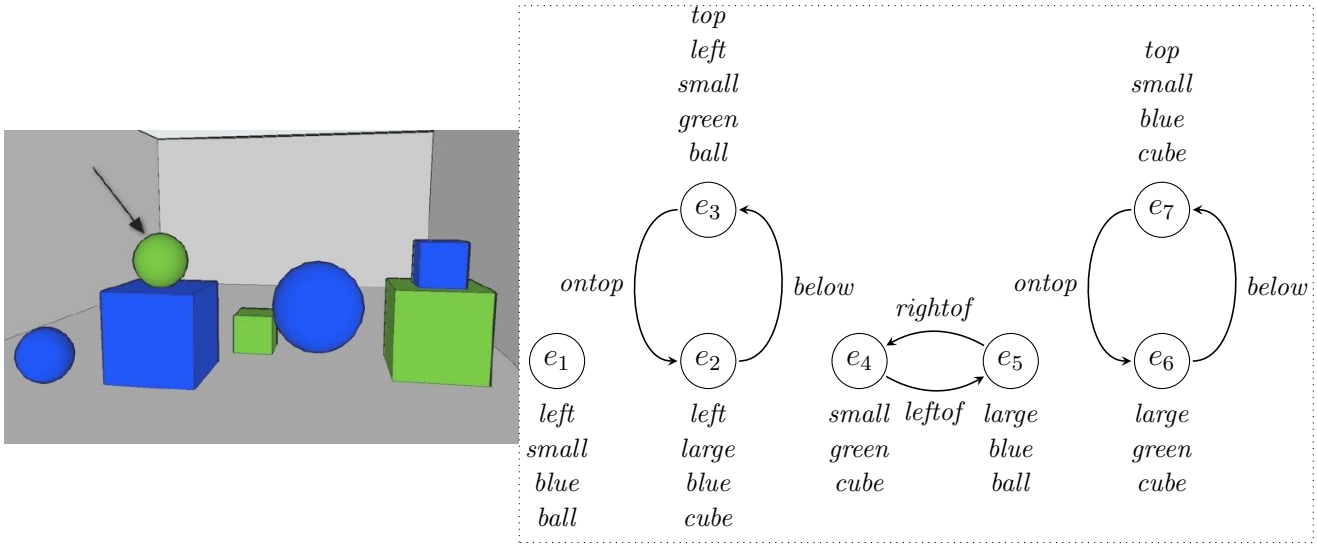


Figura 4.2: Contexto y modelo.

Utilizamos las ERs del corpus C para definir el modelo relacional utilizado por el algoritmo. Entonces definimos el valor de p_{use} para cada una de las relaciones en el modelo como el porcentaje en que aparece la relación en las ERs. Es decir,

$$R.\mathsf{p}_{use} = \frac{\#\text{de ERs en } C \text{ en las que aparece R}}{\#\text{de las ERs en } C} \tag{4.1}$$

Por ejemplo *ball* del ejemplo de la Figura 4.2, aparece en todas las ERs del corpus es decir 140, entonces *ball*.p_{use}=1 ya que 140/140=1, en cambio *green* aparece en 137 ERs, entonces *green*.p_{use}=0.98 ya que 137/140=0.98.

Esta estimación es simplista y, por ejemplo, no diferencia las propiedades del target y las propiedades de los landmarks utilizadas en una ER relacional para completar la descripción del target. Pero es intuitiva y no requiere parsear las ERs para distinguir entre target y landmark. Como vamos a ver en el Capítulo 5 produce ERs naturales que coinciden con las encontradas en corpora.

Veamos ahora el siguiente ejemplo, el contexto y el modelo se muestran en la Figura 4.2.

Expresiones Referenciales	Cantidad de ocurrencias	Porcentaje
green ball	91	65.00%
small green ball	23	16.43%
small green ball on-top red large cube	8	5.71%
green ball on-top blue cube	5	3.57%
green ball on-top large blue cube	5	3.57%
small green ball on-top blue cube	2	1.43%
ball on-top cube	1	0.71%
small green ball on-top red large left cube	1	0.71%
small ball on-top large cube	1	0.71%
top green ball	1	0.71%
small ball on-top cube	1	0.71%
green ball on-top cube	1	0.71%

Tabla 4.2: Semántica de las expresiones referenciales producidas por las personas para la Figura 4.2.

Las ERs, cantidad de ocurrencias y porcentaje del corpus para de la Figura 4.2 se muestran en la Tabla 4.2. La signatura resultante y su p_{use} asociado se muestran en la Tabla 4.3.

El p_{use} de las relaciones taxonómicas fue reemplazado por 1 ignorando el p_{use} conseguido por el método para evitar que el algoritmo genere ERs como *la esfera que está sobre algo*.

Observe que los valores R.p_{use} obtenidos de esta manera deben ser interpretados como la probabilidad de utilizar R para describir al target en el modelo $\mathcal{M}$, y podríamos argumentar que se correlacionan con la saliencia de R para el target y su modelo.

Estas probabilidades no serán útiles para describir los diferentes targets en diferentes escenas porque son dependientes de la escena para la que se calcularon. Veremos cómo se pueden obtener valores para otros targets y escenas utilizando un enfoque de aprendizaje automático en la siguiente sección.

ball	*cube*	*green*	*small*	*ontop*	*blue*	*large*	*left*	*top*	*right*	*leftof*	*rightof*	*below*
1.0	1.0	0.978	0.257	0.178	0.125	0.107	0.007	0.007	0	0	0	0

Tabla 4.3: Probabilidades de uso de las palabras del corpus GRE3D7 obtenidas usando la Fórmula 4.1 para la Figura 4.2.

4.2.2 Calculando p_{use} cuando no hay corpus para la escena

Si no hay corpus de expresiones referenciales para la escena, se puede estimar las p_{use} a partir de corpus de diferentes escenas en el mismo dominio.

En lo que sigue introduciremos regresión lineal que es la técnica de aprendizaje automático que usamos para estimar las probabilidades de uso p_{use} de las palabras que le damos al algoritmo, y luego damos un ejemplo de cómo calculamos las probabilidades de uso para una figura particular del GRE3D7.

La **regresión lineal** es un modelo matemático usado para aproximar la relación de dependencia entre una variable dependiente Y, las variables independientes X_i y un término aleatorio ε. Este modelo puede ser expresado como:

$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$ donde:

- Y: variable dependiente, explicada o regresando.
- X_1, X_2, $\cdots$, X_p : variables explicativas, independientes o regresores.
- β_0, β_1, β_2, $\cdots$, β_p : son parámetros, miden la influencia que las variables explicativas tienen sobre Y. Donde β_0 es la intersección o término *constante*, las β_i (i > 0) son los parámetros respectivos a cada variable independiente, y p es el número de parámetros independientes a tener en cuenta en la regresión.

Tomamos la parte azul y verde del corpus GRE3D7 mostrado en el Apéndice B.1, para aprender las probabilidades de uso de cada palabra de las ERs que dieron las personas, y así estimar cómo es la distribución de las probabilidades de uso de las palabras del conjunto REL en el corpus.

Para la tarea de aprendizaje automático definimos un conjunto de características de las ERs. Estas características son las variables X_i que le damos al método de regresión lineal para calcular la dependencia o independencia de la probabilidad de uso p_{use} de cada palabra con respecto a ellas.

Las características que seleccionamos son:

- target-tiene es 1 cuando el elemento target tiene la propiedad.

- #rel-prop es el número de propiedades y relaciones que el target tiene.
- #rel es el número de relaciones binarias que el target tiene.
- landmark-tiene es 1 cuando un landmark del target tiene la propiedad. Un objeto es un landmark si tiene una relación directa en el modelo, con el target.
- location-has es 1 cuando la ER puede usar la ubicación del target en la figura[2].
- disc es 1 sobre el número de objetos en el modelo que tienen la propiedad.
- adj-target-tiene es el número de adjetivos que la ER contiene.

El procedimiento entonces fue el siguiente: creamos un archivo por cada palabra del vocabulario REL. En cada línea del archivo se escribió la lista de propiedades nombradas anteriormente, por cada ER de la imagen considerada. Cada archivo fue analizado con la función de regresión lineal de WEKA [Hall et al., 2009], que tiene una colección de herramientas para aprendizaje automático.

En la Tabla 4.4 se muestra la salida del algoritmo de regresión lineal para las 16 escenas verdes y azules del corpus GRE3D7 y sus respectivas ERs. En la primer columna podemos ver la palabra, en la segunda columna está el error promedio dado por regresión lineal, en la tercera columna el error medio y en la cuarta la fórmula que la regresión lineal calculó para la probabilidad de uso de la palabra considerada.

Palabra	Error promedio LR	Error-PM	Fórmula de LR
ball	0.0465	0.0609	0.2894 * disc + 0.7883
cube	0.0417	0.0531	0.49 * disc - 0.0129
blue	0.0353	0.0454	0.848 * target-tiene + 0.1073
green	0.0264	0.046	0.8722 * target-tiene + 0.0016
large	0.1762	0.2378	0.5911 * target-tiene + 0.0354
small	0.1499	0.1755	0.3918 * target-tiene + 0.2478 * landmark-tiene - 0.0913
leftof	0.0041	0.0094	0.0131 * target-tiene + 0.0253 * adj-target-tiene - 0.0507
ontop	0.0706	0.1594	0.2942 * target-tiene
rightof	0.0029	0.0049	0.0153 * target-tiene + 0.001
left	0.0068	0.0101	0.0346 * adj-target-tiene - 0.0653
right	0.0079	0.0092	- 0.0118 * disc + 0.0141
top	0.0099	0.0135	0.0069
center	0.0023	0.0037	0.0047 * target-tiene + 0.0047 * adj-target-tiene + 0.0029 * landmark-tiene - 0.009

Tabla 4.4: Fórmulas y errores de regresión lineal para la parte del corpus GRE3D7 que se muestra en la figura del Apéndice B.1.

Las palabras *red* y *yellow* no aparecen en la Tabla 4.4 porque no aparecían en las ERs del corpus de las imágenes consideradas (sólo contenían colores verde y azul).

Podemos ver por ejemplo que *ball* tiene una p_{use} alta, es natural ya que en el corpus GRE3D7 todos los targets son *ball*. En cambio se puede ver que en *cube* no es tan alto y la p_{use} depende

[2]Para el TUNA corpus ya que tiene algunas ERs donde se le dijo a las personas que podían usar la localización del objeto y otras que no.

más de la discernibilidad (disc). La p_{use} de *cube* representa en este dominio la probabilidad de usar *cube* en la descripción del landmark.

En los casos de *blue* y *green* el valor de p_{use} depende de si el target es *blue* o *green*, entonces si el target es *blue*, le da un valor alto a *blue* y bajo a *green* y viceversa, no depende del valor de discernibilidad de la relación en el modelo.

En la tabla también podemos ver que relaciones como *small* y *large* tienen un error mucho más alto que el resto de las relaciones, esto probablemente se debe a que ellas son propiedades vagas. Este tipo de propiedades no son absolutas y dependen del contexto considerado. Una característica interesante que vemos y que no fue mencionada en trabajo previo es que tamaño es más frecuente usado para sobreespecificación cuando el target y el landmark tienen el mismo tamaño (es usado en ERs sobreespecificadas el 49% cuando el target y el landmark comparten el tamaño, y sólo el 25% cuando no lo comparten). Esto es aprendido por el modelo de regresión lineal, también vemos que *small* le da un peso considerable a la característica landmark-tiene.

Las relaciones *ontop*, *leftof* y *rightof* tienen una p_{use} baja. *Ontop* es más probable que *leftof* o *rightof*, esta observación fue reportada en [Viethen et al., 2011].

Para hacer el aprendizaje automático, separamos 1 contexto al que llamamos testing, y todos los demás los usamos de entrenamiento. Por ejemplo, para aprender las p_{use} del contexto 13 mostrado en la figura del Apéndice B.1, se usaron los otros 15 contextos. Nuestro conjunto de características es intencionadamente simplista con el fin de que sea independiente de dominio. Como resultado hay algunas relaciones complejas entre las características de las escenas que no es capaz de capturar. La característica más importante del dominio GRE3D7 que no somos capaces de aprender, y tiene un impacto en nuestro desempeño, es que las propiedades de tamaño (es decir, small y large) se utilizan mucho más cuando el target no puede ser identificado sólo con las propiedades taxonómicas absolutas (green y blue) y (ball y cube). En otras palabras, en el corpus GRE3D7 se utiliza el tamaño con más frecuencia (90,2%) cuando la ER resultante no es sobreespecificada que cuando sí es sobreespecificada (34%). Como resultado podemos ver en la Tabla 4.4, el valor estimado para *large* tiene un error considerable. En el caso del TUNA-corpus mostramos que no podemos aprender la dependencia de la dimensión-x y dimensión-y, es decir, cuando una persona añade dimensión-x es altamente probable que incluya la dimensión-y en su expresión referencial.

ball	*cube*	*green*	*small*	*ontop*	*blue*	*large*	*left*	*top*	*right*	*leftof*	*rightof*	*below*
1.0	1.0	0.993	0.346	0.179	0.124	0.03	0.002	0	0.001	0	0	0

Tabla 4.5: Probabilidades de uso aprendidas usando regresión lineal para la Figura 4.2.

Utilizamos regresión lineal para aprender la función de p_{use} para cada palabra en la signatura. Para una escena determinada, reemplazamos las variables de la función obtenida por los valores de las características de la escena que queremos describir. Por ejemplo, para la escena de la Figura 4.2 las p_{use} aprendidas usando regresión lineal se muestran en la Tabla 4.5.

En el Capítulo 5 mostramos cómo los métodos para estimar las probabilidades de uso se aplican también al corpus TUNA para generar ERs.

En la próxima sección vemos como agregar esta distribución finita de probabilidades a los algoritmos presentados en el Capítulo 3.

4.3 El algoritmo probabilístico

Los algoritmos presentados en esta sección extienden los Algoritmos 3 y 5 del Capítulo 3 con una distribución finita de probabilidades. Al igual que el Algoritmo 3, el Algoritmo 8 es paramétrico en el lenguaje lógico de las ERs generadas. El Algoritmo 9 como el Algoritmo 5 usa el lenguaje lógico $\mathcal{EL}$. De forma similar, se podrían extender algoritmos para los otros lenguajes lógicos presentados en el Capítulo 3.

Antes de empezar explicando el algoritmo, damos algunos conceptos de teoría de modelos necesarios para entenderlo.

Para esta tesis una **fórmula** corresponde a una descripción de uno o más elementos del contexto y puede ser o no una ER, dependiendo de si su interpretación coincide o no con el target. Por ejemplo la interpretación de la fórmula `ball` son todas las esferas del contexto. En el contexto de la Figura 4.1 serían: e_1, e_3 y e_5. Así como $\|\texttt{ball} \sqcap \texttt{yellow}\|$ es: e_1 y e_3. Como dijimos anteriormente la **interpretación de una fórmula** es el conjunto de elementos que satisfacen la fórmula. A la interpretación de una fórmula también la llamamos **clase**.

Decimos que una fórmula $\exists R.\varphi$ es **informativa** con respecto a otra fórmula γ cuando la interpretación de la conjunción de las dos fórmulas $\|\exists R.\varphi \sqcap \gamma\|$ contiene menos elementos que la interpretación de γ. Por ejemplo si tenemos la fórmula `ball` y queremos saber si `red` es informativa con respecto a `ball` para el modelo de la Figura 4.1 vemos que sí lo es, ya que e_5 es una esfera roja y existen otras esferas que no son rojas, es decir `red` divide el conjunto de esferas, por lo tanto es informativa. Si un algoritmo permite agregar fórmulas no informativas a las expresiones referenciales, entonces podrá generar ERs sobreespecificadas.

Una fórmula φ es **redundante** por el conjunto de expresiones referenciales si ya tenemos otras fórmulas en RE cuya unión es igual a la interpretación de φ. Por ejemplo, para el modelo de la Figura 4.1, $\top$ que es la fórmula a la cual todos los objetos pertenecen, será redundante cuando se agreguen las fórmulas `ball` y `cube` ya que todos los elementos de la figura son esferas o cubos. Es decir ya tenemos descripciones más precisas de los objetos que $\top$. Formalmente $\|\top\| = \{e_1,e_2,e_3,e_4,e_5,e_6,e_7\}$ $\|\texttt{ball}\|=\{e_1,e_3,e_5\}$ y $\|\texttt{cube}\|=\{e_2,e_4,e_6,e_7\}$. La unión de los conjuntos $\|\texttt{ball}\|$ y $\|\texttt{cube}\|$ dan exactamente $\|\top\|$.

Trivial es una fórmula para la cual no hay objetos que la satisfagan en el modelo considerado, por ejemplo en el contexto de la Figura 4.1, `large yellow ball` es trivial ya que no hay esferas amarillas grandes en el contexto considerado. En el Algoritmo 9 vamos a utilizar fórmulas del lenguaje de descripción $\mathcal{EL}$ [Baader et al., 2003] para describir clases de refinamiento. Note, sin embargo, que el lenguaje formal utilizado en particular es independiente del algoritmo principal, y diferentes funciones $\text{add}_{\mathcal{L}}$(R,φ, RE) se pueden utilizar en función del lenguaje en cuestión. La interpretación de la fórmula de $\mathcal{EL}$ $\|\psi \sqcap \exists \text{R}.\varphi\|$ es el conjunto de todos los elementos que satisfagan ψ y que están relacionados por relación R con algún elemento que satisface φ. Por ejemplo, la interpretación de la fórmula $\texttt{ball} \sqcap \exists\texttt{leftof.cube}$ es el conjunto de todas las esferas que están a la izquierda de algún cubo.

El conjunto RE contendrá la descripción formal de las clases de refinamiento y es inicializado por la descripción más general $\top$. Al finalizar contendrá las fórmulas que representan las ERs de los elementos del modelo.

Los Algoritmos 8 y 9 realizan los siguientes pasos. Primero, recorre la lista Rs, para cada R (relaciones de la signatura del dominio REL). Luego calcula R.rnd_{use}, un número aleatorio en [0,1], y R.inc_{use} que será (1 - R.p_{use}) / MaxIterations, siendo MaxIterations el número máximo de iteraciones del ciclo principal que queremos permitir. Si $\text{R.rnd}_{use} \leq \text{R.p}_{use}$ entonces vamos a utilizar R para refinar el conjunto de clases, si no la propiedad R será ignorada en esta iteración. El valor de R.p_{use} se incrementará en R.inc_{use} en cada ciclo principal, para asegurar que todas las relaciones son, en algún momento, consideradas por el algoritmo. Esto asegura que una expresión referencial se encontrará si existe; pero dará mayor probabilidad a las expresiones

que usan las relaciones con R.p_{use} más alta.

Mientras que RE contiene descripciones (fórmulas φ) que pueden ser refinadas, (es decir, clases con al menos dos elementos) vamos a llamar a la función de refinamiento add$_{\mathcal{EL}}$(R,φ,RE) sucesivamente con cada relación de Rs. Un cambio en una de las clases, puede desencadenar cambios en las otras. Por esa razón, si RE cambia, salimos del ciclo for y volvemos a empezar con las relaciones de R.p_{use} más altas.

Se refinará cada una de las descripciones en RE utilizando la relación R y las otras descripciones que ya están en RE, bajo ciertas condiciones que se detallan a continuación: La nueva descripción debe ser **no redundante** (la nueva clase no se puede obtener como la unión de clases ya representadas en RE), **no trivial** (la nueva clase no es vacía), es **informativa** (la nueva clase no debe coincidir con la clase original). Si se cumplen todas estas condiciones, la nueva descripción se añade a RE, y las descripciones redundantes posiblemente creadas por la adición de la nueva descripción son eliminadas. Con respecto a la informatividad, se permitirá el agregado de fórmulas no informativas en la primera iteración, antes de incrementar las R.p_{use}.

Entrada: Un modelo $\mathcal{M}$ y una lista Rs $\in$ (REL $\times$ $[0,1]$)* de relaciones con sus valores de p_{use} ordenados por p_{use}
Salida : Un conjunto de fórmulas RE tal que $\{\|\varphi\| \mid \varphi \in \mathsf{RE}\}$ es el conjunto de clases de $\mathcal{L}$-similaridad de $\mathcal{M}$

```
RE ← {⊤}                          // descripción más general ⊤ aplica a todos los elementos
for (R,R.p_use) ∈ Rs do
    R.rnd_use = Random(0,1)        // R.rnd_use es la probabilidad de usar R
    R.inc_use = (1 − R.p_use) / MaxIterations   // R.p_use incrementadas por R.inc_use
end
repeat
    FirstLoop ← TRUE
    while ∃(φ ∈ RE).(#‖φ‖ > 1) do          // clases con más de 2 elementos?
        RE' ← RE                   // hacer una copia para futura comparación
        for (R, R.p_use) ∈ Rs do
            if R.rnd_use ≤ R.p_use then          // R será usada en la expresión
                for φ ∈ RE do add_EL(R, φ, RE)   // refine todas las clases usando R
            end

            if RE ≠ RE' then                          // la clasificación cambió
                exit                                  // salga del ciclo for
            end
        end
        if RE = RE' then                  // la clasificación se ha estabilizado
            exit                          // salga del ciclo while para incrementar R.p_use
        end
    end
    FirstLoop ← FALSE
    for (R,R.p_use) ∈ Rs do
        R.p_use ← R.p_use + R.inc_use                          // incrementar R.p_use
    end
until ∀((R,R.p_use) ∈ Rs).(R.p_use ≥ 1)   // R.p_use incrementadas hasta que alcanzan 1
```

Algoritmo 8: Computando clases de $\mathcal{L}$-similaridad

Entrada: R, φ, RE
Salida : RE
if FirstLoop **then** // primera iteración?
  Informativa ← TRUE // permitir sobreespecificación
end
else Informativa ← $\|\psi \sqcap \exists R.\varphi\| \neq \|\psi\|$
for $\psi \in$ RE con $\#\|\psi\| > 1$ **do**
  if $\psi \sqcap \exists$R.φ no es redundante en RE
    and $\|\psi \sqcap \exists R.\varphi\| \neq \emptyset$ // es no-trivial: tiene elementos?
    and *Informativa* **then**
      Agregar $\psi \sqcap \exists$R.φ a RE // agregar la nueva clase a la clasificación
      borrar fórmulas redundantes de RE // borrar clases redundantes
  end
end

Algoritmo 9: add$_{\mathcal{EL}}$(R, φ, RE)

4.4 Ejemplo de ejecución

En esta sección vamos a mostrar un ejemplo de ejecución para el modelo de la Figura 4.3, considerando las probabilidades de uso mostradas en la Tabla 4.6, en la tabla también se muestran las probabilidades random del algoritmo y los incrementos de las probabilidades de uso, teniendo como número de máxima iteración 5. Mostraremos la ejecución del algoritmo sin dar un target específico. En este caso el algoritmo obtiene ERs para todos los elementos del modelo, si puede. Al finalizar entonces esperamos tener una ER para cada elemento del modelo. Recordemos que para conseguir no-determinismo, el algoritmo tiene un componente random que hace que se considere agregar una relación sólo si la probabilidad de uso es mayor que el número random calculado en esa ejecución. En el comienzo RE $=\{\top\}$ y $\|\top\| = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$.

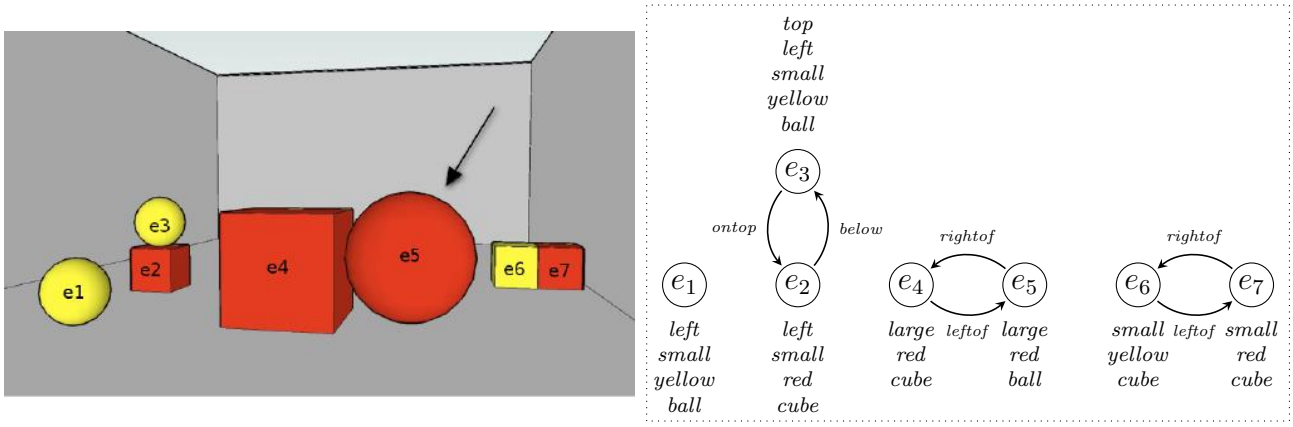


Figura 4.3: Contexto y modelo del ejemplo de ejecución.

Comenzamos a ejecutar el algoritmo, ya que estamos en la primera iteración del algoritmo permitiremos sobreespecificar. La primer relación a considerar por su R.p$_{use}$ es `ball`. `ball`.p$_{use}$ es mayor que `ball`.rnd$_{use}$, recordemos que la relación `ball` se añadirá a RE si su interpretación tiene al menos un elemento, $\|\top \sqcap$ `ball`$\|$[3] no tiene que ser vacío, y su interpretación tiene que ser distinta de $\|\top\|$. Las tres condiciones se cumplen (aunque informatividad no es necesaria ya que estamos en la primer iteración) entonces se agrega la fórmula y RE es $\{\top$, `ball`$\}$. Se ven

[3]Como las relaciones unarias se transformaron en binarias, la fórmula que el algoritmo agregará será $\top \sqcap \exists$ `ball`.$\top$. Aquí escribimos $\top \sqcap$ `ball` por simplicidad.

R	*ball*	*cube*	*red*	*large*	*ontop*	*yellow*	*small*	*rightof*	*leftof*	*top*	*left*	*below*
R.p_{use}	1.0	1.0	0.97	0.25	0.18	0.15	0.11	0.007	0	0	0	0
R.rnd_{use}	0.323	0.560	0.877	0.125	0.142	0.039	0.709	0.218	0.857	0.816	0.202	0.13
R.inc_{use}	0	0	0.004	0.148	0.164	0.17	0.178	0.198	0.2	0.2	0.2	0.2

Tabla 4.6: Distribución de probabilidad de las propiedades y relaciones de la Figura 4.3. Ejemplo de probabilidades random del algoritmo. Incremento para cada relación dadas las probabilidades de uso y las probabilidades random anteriores con *MaxIterations* 5.

los elementos de $\|\texttt{ball}\|$ enmarcados en la Figura 4.4 (e_1, e_3, e_5). El conjunto $\|\top\|$ también está enmarcado e incluye todos los elementos del modelo.

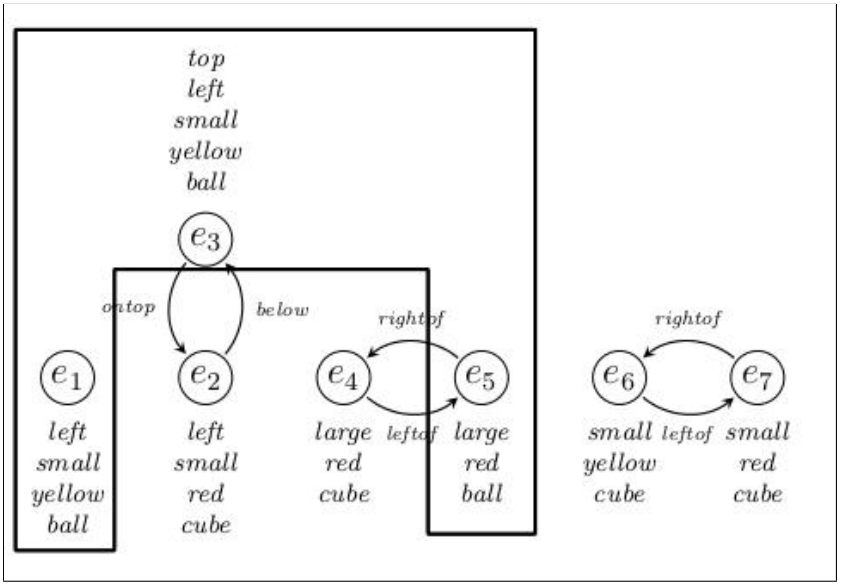


Figura 4.4: El cuadro mayor indica elementos que satisfacen $\top$. El recuadro más chico elementos que satisfacen la fórmula `ball`.

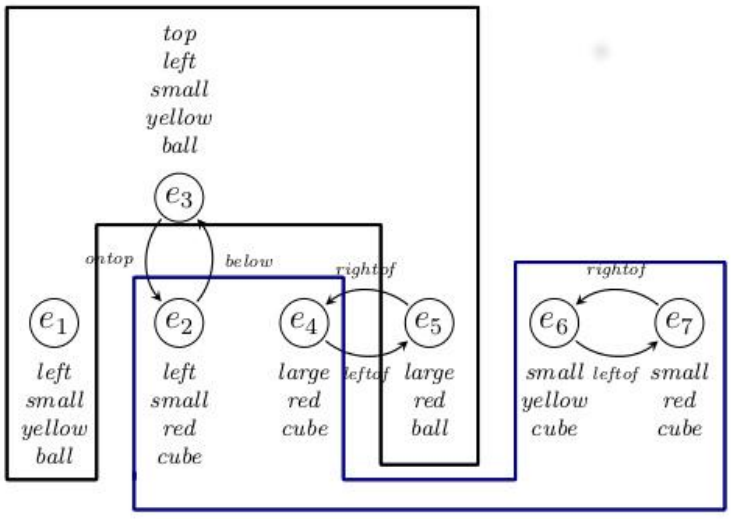


Figura 4.5: Conjuntos de elementos que satisfacen las fórmulas `ball` y `cube`.

La siguiente propiedad a considerar es `cube`, también supera la probabilidad $\mathsf{cube.rnd}_{use}$. Realizando el mismo procedimiento antes mencionado, queda $\mathsf{RE} = \{\top, \texttt{ball}, \texttt{cube}\}$ donde

$\|\texttt{ball}\| = \{e_1, e_3, e_5\}$ y $\|\texttt{cube}\| = \{e_2, e_4, e_6, e_7\}$. Notar que las particiones de `ball` y `cube` hacen que $\top$ no agregue información es decir $\top$ es redundante, por lo tanto podemos borrarla. Quedando las particiones como se muestran en la Figura 4.5.

La siguiente propiedad es `red`, que también supera la probabilidad $\texttt{red.rnd}_{use}$. Tenemos que $\|\texttt{red}\| = \{e_2, e_4, e_5, e_7\}$, haciendo la intersección con la $\|.\|$ de cada fórmula en RE obtenemos, $\{e_5\}$ y $\{e_2, e_4, e_7\}$. Las interpretaciones de las fórmulas en RE se pueden ver en la Figura 4.6. El conjunto RE hasta el momento es $\{\texttt{ball}, \texttt{cube}, \texttt{ball} \sqcap \texttt{red}, \texttt{cube} \sqcap \texttt{red}\}$. Cuando la interpretación de una fórmula es un conjunto singleton lo indicamos con una elipse en lugar de un recuadro.

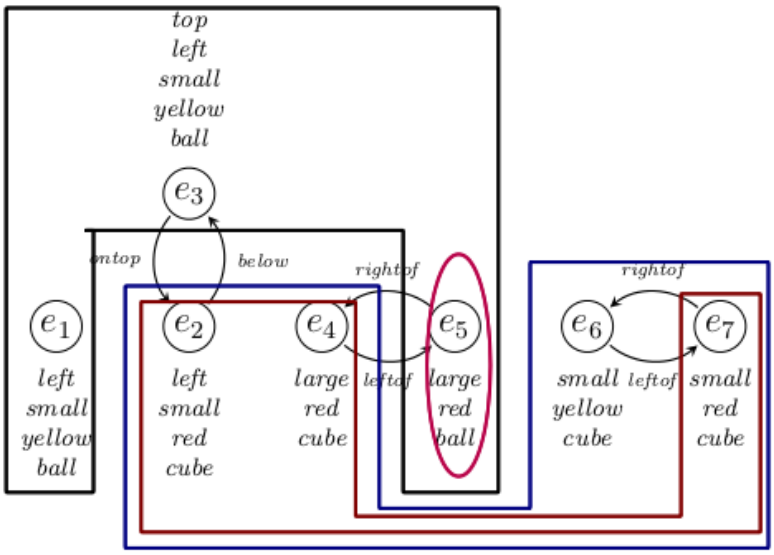


Figura 4.6: Conjuntos de elementos que cumplen las fórmulas: `ball`, $\texttt{ball} \sqcap \texttt{red}$, `cube`, $\texttt{cube} \sqcap \texttt{red}$.

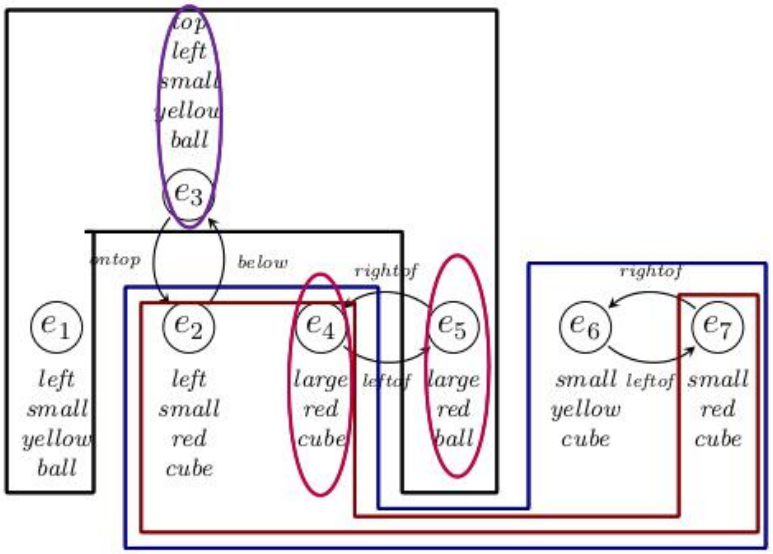


Figura 4.7: Cuadros indicando particiones de elementos que cumplen las fórmulas: `ball`, $\texttt{ball} \sqcap \texttt{red} \sqcap \texttt{large}$, `cube`, $\texttt{cube} \sqcap \texttt{red}$, $\texttt{cube} \sqcap \texttt{red} \sqcap \texttt{large}$, $\texttt{ball} \sqcap \exists \texttt{ontop.cube} \sqcap \texttt{red}$.

Ahora es el turno de `large` que también supera la probabilidad $\text{R.}\texttt{rnd}_{use}$. Tenemos que $\|\texttt{large}\| = \{e_4, e_5\}$, haciendo la intersección con las demás fórmulas que tenemos hasta el momento nos queda RE de la siguiente manera: $\{\texttt{ball}, \texttt{cube}, \texttt{ball} \sqcap \texttt{red} \sqcap \texttt{large}, \texttt{cube} \sqcap \texttt{red}, \texttt{cube} \sqcap$

red⊓large}. Notar que como estamos en la primera iteración, y no chequeamos informatividad, se agregó large a ball ⊓ red se agregó sobreespecificando, porque no dividía el conjunto.

La siguiente relación es ontop, la cual supera la probabilidad ontop.rnd$_{use}$. e_3 es el único elemento que está ontop quedando RE = {ball, cube, ball ⊓ red ⊓ large, cube ⊓ red, cube ⊓ red ⊓ large, ball ⊓ ∃ontop.(cube ⊓ red)}. Los conjuntos resultantes se muestran en la Figura 4.7.

Siguiendo con yellow, tenemos que ‖yellow‖ = $\{e_1, e_3, e_6\}$ y obtenemos RE = {ball ⊓ yellow, cube ⊓ yellow, ball ⊓ red ⊓ large, cube ⊓ red, cube ⊓ red ⊓ large, yellow ⊓ ball ⊓ ∃ontop.cube ⊓ red}. Algunas relaciones se agregaron sobreespecificando, ya que todavía estamos en el primer ciclo. Aquí ya borramos la fórmula ball porque era redundante, y la fórmula cube también. Se muestran las particiones resultantes en la Figura 4.8.

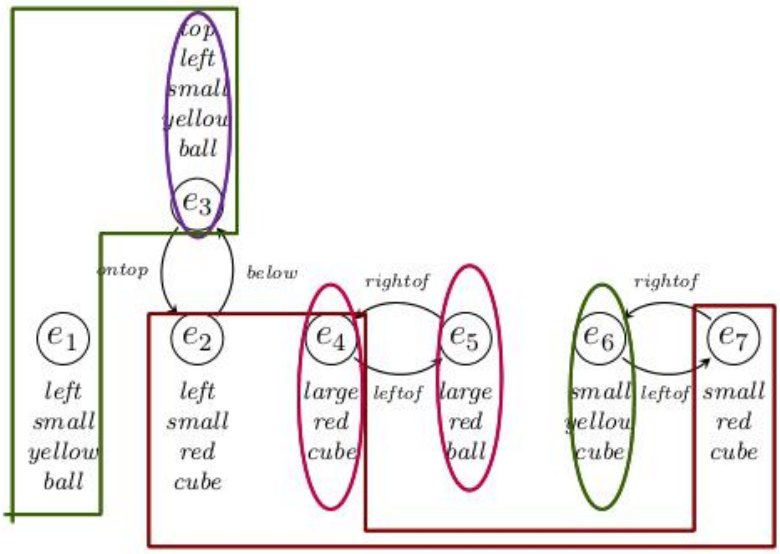


Figura 4.8: Cuadros indicando particiones de elementos que cumplen las fórmulas: ball⊓red⊓large, cube⊓red, cube⊓red⊓large, ball⊓∃ontop.cube⊓red, cube⊓yellow, ball⊓yellow.

Las demás propiedades no pasan la probabilidad R.rnd$_{use}$, por lo tanto salimos del primer ciclo, e incrementamos las probabilidades con R.inc$_{use}$, hasta que alguna relación R es tal que R.rnd$_{use}$ ≤ R.p$_{use}$, quedando como se muestran en la Tabla 4.7.

Las interpretaciones de las fórmulas hasta el momento e ilustradas gráficamente en la Figura 4.8 son:

‖ball ⊓ yellow‖ = $\{e_1, e_3\}$
‖cube ⊓ yellow‖ = $\{e_6\}$
‖ball ⊓ red ⊓ large‖ = $\{e_5\}$
‖cube ⊓ red‖ = $\{e_2, e_4, e_7\}$
‖cube ⊓ red ⊓ large‖ = $\{e_4\}$
‖yellow ⊓ ball ⊓ ∃ontop.cube‖ = $\{e_3\}$

De ahora hasta que termine el algoritmo, no se agregarán propiedades por sobreespecificación. Notar que ya tenemos 4 conjuntos singleton, los cuales no cambiarán hasta el final de la ejecución. El algoritmo continúa porque siguen quedando clases para refinar, las cuales son ball ⊓ yellow y cube ⊓ red.

La relación small no pasa la probabilidad random, pero rightof sí. Continuamos con rightof. Agregamos cube ⊓ red ⊓ ∃rightof.cube ⊓ yellow.

R	*ball*	*cube*	*red*	*large*	*ontop*	*yellow*	*small*	*rightof*	*leftof*	*top*	*left*	*below*
R.p_{use}	1.0	1.0	0.9868	0.5542	0.5068	0.49	0.4642	0.4042	0.4	0.4	0.4	0.4
R.rnd_{use}	0.323	0.560	0.877	0.125	0.142	0.039	0.709	0.218	0.857	0.816	0.202	0.13
R.inc_{use}	0	0	0.004	0.148	0.164	0.17	0.178	0.198	0.2	0.2	0.2	0.2

Tabla 4.7: Probabilidad R.p_{use} de las propiedades y relaciones, incrementadas en R.inc_{use} luego del primer ciclo de ejecución.

La siguiente que pasa la probabilidad random es `left`. Se agrega la fórmula `left` ⊓ `red` ⊓ `cube`, y su correspondiente interpretación. Se muestran los conjuntos resultantes en la Figura 4.9. `cube` ⊓ `red` se elimina porque es redundante.

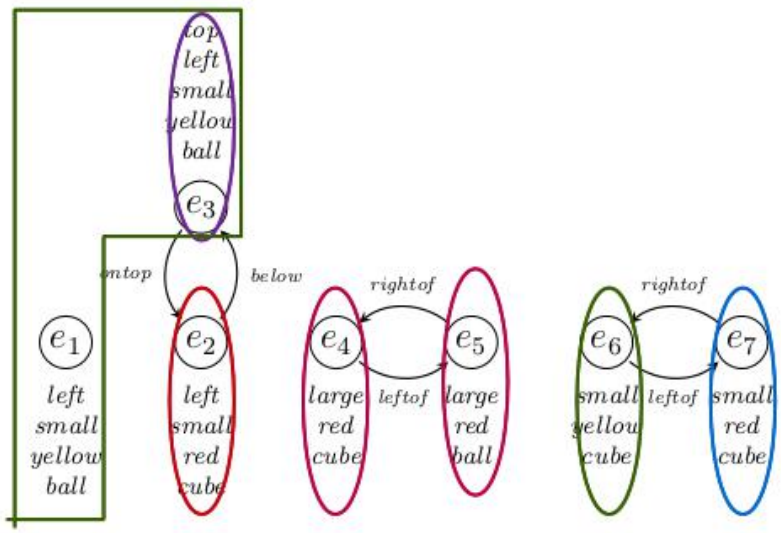


Figura 4.9: Cuadros indicando particiones de elementos que cumplen las fórmulas: `ball` ⊓ `red` ⊓ `large`, `cube` ⊓ `red` ⊓ `large`, `ball` ⊓ ∃`ontop.cube` ⊓ `red`, `cube` ⊓ `yellow`, `ball` ⊓ `yellow`, `cube` ⊓ `red` ⊓ ∃`rightof.cube` ⊓ `yellow`, `left` ⊓ `red` ⊓ `cube`.

Subiendo un par de veces las probabilidades de uso, el algoritmo finaliza por no poder hacer más refinamientos quedando las siguientes clases y sus interpretaciones:

$\|\texttt{ball} \sqcap \texttt{yellow}\| = \{e_1, e_3\}$
$\|\texttt{left} \sqcap \texttt{cube} \sqcap \texttt{red}\| = \{e_2\}$
$\|\texttt{yellow} \sqcap \texttt{ball} \sqcap \exists\texttt{ontop.cube} \sqcap \texttt{red}\| = \{e_3\}$
$\|\texttt{cube} \sqcap \texttt{red} \sqcap \texttt{large}\| = \{e_4\}$
$\|\texttt{ball} \sqcap \texttt{red} \sqcap \texttt{large}\| = \{e_5\}$
$\|\texttt{cube} \sqcap \texttt{yellow}\| = \{e_6\}$
$\|\texttt{cube} \sqcap \texttt{red} \sqcap \exists\texttt{rightof.cube} \sqcap \texttt{yellow}\| = \{e_7\}$

Notar que `small` no se agrega a `ball` ⊓ `yellow` porque no es informativo. Concluimos que el lenguaje elegido $\mathcal{EL}$ no pueden identificar a e_1 en el modelo. Si usáramos $\mathcal{ALC}$, que contiene negación, e_1 se podría distinguir de e_3. Los demás elementos sí quedan en clases singleton y son expresiones referenciales, podrían realizarse como sigue: *el cubo amarillo* es e_6, *la bola grande roja* es e_5, *el cubo rojo grande* es e_4, *el cubo rojo que está a la derecha del cubo amarillo* es e_7, e_3 es *la bola amarilla sobre el cubo rojo* y e_2 es *el cubo rojo de la izquierda.*

4.4.1 Asegurando terminación generando expresiones relacionales

Uno de los problemas que tenían otros algoritmos es que no siempre terminaban como explicamos en el Capítulo 2. Para asegurar terminación nosotros incrementamos las probabilidades p_{use} en cada ciclo, hasta que alcancen 1. Definimos MaxIterations como la cantidad máxima de iteraciones que vamos a permitir, y calculamos R.inc_{use} es cuanto vamos a incrementar la probabilidad de R en cada ciclo, R.inc_{use} = (1 - R.p_{use}) / MaxIterations, con esto nos aseguramos que en MaxIterations todas lleguen a 1, y en consecuencia el algoritmo termine, ya que en el código, se incrementa R.p_{use} con R.inc_{use} y sale del ciclo principal si para todas las R, R.p_{use} son mayores o iguales a 1.

El fragmento del Algoritmo 8 definido por el while, es un algoritmo de punto fijo, se refinan las particiones hasta que en algún momento no se pueden hacer más refinamientos, se dice que el algoritmo llegó a un punto fijo, en el que no le es posible progresar. RE se mantiene sin modificaciones. Los conjuntos se van haciendo más chicos en las sucesivas iteraciones y considerando que la cantidad de elementos del modelo y las relaciones son finitas, se llega al punto fijo en una cantidad finita de pasos y el algoritmo siempre termina.

Nuestro algoritmo permite generar relaciones binarias con otros objetos e incluir descripciones de los demás objetos (en las ERs). No tiene preferencia por las relaciones ni por las propiedades proposicionales (como otros algoritmos descriptos de trabajo previo), en el sentido de que va a incluirlas de acuerdo a la probabilidad de uso que ellas tengan, las cuales fueron calculadas o aprendidas de un corpus. En el ejemplo de ejecución anterior mostramos que el algoritmo agregó la fórmula `cube`$\sqcap$`red`$\sqcap\exists$`rightof.cube`$\sqcap$`yellow` que incluye la relación binaria `rightof` y por ejemplo no se incluyó *small*.

4.4.2 Generando no-determinísticamente sobreespecificación

Al iniciar, el algoritmo calcula para cada R, R.rnd_{use} que es un número aleatorio entre 0 y 1, ese número va a hacer que el algoritmo sea no-determinístico, ya que, el algoritmo usará R para refinar las clases solamente si R.$\mathsf{rnd}_{use} \leq$ R.p_{use}. Por lo tanto y considerando que en las siguientes ejecuciones R.rnd_{use} serán distintos, va a poder generar distintas ERs.

El algoritmo hace caso omiso de la restricción de informatividad (es decir, que permite la inclusión de nuevas relaciones en la descripción, incluso si no refinan la clase asociada) *pero sólo durante el primer ciclo del algoritmo*. Es decir, durante el primer ciclo del repeat principal del algoritmo, se permitirá la inclusión de todas las relaciones que no trivializan la descripción (es decir, la clase asociada no está vacía y que no son redundantes). Debido a que esto se hace sólo durante el primer ciclo, sabemos que no aparecerán propiedades repetidas en las ERs generadas (como `green green ball`). En los ciclos restantes, se añadirán propiedades adicionales sólo si son de carácter informativo. Este diseño del algoritmo que permite sobreespecificación está inspirado en la obra de [Keysar et al., 1998] sobre el egocentrismo y la producción del lenguaje natural. Keysar et al. argumentan que cuando se produce el lenguaje, el hablante no identifica unívocamente al target desde el principio, sino que es más bien un ajuste de último momento. Los hablantes producen ERs primero egocéntricamente, agregando aquellas propiedades más prominentes por más de que no sean informativas. Pero luego las ajustan de modo que el destinatario de la ER sea capaz de identificarla unívocamente. El primer paso, egocéntrico, es un proceso heurístico basado en un modelo de la prominencia de la propiedades de la escena que contiene el target. Nuestra definición de p_{use} está destinada a capturar las prominencias de las propiedades de diferentes escenas y targets. Los valores de p_{use} cambian en relación a la escena considerada. Esto está en contraste con el trabajo previo en GER descripto en el Capítulo 2 donde la prominencia de una propiedad es constante en un dominio. Keysar et al. argumentan que la razón del procedimiento de generar-y-ajustar puede tener que ver con las limitaciones de procesamiento de información de la mente: si la heurística que guía la fase

egocéntrica está bien sintonizada, consigue una adecuada ER al principio y rara vez requerirá ajustes. Se observa un comportamiento similar con nuestro algoritmo: cuando los valores de p_{use} son aprendidos del dominio, el algoritmo no es sólo es más preciso, sino que también es mucho más rápido que cuando se utilizan valores de p_{use} aleatorios.

4.4.3 Generando expresiones referenciales plurales

Como dijimos el algoritmo puede dar ERs para todos los elementos del modelo.

Para dar las ERs plurales simplemente damos la conjunción de las ERs de los elementos que pertenecen al conjunto target. El algoritmo en algunas ocasiones particiona por alguna relación en la que quedan los elementos del target juntos, generando así expresiones referenciales colectivas, pero esto no pasa normalmente. En el Capítulo 6 se muestra un caso de estudio de ER plurales.

4.5 Notas finales y linkeo del capítulo

En este capítulo se explicó la entrada y salida de un algoritmo probabilístico. Una de las entradas es el modelo, el cual es una representación de la figura considerada. Otra es una distribución de probabilidades finita, la cual explicamos cómo obtenerla en 2 casos. Primero en el caso que hay corpus disponible para la escena considerada y en casos donde al no haber corpus disponible proponemos una aproximación usando aprendizaje automático. Se explicó el algoritmo en detalle y se ilustró con un ejemplo de ejecución. Se explicó cómo hace el algoritmo para conseguir las siguientes características: asegurar terminación, generar ER relacionales, generar ER no-determinísticas y generar sobreespecificación. El diseño del algoritmo está inspirado en el modelo cognitivo de generación de expresiones referenciales de [Keysar et al., 1998]. En el Capítulo 5 veremos que este algoritmo es capaz de generar rankings de ERs cercanos a los encontrados en corpora. El algoritmo presentado en este capítulo introduce una familia de algoritmos que extiende probabilísticamente a los algoritmos presentados en el Capítulo 3. Se explicaron los detalles de la subrutina *add* para el lenguaje $\mathcal{EL}$. Las definiciones de *add* para otros lenguajes como $\mathcal{ALC}$ o $\mathcal{FO}^-$ son similares. Como se discutió en el Capítulo 1, los dominios realistas de posibles aplicaciones de la GER contienen incertidumbre. En este capítulo modelamos esa incertidumbre como las probabilidades de usar una característica en una ER. Los algoritmos presentados en este capítulo pueden también ser usados con probabilidades provenientes de confidence scores de sensores, por ejemplo, para modelar otros tipos de incertidumbre. En el Capítulo 6 evaluamos estos algoritmos sobre un corpus de descripciones de mapas, el corpus ZOOM.

Capítulo 5

Evaluación de rankings sobre benchmarks

En este capítulo se presentan diversas formas de evaluar a los algoritmos propuestos en el capítulo anterior. En particular, se muestra que el algoritmo de refinamiento probabilístico con sobreespecificación es capaz de generar un ranking de ERs que es similar a la distribución de frecuencias de las ERs observadas en corpora usando métricas automáticas y que también tiene un buen desempeño cuando se usan métricas manuales. Parte de esta evaluación fue presentada en el paper [Benotti and Altamirano, 2013].

Este capítulo está dividido en 4 secciones, en la Sección 5.1 se muestra una evaluación automática con respecto al GRE3D7. En la Sección 5.2 se presenta una evaluación automática con respecto a resultados de una competencia sobre el corpus TUNA. Luego en la Sección 5.3 se presenta una evaluación de jueces humanos con respecto al corpus TUNA. Para finalizar en la Sección 5.4 se describe el resumen y linkeo del capítulo.

5.1 Evaluación de rankings sobre el corpus GRE3D7

En esta sección presentamos una evaluación cuantitativa y automática de los algoritmos en el dominio del corpus GRE3D7 [Viethen and Dale, 2011] introducido en la Sección 2.1.3 del Capítulo 2. Mostraremos una comparación con el ranking de ERs dadas por el algoritmo con las probabilidades de uso calculadas como se describe en la Sección 4.2, es decir con aprendizaje automático a partir de las demás imágenes del corpus y ejecutando nuestro algoritmo 10000 veces. El algoritmo probabilístico con sobreespecificación es capaz de generar una distribución de ERs similar a la que se observa en el corpus. Describimos primero en detalle los experimentos que realizamos para la escena que se muestra en la Figura 5.1, en la Sección 5.1.1. Luego, en la Sección 5.1.2, resumimos los resultados obtenidos en evaluaciones similares para otras 7 escenas del corpus.

5.1.1 Caso de estudio de una escena del corpus

Las probabilidades de uso aprendidas usando regresión lineal como se muestra en la Sección 4.2 se muestran en la Tabla 5.1.

R	*ball*	*cube*	*green*	*blue*	*large*	*small*	*ontop*	*left*	*top*	*front*	*below*
R.p_{use}	1.0	1.0	0.993	0.124	0.03	0.346	0.179	0.002	0	0	0

Tabla 5.1: Distribución de probabilidad de las propiedades y relaciones de la figura de ejemplo.

Ejecutando el algoritmo 10000 veces, obtuvimos 14 expresiones referenciales diferentes para la escena de la Figura 5.1, el corpus tiene 12 ERs diferentes generadas por personas para esa escena. Es interesante ver que aunque es posible generar cientos de ERs para esta escena,

el algoritmo, guiado por las probabilidades de uso, genera sólo 14 ERs en 10000 ejecuciones. Además, el algoritmo genera, para esta imagen, 2 ERs con alta frecuencia (la bola verde y la bola verde pequeña representan el 98% de las ERs generadas automáticamente) y otras ERs con una frecuencia mucho menor (representan el 2% de las ERs generadas). Estas 2 ERs más frecuentes generadas por el algoritmo coinciden con las 2 ERs más frecuentes generadas por las personas (la bola verde y la bola verde pequeña representan el 81% de las ERs generadas por humanos para la imagen).

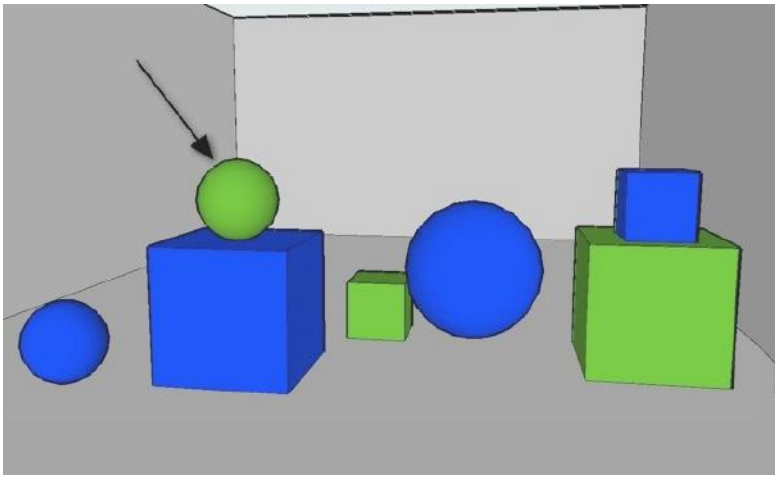


Figura 5.1: Imagen del GRE3D7 corpus.

La **exactitud** es una métrica automática estricta que se puede usar para comparar rankings de ERs. Es la proporción de coincidencias perfectas entre la salida de algoritmo y las ERs generadas por humanos encontradas en corpora. La primer comparación de rankings de ERs se muestra en la Tabla 5.2. Para cada ER, indicamos el número de veces que aparece en el corpus (#Cor), la proporción que representa (%Cor), el número de veces que la generó nuestro algoritmo (#Alg) y la proporción que representa (%Alg). Por último, la exactitud (%Acc) nos da el porcentaje de aciertos de las ERs generadas por el algoritmo y se calcula como el mínimo entre el %Cor y %Alg.

La exactitud del algoritmo con respecto al corpus GRE3D7 para esta escena es del 80%. De las 14 ERs diferentes generadas por el algoritmo, 5 se encuentran en el corpus y las otras 9 no. Estas 9 expresiones referenciales incluyen propiedades de ubicación del target no con respecto a un landmark sino a su posición en la escena 3D. Por ejemplo *la esfera verde que está a la izquierda* se refiere a que la esfera verde está a la izquierda de la escena. La probabilidad de uso de este tipo de propiedades es menor al 1%, en particular la de left es 0,007, pero, como esa probabilidad no es cero, dichas propiedades aparecen en ERs (siempre con menos de un 1% de frecuencia). La métrica de exactitud ha sido utilizada en trabajos anteriores para comparar la salida de un algoritmo de generación de ERs con las ERs que se encuentran en corpora [van der Sluis et al., 2007; Viethen, 2011] y se considera una métrica muy estricta para esta tarea.

5.1.2 Evaluación automática sobre el corpus

Como es habitual cuando hacemos una evaluación con un caso particular, podemos pensar que como el algoritmo tiene un componente de azar, por eso nos dio bastante bien, para esa ejecución particular. Por eso aquí se muestra la misma comparación realizada en la sección anterior pero con todas las imágenes verde-azules del corpus GRE3D7. Para enriquecer esta parte mostraremos otros baselines, los cuales nos darán pistas de que las probabilidades de uso aprendidas con el método que presentamos en esta tesis son útiles para aproximar al ranking de ERs que aparece en corpora. Un baseline que se presenta ejecuta el algoritmo con una distribución aleatoria de probabilidades de uso. Llamamos uniforme a otro baseline, en el cual

Expresiones Referenciales	Corpus		Algoritmo		Exactitud
	#Cor	%Cor	#Alg	%Alg	%Acc
ball, green	91	65.00	6376	63.76	63.76
ball, green, small	23	16.43	3440	34.40	16.43
ball, green, small, ontop(blue, cube, large)	8	5.71	0	0.00	0.00
ball, green, ontop(blue, cube)	5	3.57	0	0.00	0.00
ball, green, ontop(blue, cube, large)	5	3.57	0	0.00	0.00
ball, green, small, ontop(blue, cube)	2	1.43	0	0.00	0.00
ball, ontop(cube)	1	0.71	27	0.27	0.27
ball, green, small, ontop(blue, cube, large, left)	1	0.71	0	0.00	0.00
ball, small, ontop(cube,large)	1	0.71	2	0.02	0.02
ball, green, top	1	0.71	0	0.00	0.00
ball, small, ontop(cube)	1	0.71	3	0.03	0.03
ball, green, ontop(cube)	1	0.71	0	0.00	0.00
ball, front, green	0	0.00	97	0.97	0.00
ball, front, green, small	0	0.00	13	0.13	0.00
ball, front, top	0	0.00	12	0.12	0.00
ball, green, left	0	0.00	11	0.11	0.00
ball, top	0	0.00	10	0.10	0.00
ball, green, left, small	0	0.00	5	0.05	0.00
ball, left, top	0	0.00	2	0.02	0.00
ball, small, top	0	0.00	1	0.01	0.00
ball, front, ontop(cube, left)	0	0.00	1	0.01	0.00
Total	140	100	10000	100	80.51

Tabla 5.2: ERs del corpus, y las producidas por nuestro algoritmo para la Figura 5.1.

tomamos las ERs generadas por el sistema, las del corpus y las generadas con el modelo aleatorio les dimos la misma probabilidad de ocurrencia. Por ejemplo si eran 10 en total, y ejecutamos el algoritmo 100 veces, cada ER iba a aparecer 10 veces. En este baseline no ejecutamos el algoritmo. El Top baseline, es en el que el algoritmo usó las probabilidades de uso sacadas del corpus mismo.

	p_{use} de escena	p_{use} aprendidas	p_{use} random	uniforme
Escena 1	85.75%	84.49%	17.95%	5.37%
Escena 3	82.81%	80.51%	9.89%	4.40%
Escena 6	90.11%	83.30%	4.13%	4.16%
Escena 8	86.52%	64.06%	16.32%	9.75%
Escena 10	89.49%	75.80%	7.56%	3.70%
Escena 12	80.21%	81.29%	57.09%	6.68%
Escena 13	89.98%	50.79%	9.30%	3.59%
Escena 21	92.13%	80.01%	8.45%	6.77%
Promedio	87.13%	75.03%	16.34%	5.55%

Tabla 5.3: exactitud entre las ERs del corpus y las generadas usando valores de p_{use} calculados desde la escena, aprendidos automáticamente, provenientes de distribuciones random y uniforme.

La primera columna muestra los valores obtenidos cuando corremos el algoritmo sobre la escena con los valores de p_{use} obtenidos *de la propia escena*. Como se puede esperar, esta columna tiene el mayor promedio de exactitud.

La segunda columna muestra los resultados del algoritmo cuando se ejecuta con p_{use} aprendido de corpora como se explica en la Sección 4.2 del capítulo anterior. Este es el caso más

interesante porque muestra la capacidad de nuestra propuesta de generalizar a escenas no vistas con anterioridad y para las que no hay corpus. Para la mayoría de las escenas la exactitud es mayor al 80% y la exactitud promedio es 75%. La relativamente baja exactitud obtenida en la escena 13 se explica principalmente por las pobres estimaciones del valor de p_{use} para las palabras *large* y *small* que son propiedades vagas y no absolutas ya que dependen de cuán grandes son los objetos en relación con los otros elementos del contexto.

En el corpus, las relaciones *large* y *small* se utilizan mucho más cuando el target no puede ser identificado usando sólo propiedades taxonómicas (*ball* y *cube*) y propiedades absolutas (*green* y *blue*), pero las características que hemos utilizado para el aprendizaje automático no capturan dichas dependencias como se discute en la Sección 4.2.1 del Capítulo 4. A pesar de esta limitación, el promedio de la segunda columna es 75%. Para una métrica como la de exactitud que es considerada demasiado estricta, estos resultados son buenos. Además, el 25% no cubierto por el algoritmo puede contener ERs igualmente buenas como vamos a mostrar en la Sección 5.3. Uno podría argumentar entonces que los valores de p_{use} aprendidos a partir del corpus son lo suficientemente buenos para ser utilizados para generar ERs para nuevas escenas del dominio. Las dos últimas columnas pueden ser consideradas como baselines. En la primera generamos valores aleatorios para p_{use} y luego ejecutamos el algoritmo con esos valores de p_{use}. La exactitud obtenida es en la mayoría de los casos pobre, pero con una variación notable debido al azar. Además estas ejecuciones toman mucho más tiempo en finalizar, lo que indica que se están realizando demasiadas particiones sin conseguir llegar a la meta. Esta observación de performance es consistente con los resultados de los experimentos de generación humana realizados por [Keysar et al., 1998] y discutidos en la Sección 2.1.2 del Capítulo 2. Keysar argumenta que, las personas usan la prominencia de las propiedades en el dominio como heurística para guiar la generación de ERs de forma de disminuir la carga cognitiva de tener que probar con todas las propiedades del dominio. Si la heurística es buena, en muchos casos no es necesario "revisar" la ER para que identifique unívocamente al target. Como efecto colateral de este proceso heurístico la ER resultante puede estar sobreespecificada, pero se genera rápidamente. Además de poca exactitud, cuando se utilizaron probabilidades random, muchas de las ERs generadas suenan poco naturales y son difíciles de realizar sin introducir ambigüedades como, por ejemplo, (*pequeña cosa sobre el cubo azul que está abajo de algo que es pequeño*). En la última columna se presenta la exactitud de una corrida artificial, le llamamos uniforme, uniforme tomando todas las ERs de las demás columnas y asignándoles la misma probabilidad. La uniforme consigue los peores resultados, ya que acumula ERs que no son del corpus, y las del corpus las tienen baja probabilidad. Para comparar nuestros resultados con los 3 baselines usamos también la entropía cruzada.

En teoría de la información, la **entropía cruzada** entre dos distribuciones de probabilidad mide la media de bits necesarios para identificar un evento de un conjunto de posibilidades, si un esquema de codificación está basado en una distribución de probabilidad dada q, más que en la verdadera distribución p. Vamos a comparar la entropía cruzada entre la distribución de probabilidad que se encuentra en el corpus, y la distribución de ERs del corpus y las de ejecuciones del algoritmo con las probabilidades que acabamos de describir (ver [Jurafsky and Martin, 2008] para obtener detalles sobre evaluación de entropía cruzada). En la Figura **??** se muestran los resultados para las ocho escenas que hemos considerado.

Las entropías cruzadas de las dos primeras ejecuciones (*escena* y *aprendizaje automático*) son, en general, mucho más cercanas de la entropía del corpus, que las entropías cruzadas de *random* y *uniforme*. Sólo en la escena 12 random, por azar se acerca un poco más.

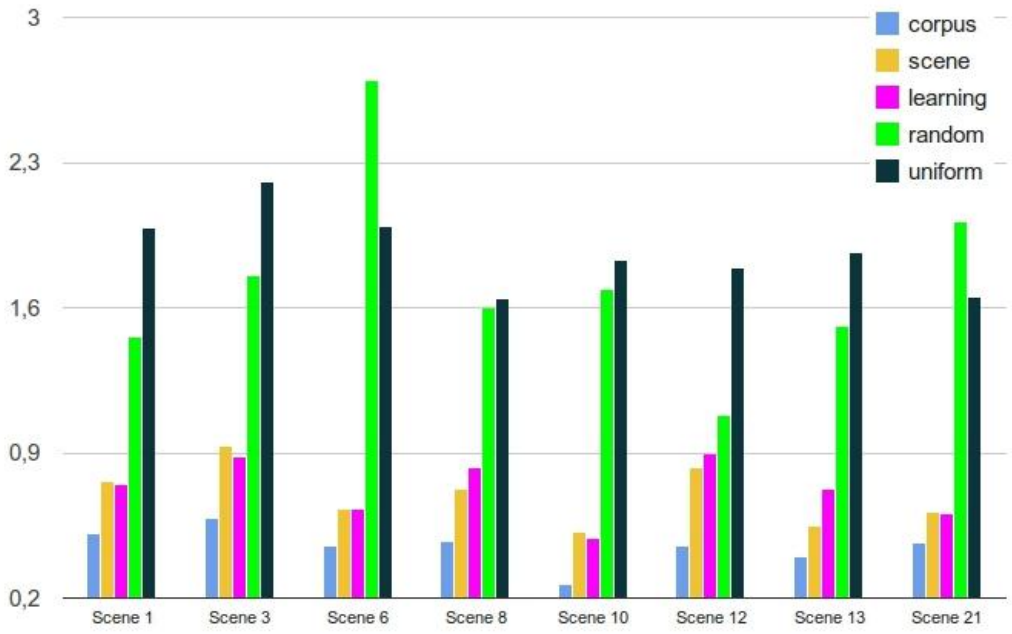


Figura 5.2: Entropía cruzada entre la distribución del corpus y diferentes baselines.

5.2 Evaluación de rankings sobre el corpus TUNA

En esta sección se presenta la comparación de nuestro algoritmo con el algoritmo que tuvo el mejor desempeño en el TUNA Challenge.

5.2.1 El TUNA Challenge

El TUNA Challenge, fue una serie de 2 desafíos organizados en el 2008 y 2009 en el que se pedía a los participantes desarrollar programas para generación de expresiones referenciales, donde podían generar sólo la selección de contenido o dar la realización sintáctica también. Se daba parte del corpus TUNA como entrenamiento, y un software de evaluación automática, el cual comparaba la salida del algoritmo con la ER humana del corpus TUNA. El algoritmo GRAPH fue el que mejores puntuaciones sacó en la competencia. El algoritmo GRAPH que describimos en el Capítulo 2 es un algoritmo determinístico y por lo tanto produce la misma expresión referencial cuando se ejecuta con el mismo target y contexto. Nuestro algoritmo es no-determinístico, puede dar una expresión referencial diferente cada vez que se ejecuta. Con el fin de compararlos corremos nuestro algoritmo 100 veces y hacemos un ranking de las 20 ERs más frecuentes generadas por el algoritmo. Utilizamos la parte de prueba del corpus TUNA (el cual fue introducido en la Sección 2.1.3), para comparar el ranking de ERs dado por nuestro algoritmo con las salidas del algoritmo GRAPH cuyos resultados se describen en [Krahmer et al., 2008] y se reproducen en la Tabla 5.6.

El algoritmo GRAPH define la generación de expresiones referenciales como un problema de búsqueda en un grafo, devuelve el grafo distintivo de menor costo (si existe) dada una función de costo particular. Comparamos a este algoritmo usando las métricas exactitud, Dice y MASI, introducidas en la Sección 2.3.2.

Recordemos que el corpus TUNA tiene 2 partes, una parte cuyo dominio son muebles, y otra parte cuyo dominio son personas, ambos ubicados en una grilla. El corpus tiene una parte con target singular, y otra con targets plurales. Para realizar esta comparación se usó solamente la parte singular del corpus.

En la Tabla 5.4 y 5.5 se muestran las probabilidades de uso aprendidas desde el corpus. Se puede ver en la Tabla 5.4 por ejemplo que la probabilidad de usar *blue* es mucho más alta que

la probabilidad de usar *facing left* para describir al target. La probabilidad de usar *green* no es 0 porque se puede usar para describir a un landmark.

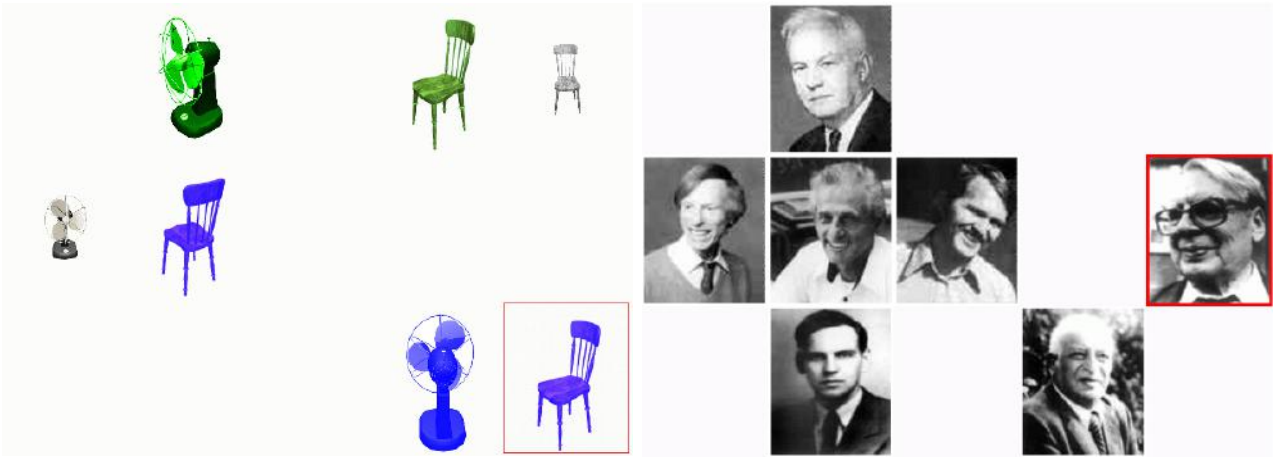
Para aprender las probabilidades de uso se siguió el mecanismo descripto en el Capítulo 4, se unificó el vocabulario del corpus y se aprendió una función de regresión lineal por cada propiedad y se reemplazaron las variables por los valores correspondientes a cada escena. Se usó la parte de entrenamiento del corpus que consistía del 80% del corpus TUNA.

Relaciones Figura 5.3a	p_{use} Aprendida
chair	0.94
blue	0.89
y3	0.29
x5	0.27
left	0.25
large	0.21
green	0.05
small	0.05
back	0.02
y1	0.02

Tabla 5.4: Probabilidades aprendidas desde el corpus de la Figura 5.3a

Relaciones Figura 5.3b	p_{use} Aprendida
person	0.79
hasGlasses	0.71
y2	0.20
x5	0.18
hasHair	0.13
hairDark	0.13
hairLight	0.11
ageOld	0.05
y3	0.03
x2	0.02

Tabla 5.5: Probabilidades de uso aprendidas desde el corpus para la Figura 5.3b



(a) El target de la escena es *blue chair facing left.* (b) El target de la escena es *man with glasses.*

Figura 5.3: Escenas usadas durante la recolección del corpus TUNA.

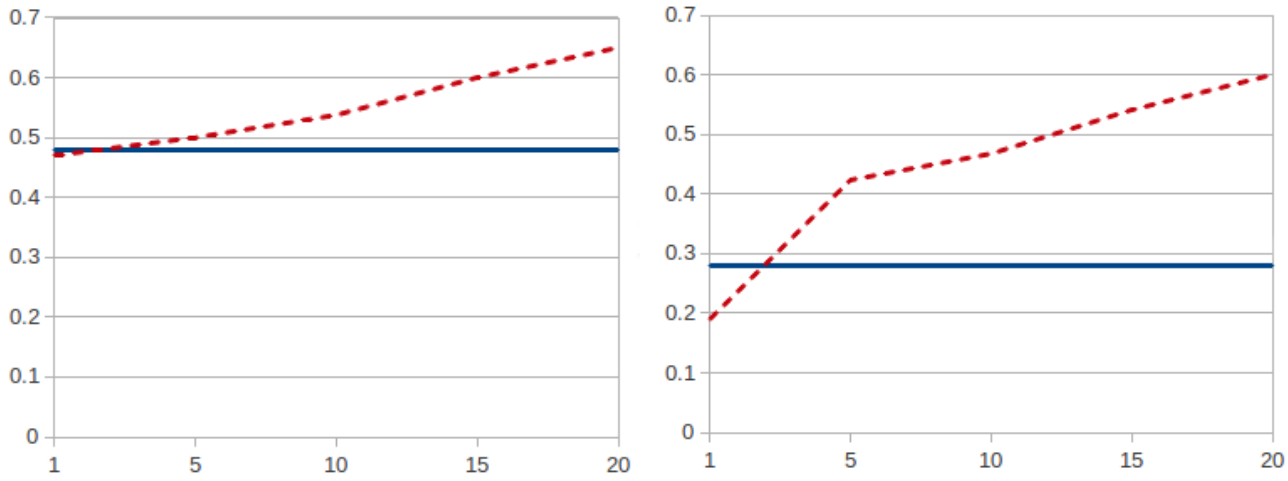
Recordemos las métricas automáticas nombradas en la Sección 2.3.2, la **exactitud** se define como el porcentaje de coincidencias exactas entre cada ER producida por un ser humano y la producida por el sistema para la misma escena y target. Se considera que es una métrica demasiado estricta. El coeficiente **Dice** es una métrica de comparación de conjuntos al igual que MASI, el valor va entre 0 y 1, 1 indica un matcheo perfecto entre los conjuntos. La diferencia entre ellas es que MASI varía en favor de la similaridad cuando un conjunto es un subconjunto de otro. Las dos son métricas que no son tan estrictas como la exactitud por ejemplo con respecto al orden las palabras y justamente es lo que necesitamos ya que no estamos en la parte de realización sintáctica.

En la Tabla 5.6 mostramos métricas automáticas y comparamos la performance de nuestro sistema con el sistema GRAPH para la primer ER del ranking (es decir, la más frecuente) y para las primeras 20 ERs del ranking. En la Figura 5.4 se muestra una comparación entre la

	Dice	MASI	Exactitud
sistema GRAPH, Dominio muebles	80%	59%	48%
sistema GRAPH, Dominio personas	72%	48%	28%
Nuestro sistema, Dominio muebles (top 1)	80%	60%	47%
Nuestro sistema, Dominio personas (top 1)	65%	37%	19%
Nuestro sistema, Dominio muebles (top 20)	87%	75%	65%
Nuestro sistema, Dominio personas (top 20)	81%	68%	60%

Tabla 5.6: Comparación del algoritmo GRAPH y nuestro sistema. Consideramos 3 métricas automáticas para el top 1 y para el top 20 ERs producidas por nuestro algoritmo.

exactitud de nuestro sistema y el sistema GRAPH. El gráfico de la izquierda corresponde al dominio muebles y el gráfico de la derecha corresponde al dominio personas. En el dominio muebles estamos obteniendo casi los mismos números en las métricas evaluadas. Podemos ver que si tomamos la parte superior, es decir 1 ER (la primera, la más probable), nuestra exactitud es menor que de GRAPH para el dominio de las personas. Sin embargo, si tenemos en cuenta las 20 mejores ERs que nuestro algoritmo es capaz de producir, podemos ver que la exactitud para ambos dominios se hace mayor del 60% (60% para personas y 65% para muebles). Esto demuestra que nuestro algoritmo es capaz de generar ERs que son más similares a las producidas por los seres humanos que el algoritmo GRAPH, aunque estas ERs no estén en primer lugar. Si el algoritmo GRAPH fuera modificado para ser no-determinístico es posible que también mejorara su exactitud en 20 o más ejecuciones. Esta es una línea interesante de trabajo futuro.



(a) Muebles.

(b) Personas.

Figura 5.4: Comparación de la exactitud del algoritmo GRAPH y nuestro sistema. El eje x indica las x primeras ERs en el ranking. El eje y indica la exactitud. Nuestro sistema es representado como una línea de puntos y GRAPH como una línea continua.

Otro resultado que podemos observar es que la exactitud del top 1 en el dominio personas es mucho menor (19%) que para el dominio de muebles (47%), pero la exactitud se estabiliza cuando se consideran más ERs de nuestro ranking. Esto puede explicarse por el hecho de que el conjunto del dominio de personas contiene muchas más propiedades que se pueden elegir para describir a las personas que las que contiene el dominio de muebles.

Normalmente las métricas automáticas tienen la desventaja de calificar como malas a ERs que son buenas ERs, es decir identifican el target unívocamente, en el contexto considerado, pero las consideran malas por no ser exactamente como las que aparecen en corpora. Otra manera de evaluar qué tan buena es una ER es con evaluaciones manuales. Mostraremos una evaluación humana en la siguiente sección.

5.3 Evaluación humana

En las secciones anteriores describimos evaluaciones automáticas de nuestro algoritmo. En esta sección explicamos una evaluación humana que hicimos de las ERs generadas por nuestro algoritmo con las probabilidades de uso aprendidas para el TUNA corpus y descriptas en la sección anterior.

Como las ERs que generamos las generamos para el idioma inglés, pedimos a dos jueces nativos de idioma inglés evaluar nuestras expresiones referenciales a través de un experimento en la web. Los jueces podrían entrar en el sistema de evaluación varias veces, es decir no tenían que terminar la evaluación en la primera vez, para cada ER podían resolverla o podían volver a ella más tarde.

Durante la evaluación mostramos a cada juez una escena y dos ERs ordenadas al azar. Una ER correspondía a la ER presente en el corpus TUNA producida por una persona para la escena y la otra ER correspondía a la ER producida por nuestro sistema para la misma escena. Solicitamos a los jueces seleccionar la ER que sería más útil para identificar el target en la escena. El juez podía elegir una ER o indicar que las 2 ERs le parecían igualmente buenas.

Nuestro objetivo es mostrar que incluso si la ER generada automáticamente no coincide con la ER producida por un ser humano del corpus, puede ser juzgada como buena o incluso mejor que la ER generada por una persona.

En la Tabla 5.7 se muestran los resultados del experimento de evaluación humana. Las ERs producidas por el sistema en el top 1 fueron consideradas igual o mejor por los 2 jueces que las generadas por personas en el 75% de los casos para los muebles y en el 43% para las personas. Esto contrasta con el 47% para muebles y 19% para personas de la Tabla 5.6. Esto muestra que la métrica de exactitud es demasiado estricta para la tarea, y las métricas de DICE o MASI se acercan más a evaluaciones humanas. Además al menos 1 juez consideró que el 97% de las ERs del sistema eran tan buenas o mejores que las humanas para los muebles y (87% para las personas). Esto muestra que la evaluación de la calidad de las ERs es parcialmente subjetiva. Sólo en el 8% de los casos ambos jueces coincidieron que una ER generada automáticamente era peor que la humana.

	Dominio muebles	Dominio personas	Media ponderada
sistema igual o mejor por 2 jueces	.75	.43	.60
sistema igual o mejor por 1 juez	.97	.87	.92
sistema peor por 2 jueces	.03	.13	.08

Tabla 5.7: Evaluación humana de las ERs del corpus con las generadas para escenas del corpus TUNA (parte singular).

A continuación, se ilustra el experimento de evaluación, mostrando ejemplos de casos en los que la expresión del sistema fue considerada mejor por ambos jueces, por un solo juez o por ninguno de los dos.

Figura 5.5 ilustra un caso en el que el humano genera una ER subespecificada mientras que el sistema produce un ER que identifica de manera unívoca al target. La ER generada por el sistema para esta figura es *small blue fan*, mientras que la ER producida por el ser humano es *blue fan*. La ER del humano no logra identificar de forma única el target y entonces no es preferida por los jueces humanos. Los seres humanos son conocidos por producir ERs subespecificadas, esto puede ser debido a las limitaciones cognitivas por no ser capaz de considerar todo el contexto referencial al mismo tiempo. Nuestro algoritmo es capaz de considerar todo el contexto referencial y combinar esta capacidad con la probabilidad de uso de las ERs aprendidas de los seres humanos.

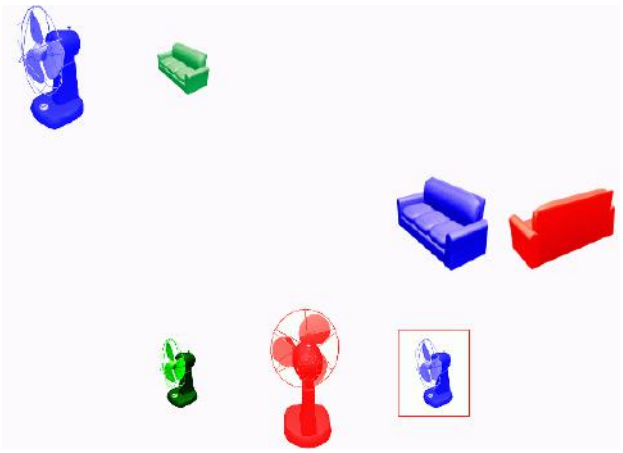


Figura 5.5: Escena usada durante la recolección del TUNA corpus. La ER humana *blue fan*, y la del sistema *small blue fan*. Los jueces prefirieron la ER generada por el sistema.

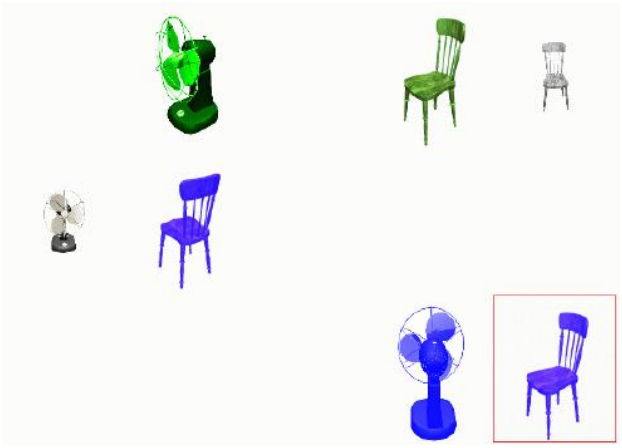


Figura 5.6: Escena usada durante la recolección del TUNA corpus. La ER humana *front blue chair*, y la del sistema *bottom blue chair*. Ambos jueces humanos prefirieron la generada por el sistema.

En la Figura 5.6 la ER humana era *front blue chair*, y la ER sistema era *bottom blue chair*; ambos jueces seleccionaron la ER sistema. Este caso se puede explicar por el hecho de que, en este ámbito, la propiedad *bottom* ayuda más durante la identificación de la propiedad *front* porque concentra la atención del interlocutor en la parte inferior de la escena. Nuestro sistema aprende este hecho por el aprendizaje de un mayor valor de p_{use} para *bottom* que para *front* a partir de los datos de entrenamiento.

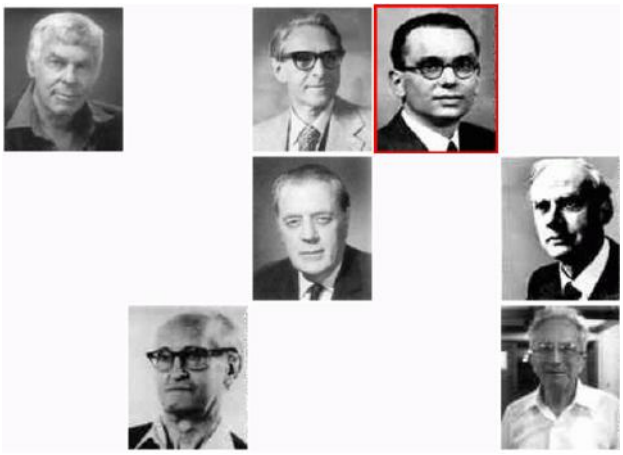


Figura 5.7: Escena usada durante la recolección del TUNA corpus. La ER humana *the man with black hair*, y la del sistema *the man wearing glasses in the fourth column*. Los jueces prefirieron la ER humana.

Figura 5.8: Escena usada durante la recolección del TUNA corpus. La ER humana *man with a beard*, y la del sistema *man with a beard wearing glasses*. Los jueces no estuvieron de acuerdo en su preferencia.

Figura 5.7 es un ejemplo en el que ambos jueces prefirieron la expresión humana. La ER humana era *the man with black hair*, y del sistema de *the man wearing glasses in the fourth column*. Este ejemplo pone de manifiesto el hecho de que, en el dominio de las personas algunas propiedades son más destacadas en algunas imágenes que en otros debido a diferentes tonos de colores. Propiedades graduables como estas (en contraste con las propiedades absolutas) son todavía un problema abierto para los algoritmos de GER.

Figura 5.8 ilustra un caso en el que la ER del sistema era más sobreespecificada que la ER humana; el sistema incluye *wearing glasses*, mientras que el ser humano no lo hizo. En este

caso un sujeto humano prefiere la ER del sistema y el otro la ER del humano. La cantidad de sobreespecificación es una cuestión subjetiva, donde los humanos mismos no están de acuerdo. Una evaluación donde las ERs se utilicen para resolver una tarea sería interesante para investigar este asunto.

5.4 Notas finales y linkeo del capítulo

En este capítulo se presentó una evaluación automática de los rankings de ERs generados por el algoritmo sobre 2 benchmarks del área, uno el corpus GRE3D7 y otro son los resultados de una competencia el TUNA challenge, en este caso nos comparamos con el algoritmo que mejores puntuaciones tuvo en la competencia. Presentamos también una evaluación humana sobre ERs del corpus TUNA y las generadas por el algoritmo. Si bien conseguimos buenas puntuaciones en el siguiente capítulo veremos una evaluación de un caso mucho más natural, en el que consideramos ERs generadas por el algoritmo para puntos de interés en mapas.

Capítulo 6

Recolección y análisis del corpus ZOOM

Hay tres áreas principales en las que los corpora se pueden utilizar en la investigación sobre generación de expresiones referenciales: evaluación, diseño y metodologías para la recolección de corpus útiles para investigación y el análisis y modelización estadística de datos de corpora.

El desafío de recopilación de corpus, se centra en torno al equilibrio que se necesita entre el control de los parámetros experimentales tanto como sea necesario y mantener la configuración del sistema de recolección lo más natural posible.

Para evaluar la salida de un sistema de GER, se puede comparar con los datos de corpora, bajo la premisa de que el objetivo de la comparación es para evaluar si el sistema podría tener un modelo adecuado del comportamiento humano para la generación de expresiones referenciales.

El trabajo descripto en [Viethen, 2011] se encuadra en el área de análisis y modelización estadística de datos de corpora. Entre otros resultados, este trabajo mostró que, en lo que respecta a la GER, distintas personas pueden no hacer lo mismo en la misma situación. De hecho, incluso la misma persona podría describir el mismo target de diferentes maneras en distintas ocasiones. Este capítulo se encuadra en las áreas de diseño y metodologías para la recolección de corpus y su uso para evaluar sistemas de GER.

El corpus ZOOM es una colección de datos, resultante de un experimento realizado en un trabajo conjunto con la Universidad de São Paulo (Brasil). El objetivo del experimento fue crear un corpus de expresiones referenciales de un dominio más complejo y más cercano a las aplicaciones del mundo real, que los corpus existentes hasta el momento. Este corpus contiene referencias a targets singulares y plurales y hace gran uso de propiedades relacionales. Las descripciones del ZOOM corpus fueron producidas por humanos de español y portugués, lo cual permitiría investigar la realización lingüística en estos lenguajes, que actualmente tiene pocos recursos. La mayor parte de los recursos del área están en inglés. Además permitiría investigar la variación entre humanos en la GER ([Fabbrizio et al., 2008; Altamirano et al., 2012; Gatt et al., 2011]). En este Capítulo se describe el dominio y las características del corpus, la metodología usada su recolección y anotación y se lo compara con trabajo previo. Luego haremos una evaluación en la Sección 6.2 de los datos obtenidos. La evaluación realizada motivó al caso de estudio de la Sección 6.3 en la que mostramos tres casos particulares de fragmentos de las ciudades de Lisboa y Madrid. Parte de lo que se presenta en este capítulo fue publicado en [Altamirano et al., 2015], además comparamos nuestra aproximación con trabajo previo sobre el corpus publicado en [Ferreira and Paraboni, 2014].

6.1 Un corpus de descripciones de lugares en mapas

En colaboración con el grupo de investigadores de procesamiento de lenguaje natural de la Universidad de São Paulo Brasil (EACH-USP) diseñamos un experimento en la web para recolectar

descripciones de ubicaciones en mapas. Las descripciones se recolectaron tanto en español como en portugués. El conjunto de datos obtenidos constituye un corpus de expresiones referenciales para investigación en GER y campos relacionados. Este trabajo se describe en [Altamirano et al., 2015].

Las situaciones de referencia en cuestión hacen uso de mapas con dos grados de detalle (representados por los niveles de zoom), e incluyen descripciones singulares y plurales.

6.1.1 Procedimiento de recolección del corpus

Las ERs fueron hechas por personas voluntarias a las cuales se les envió una invitación por correo electrónico o redes sociales. La parte en portugués del corpus tuvo 93 participantes, siendo 66 (71%) hombres y 27 (29%) mujeres. El corpus español tuvo 85 participantes, siendo 59 hombres (69%) y 26 mujeres (30%) como se muestra en la Figura 6.1.

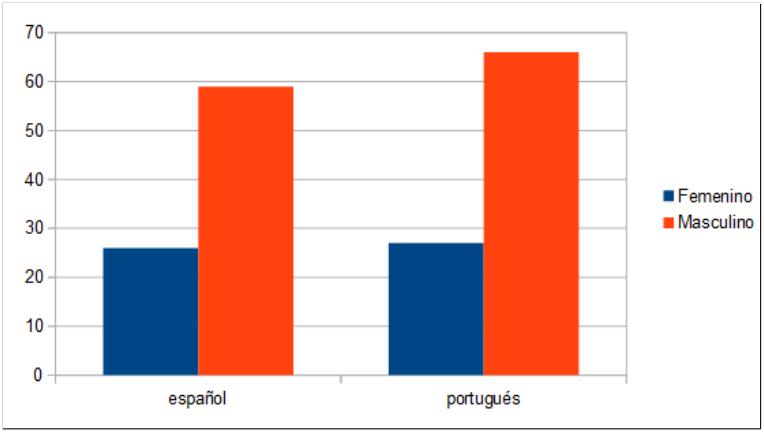


Figura 6.1: Estadística de género de las personas que completaron el experimento.

Las personas voluntarias recibieron un link a la interfaz del experimento, como se muestra en la Figura 6.2. Esa página contenía las instrucciones para el experimento en 3 idiomas: inglés, portugués y español, y un link para completar el experimento en el idioma seleccionado. Las instrucciones solicitaban una ER que permitiera a "un amigo" identificar puntos de interés en mapas de ciudades. La idea es que la persona imaginara que estaba dando sugerencias a un amigo que posiblemente visitara dicha ciudad próximamente.

Lo que se pretendía con esas instrucciones era por un lado que la persona dé una expresión referencial del objeto o los objetos señalados que sea un sintagma nominal, que dicha expresión fuera natural como le daría a un amigo y por otro lado, no incentivar a la gente a hacer ER ni minimales ni sobreespecificadas, es decir, conseguir una expresión natural. El experimento pedía completar la frase "Es interesante visitar..." porque a dicha frase le falta un sintagma nominal para ser una oración gramaticalmente completa. Como las ER se realizan como sintagmas nominales, esa fue la construcción gramatical generada por casi todas las personas que completaron el experimento.

Luego de seleccionar el idioma se mostraba la siguiente página, que recolectaba información demográfica, luego se mostraban los términos y condiciones[1]. Si la persona completaba los datos y aceptaba los términos y condiciones, se comenzaba el experimento mostrando, por ejemplo, el mapa de la Figura 6.3. La página tenía una barra indicadora de progreso, la cual se iba actualizando a medida que el usuario completaba las ERs. Los mapas fueron mostrados en forma aleatoria y al final del experimento se mostraba un mensaje de agradecimiento. Cada mapa mostraba un lugar determinado a ser descripto (por ejemplo, un restaurante, pub, teatro,

[1] *"Acepto que los datos ingresados en este cuestionario sean usados anónimamente para investigación".*

Figura 6.2: Página principal del experimento.

etc.) señalado por una flecha roja como se ve en la Figura 6.3. En esta figura, el lugar señalado por la flecha es un teatro. En el caso de los mapas estímulo de ERs plurales se usaban 2 flechas como se ve en la Figura 6.4, donde los lugares señalados por las flechas son restaurantes. Después de completar la expresión, la persona debía pulsar el botón *Siguiente* y entonces se seleccionaba otro estímulo al azar y así hasta el final del experimento. Las primeras 2 imágenes fueron mostradas sólo para que los participantes se familiarizaran con el entorno del experimento y las respuestas dadas no fueron guardadas.

6.1.2 Materiales utilizados en la recolección

Para la recolección se usaron fragmentos de mapas de las ciudades de Lisboa y Madrid obtenidos desde `openstreetmaps.org`. Openstreetmaps.org es una página que tiene información de mapas de todas partes del mundo, es una organización en la que muchas personas de distintas partes del mundo colaboran para mantener la información actualizada. Esta información es de libre uso, es decir uno puede usar los mapas y sólo hay que nombrar de donde fueron sacados. Se consideraron 2 idiomas: español y portugués. Se obtuvieron ERs para targets singulares como se ve por ejemplo en la Figura 6.3 y para targets plurales como se ve en la Figura 6.4. En los estímulos plurales se usaron referencias del mismo tipo (2 restaurantes, 2 iglesias, etc.) y además se tuvo en cuenta mapas con 2 niveles de zoom. El mapa de la Figura 6.4 muestra un fragmento del mapa de la Figura 6.3 con un mayor nivel de zoom. Como se puede ver en las figuras, los mapas con mayor zoom (a los que describimos como zoom 2X) cubren una porción más chica de la ciudad pero, en general hay más detalle.

Cada mapa contenía iconos de lugares o cosas, calles, nombres de lugares, por ejemplo podemos ver en el mapa de la Figura 6.3 la Calle Mayor, se pueden ver cajeros, restaurantes, fast-foods, café, teléfonos, correos, farmacias, semáforos. Notar que algunos iconos tienen

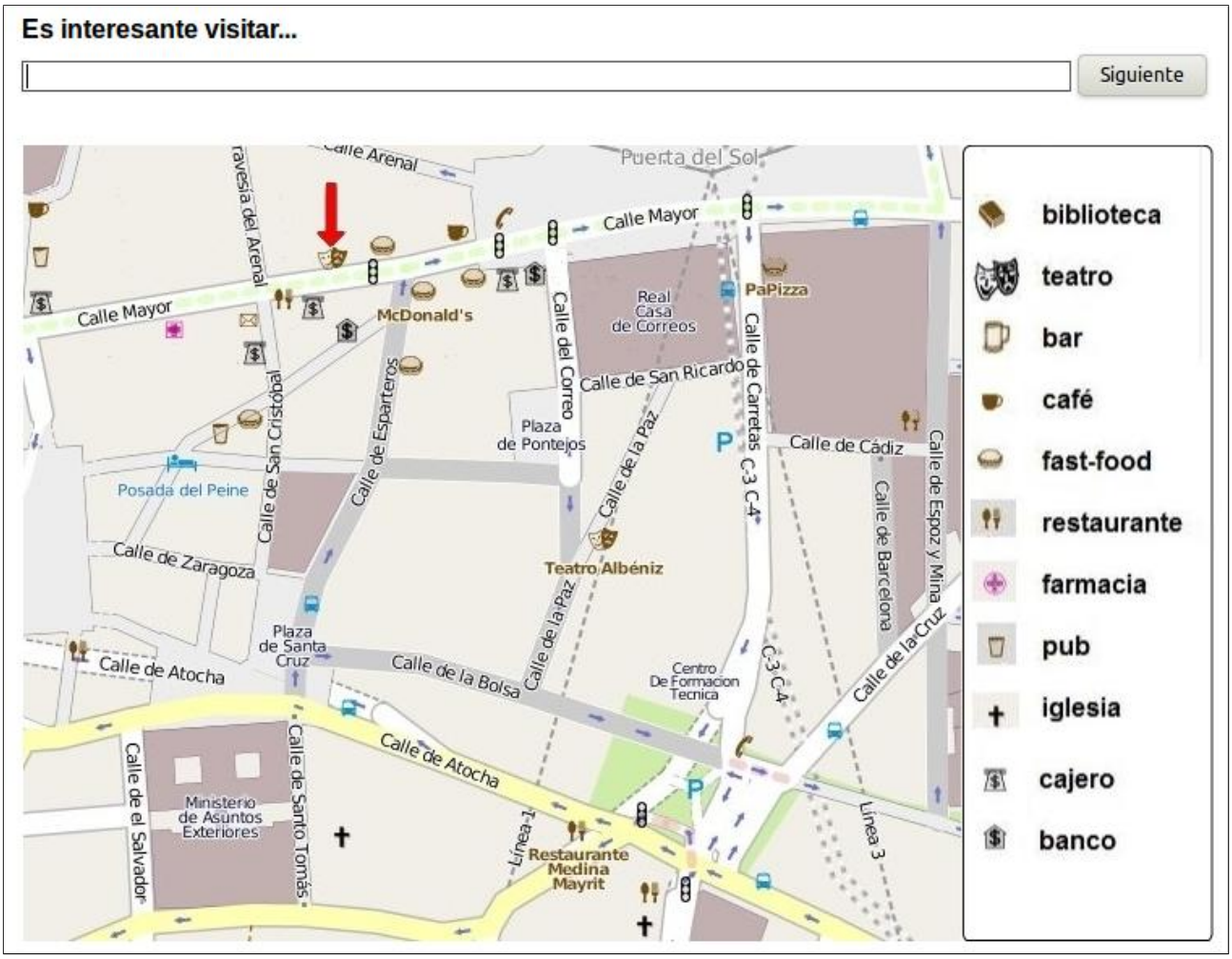


Figura 6.3: Interfaz del experimento.

nombre, por ejemplo el fast-food McDonald's. Algunos edificios o plazas también tienen nombre como la Real Casa de Correos y la Plaza de Pontejos. En el mapa también podemos ver algunos objetos sin nombre. Se puede ver en la Figura 6.3 el nombre del restaurante Medina Mayrit, pero en la Figura 6.4 no se muestra el nombre, se ve solamente el icono. Para el Teatro Alveniz se ve el nombre en ambos niveles de zoom. Notar que no siempre en el nivel 2X de zoom se verá todo lo que se ve y más que en el nivel X, como es el caso del restaurante Medina Mayrit, cuyo nombre se ve en la Figura 6.3 pero no se ve en la Figura 6.4.

Los mapas eran fragmentos del centro de la ciudad de Madrid (para la parte española del corpus) y de Lisboa (para la parte portuguesa). Para cada ciudad, se utilizaron 10 mapas con distintas ubicaciones. Cada ubicación se muestra con un nivel X y 2X de zoom, haciendo 20 imágenes en total. A diferencia de lo que se muestra en esta sección a fines ilustrativos, en el experimento, en ambos niveles de zoom el objetivo señalado se mantuvo. Algunos nombres de las calles y de landmarks podían no aparecer en diferentes niveles de zoom. La mitad de las imágenes mostraron una sola flecha que apuntaba a un objeto en el mapa. Es decir, requerían una descripción como por ejemplo en la Figura 6.3 *el teatro que está en la calle Mayor, cerca de un MacDonald's.* Mientras que la otra mitad mostraron dos flechas que apuntaban a dos lugares diferentes. Y por lo tanto requerían una ER plural, como en Figura 6.4 *los dos restaurantes sobre calle de Atocha cerca del Teatro Calderón.*

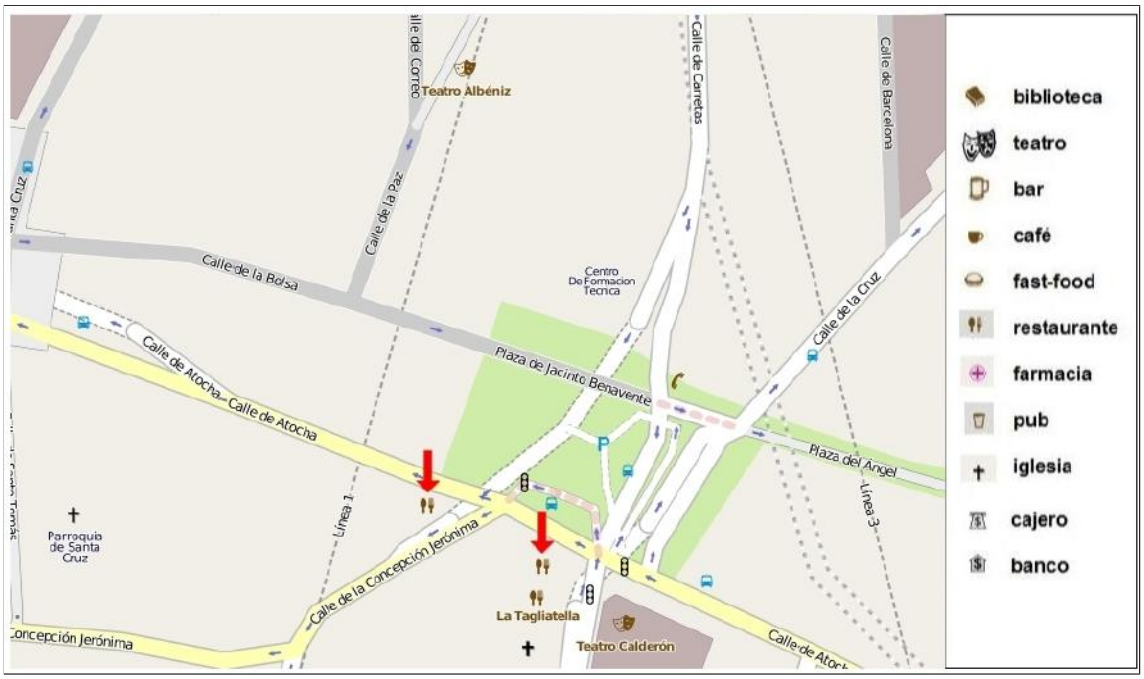


Figura 6.4: Imagen del corpus ZOOM con target plural.

6.1.3 Características de los datos recolectados

La página web del experimento se mantuvo en línea hasta que se obtuvieron 100 experimentos completos para la parte en portugués del corpus y 85 para la parte en español. Luego de la verificación manual, se descartaron 602 descripciones portuguesas mal formadas y 366 descripciones españolas. Así, la parte portuguesa del corpus consta de 1358 descripciones mientras que la parte española contiene 1234 ERs.

Las descripciones mal formadas fueron descartadas siguiendo los siguientes criterios. A pesar de que las personas tenían que completar la frase "Es interesante visitar ..." con un sintagma nominal que describe la ubicación señalada por la flecha (o las flechas, en los casos plurales), algunas de las expresiones no eran frases nominales, sino frases completas, por ejemplo, *"Vamos a ir a pizza express, es realmente barato"*, *"No me gusta la comida rápida"*. Dichas frases fueron descartadas por no ser sintagmas nominales. Del mismo modo, se eliminaron todas las expresiones que describen un objeto que no sea el target previsto y las que utilizaban propiedades que no eran ciertas para el target. La página fue diseñada para obtener el corpus en 3 idiomas (español, portugués e inglés). Hasta el momento de escribir esta tesis se recolectó en dos idiomas: español y portugués. La página sigue en línea, y el código de la misma está disponible para recolectar un corpus similar[2].

En la parte portuguesa de los datos, el 78,6% de las descripciones incluyen propiedades relacionales. Además de eso, el 36,4% eran minimales un 44,3% eran sobreespecificadas, y el 19,3% eran subespecificadas. En la parte española, 70% de las descripciones incluyeron propiedades relacionales. Por otra parte, el 35% eran minimales, el 40% eran sobreespecificadas, y el 25% eran subespecificadas. Esta proporción tan grande de descripciones subespecificadas así como descripciones relacionales no son comunes en corpora de GER existente (o por lo menos en tal proporción), esto puede reflejar la complejidad del dominio o limitaciones en el entorno web en el cual se basó la obtención del corpus. Esto se discute en la Sección 6.1.4.

[2]http://cs.famaf.unc.edu.ar/∼romina/pagina/

Figura 6.5: Mapa con identificadores usado para la anotación.

6.1.4 Anotación del corpus

Para cada mapa se generó una tabla con los identificadores únicos de los objetos visibles. Por ejemplo, para la imagen de la Figura 6.5, la tabla correspondiente es la Tabla 6.1. Los identificadores se generaron usando las 3 o 4 primeras letras del tipo y un número correlativo dentro de ese tipo. Por ejemplo *pub1* es el que está siendo apuntado por la fecha roja en la Figura 6.5, o sea es el target. A cada calle visible en el mapa también se le asignó un identificador único, como se ve en la Tabla 6.1 desde la posición 36 en adelante. Dicha tabla tiene 4 columnas para cada objeto. El tipo es la representación semántica del sustantivo que describe al objeto, por ejemplo, restaurante, bar, street. El nombre es el nombre propio del objeto, si tenía, (se mantuvieron en el idioma original, español o portugués). "en", contenía la calle en la que estaba situado el objeto, podía estar vacío si no estaba en una calle. Y el id que fue compuesto por las 3 o 4 primeras letras del tipo del objeto más un número para identificarlo unívocamente, por ejemplo para los restaurantes "rest1", "rest2".

Se seleccionó un conjunto fijo de atributos para cada objeto, pero dando flexibilidad con un atributo llamado "other" que permitía anotar otras cosas que hayan aparecido en la expresión dada por la persona, que no estuviera en la lista fija. En esta manera de anotar, cada objeto tiene como máximo 26 posibles atributos. En el caso de las descripciones plurales, el conjunto de atributos se repite para cada objeto, por lo que el anotador podía utilizar hasta 52 atributos.

Los 26 o 52 atributos nombrados anteriormente están formados por: 10 atributos que denotan características del objeto target, o relaciones del target con otros objetos y se detallan a continuación:

- tipo como por ejemplo, restaurant
- nombre propio como por ejemplo, McDonalds
- en calle como por ejemplo si el target está en la *Calle Mayor* esta relación binaria entre el target y la calle tendría el identificador str1
- izquierda como por ejemplo en la Figura 6.5 pub1 está a la izquierda de de fast5. En este caso, si el target fuera fast5, estaría relacionado con la relación izquierda con pub1
- derecha como por ejemplo, en la Figura 6.5 fast5 está a la derecha de de pub1
- en esquina, esta es una relación ternaria, es decir entre 3 objetos, el target y 2 calles. Esta clase de relaciones se ha modelado como 2 relaciones binarias
- cerca como por ejemplo pub1 está cerca de fast5
- en-frente-de como por ejemplo atm2 está al frente de thea2
- detrás como por ejemplo atm2 está detrás de ban1
- otro cualquier otra cosa mencionada en la descripción, podía ser una propiedad o relación. Por ejemplo un color. El atm2 es azul.

Número	Tipo	Nombre propio	En calle	ID
1	cafe			cafe2
2	pub			pub2
3	atm		str1	atm4
4	drugstore		str1	drug1
5	snailpost		str1,str2	snai1
6	restaurant	el museo del jamon	str1,str2	rest1
7	atm		str1	atm2
8	theater		str1	thea2
9	semaphore		str1	sem0
10	semaphore		str1	sem1
11	semaphore		str1	sem2
12	semaphore		str1,str3	sem3
13	fast-food	maoz	str1	fast1
14	fast-food	mcdonalds	str1,str4	fast2
15	fast-food	kfc	str1	fast3
16	cafe		str1	cafe1
17	fast-food	burger king	str4	fast6
18	atm		str1	atm3
19	pub		str1,str3	ban3
20	fast-food	pans and company	str2	fast5
21	**pub**		**str15**	**pub1**
22	hotel	posada del peine	str15,str19	hot1
23	bank	banco popular	str15	ban1
24	semaphore		str1,str5	rest5
25	fast-food	papizza	str5	fast4
26	restaurant		str6,str7	rest5
27	theater		str8	thea1
28	semaphore		str9	sem5
29	semaphore		str9	sem6
30	restaurant		str9	rest2
31	church		str9,str10	chur1
32	restaurant	medina mayrit	str9	rest3
33	restaurant		str9	rest4
34	church		str11	chur2
35	bank	bankinter	str15	ban0
36	street	calle mayor		str1
37	street	calle de san cristobal		str2
38	street	calle del correo		str3
39	street	calle de esparteros		str4
40	street	calle de carretas		str5
41	street	calle de espoz y mina		str6
42	street	calle de cadiz		str7
43	street	calle de la paz		str8
44	street	calle de atocha		str9
45	street	calle de santo tomas		str10
46	street	calle de la cruz		str11
47	street	calle del salvador		str12
48	street	calle de la bolsa		str13
49	street	calle de zaragoza		str14
50	street	calle de las postas		str15
51	street	calle botoneras		str16
52	street	calle de gerona		str17
53	street	calle de la sal		str18
54	street	calle del marquez viudo de pontejos		str19

Tabla 6.1: Objetos identificados de la Figura 6.5 indicando su tipo, su nombre propio (en caso de estar visible), calle (o calles) en las que se encuentra y un identificador único. El target se resalta en negrita en la posición número 21.

```
<TRIAL ID="2" SPEAKER="166" AGE="18" GENDER="m">
  <CONTEXT ID="3" SEQ="1300">
     <ATTRIBUTE-SET TARGET="pub1" LANDMARK="fast5"
                    STRING="El pub que est\'a al lado de Pans & Company,
                            en la calle de las Postas">
        <ATTRIBUTE NAME="type" VALUE="pub" />
        <ATTRIBUTE NAME="in" VALUE="str15" />
        <ATTRIBUTE NAME="landmark-name" VALUE="Pans & Company" />
     </ATTRIBUTE-SET>
  </CONTEXT>
</TRIAL>
```

Figura 6.6: Fragmento de XML de la ER "El pub que está al lado de Pans & Company, en la calle de las Postas".

Los restantes 16 atributos denotan propiedades de los objetos adicionales (landmarks) que se mencionaron en la descripción, a los cuales llamamos d1.. d4. Por ejemplo, en *El pub que está al lado de Pans & Company, en la calle de las Postas*) tenemos: target = pub, d1 = fast-food (Pans & Company).

Para cada target se consideró un máximo 4 objetos tomados como puntos de referencia (landmarks), ya que con esta cantidad cubríamos las descripciones más largas, estos landmarks se anotaron como d1..d4, para cada landmark se anotaron 4 atributos:

- Identificación (id)
- Tipo (un sustantivo, restaurant, pub...)
- Nombre (Posada del Peine)
- Otro (alguna otra cosa dicha sobre el objeto)

A su vez cada atributo tiene una lista estricta de los valores permitidos. Por ejemplo, para el atributo tipo los posibles valores son biblioteca, teatro, bar, café, fast-food, etc. También se permite el valor "otro" que le da al anotador una oportunidad de dar una respuesta cuando la descripción muestra algo muy inusual.

Para los atributos espaciales (izquierda, cerca, etc.) permitimos que incluyan la mayoría de los objetos en el mapa, a excepción de aquellos que son claramente falsos. Por ejemplo, en el caso del atributo "izquierda" sus posibles valores incluyen la mayoría de los objetos en el mapa de la Figura 6.5, a excepción de los de la derecha del objeto target. Por ejemplo, es posible decir *el pub que está a la izquierda de Pans & Company*, pero no es admitido *el pub que está a la izquierda de Posada del Peine.*

Las descripciones recolectadas fueron anotadas por dos anotadores independientes. Luego, un tercer anotador asumió el rol de juez y dio la anotación final. El acuerdo entre los jueces, se midió con el coeficiente Kappa [Cohen, 1960], éste fue del 84% en el nivel de atributo. Ambos, el contexto y las ERs se representaron en formato XML utilizando una versión adaptada del formato XML adoptado en el corpus TUNA [Gatt et al., 2007]. Las ERs se agruparon en nodos TRIAL, que como se muestra en la Figura 6.6, que contienen información general sobre cada persona (es decir, identificación, edad y sexo), seguido de su lista de respuestas. Cada respuesta identifica el contexto en que se produjo, y la ER que la persona dijo. Al igual que en [Gatt et al., 2007], el contenido de las ERs está representado por nodos que son conjunto de atributos que contienen una lista de nombres de atributos y sus valores. La Figura 6.6 ilustra la representación para la ER *el pub que está al lado de Pans & Company, en la calle de las Postas.*

En cada TRIAL tenemos la información de la identificación del TRIAL, de la persona, edad y género. En ATTRIBUTE-SET se encuentran los datos de la ER, TARGET y LANDMARKS

con sus id's, STRING es la expresión completa tal cual la dio la persona, y los atributos del target que en este ejemplo son su tipo pub, y su ubicación en la calle str15, en este caso hubo 1 sólo landmark. En casos donde había más de 1 landmark, los landmarks siguientes fueron etiquetados como "second-landmark" y así sucesivamente. Esto fue motivado por la necesidad de proporcionar nombres de atributos únicos para el beneficio de los algoritmos de GER. El conjunto de mapas, textos descriptivos y sus representaciones XML constituye el corpus ZOOM de ERs, el cual es público para fines de investigación. En lo que sigue presentamos una comparación de este corpus con corpora de ERs existentes hasta el momento.

6.1.5 Comparación con trabajo previo

En la Tabla 6.2 se presenta una comparación entre el corpus ZOOM y otros corpus de ERs existentes descripto en el Capítulo 2. La información de dominio representa el número de posibles atributos atómicos más el número de relaciones en cada descripción (columna Atributos). La información sobre el TUNA-corpus y las descripciones de ZOOM se basan sólo en la parte singular de cada corpus. La cantidad máxima de landmarks permitidos (en columna Landmarks). También se representa el tamaño medio de la descripción, es decir en número de propiedades anotadas, (en la columna tamaño promedio). La columna Uso representa la utilización de las propiedades, la cual se toma como la proporción de las propiedades que aparecen en la descripción sobre el número total de posibles atributos y puntos de referencia. Desde una perspectiva de GER, las ERs de mayor largo y las puntuaciones más bajas de uso representan las situaciones más complejas de referencia. Estas dos características combinadas muestran que el corpus ZOOM es el más complejo disponible en el área. Los atributos del corpus ZOOM son 19 ya que no tienen en cuenta para la comparación los identificadores, ni el atributo other, por ser genérico y tener información variada.

Corpus	Atributos	Landmarks	Tamaño promedio	Uso
TUNA-Furniture	4	0	3.1	0.8
TUNA-People	10	0	3.1	0.3
GRE3D3	9	1	3.4	0.3
GRE3D7	6	1	3.0	0.4
Stars	8	2	4.4	0.4
Stars2	9	2	3.3	0.3
Zoom-Pt	19	4	6.7	0.3
Zoom-Sp	19	4	7.2	0.3

Tabla 6.2: Comparación con corpora de ERs existentes.

Los corpora de expresiones referenciales existentes se caracterizan por su diseño y recolección en experimentos muy controlados. La recopilación de corpus de un dominio que es directamente relevante para aplicaciones del mundo real plantea un reto importante no sólo para los algoritmos de GER existentes, sino también para el proceso de recolección y anotación en sí. Esto se hace evidente por la gran proporción de las expresiones referenciales que deben ser desechadas desde nuestra base de datos porque no constituyen una ER para el objeto target. Esta proporción es del 30% de la parte portuguesa del corpus y el 23% de la parte española del corpus. Después de este proceso de limpieza, la parte portuguesa del corpus contiene todavía un 19,3% de expresiones referenciales subespecificadas y la parte española un 25%. El porcentaje de ERs subespecificadas en el corpus ZOOM es mucho mayor que lo reportado en los corpora de ERs descriptos en el la Sección 2.1.3, que no es mayor que el 5%.

Estudios psicolingüísticos [Clark and Wilkes-Gibbs, 1986] han encontrado que más del 20% de las ERs son subespecificadas con respecto al contexto establecido. En dominios relativamente complejos, se encontró que los hablantes a menudo producen una ER subespecificada inicial,

y luego agregan más información si es necesario, en lugar de producir una descripción que identifica al target de forma unívoca desde el principio. El dominio del ZOOM corpus es probable que contenga situaciones más complejas de referencia que los corpora existentes de este tipo. Es decir, las descripciones del ZOOM son en promedio más largas, y tienen un promedio de uso de atributos más bajo. Esto en nuestra opinión, puede explicar la alta proporción de subespecificación en nuestros datos. La proporción de descripciones relacionales también es mayor que en el trabajo anterior. Las descripciones relacionales constituyen un desafío para los algoritmos de GER debido a la complejidad computacional asociada a su generación [Krahmer and van Deemter, 2012]. Otro desafío que plantea el corpus ZOOM es el hecho de que contiene dos descripciones para cada objeto target basándose en diferentes modelos correspondientes a la misma ubicación en el mapa visto con diferentes niveles de zoom. Por ejemplo, un mapa

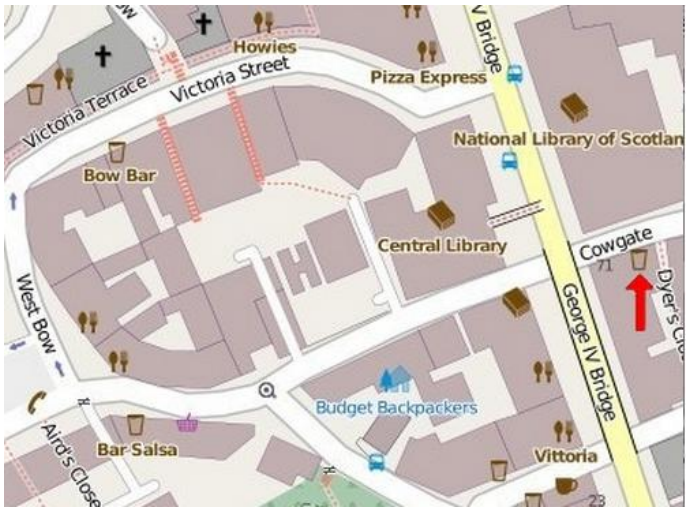


Figura 6.7: Mapa con nivel de zoom X.

con mayor nivel de zoom (2X) se ilustra en la Figura 6.8, y el mismo mapa con un menor nivel de zoom se muestra en la Figura 6.7.

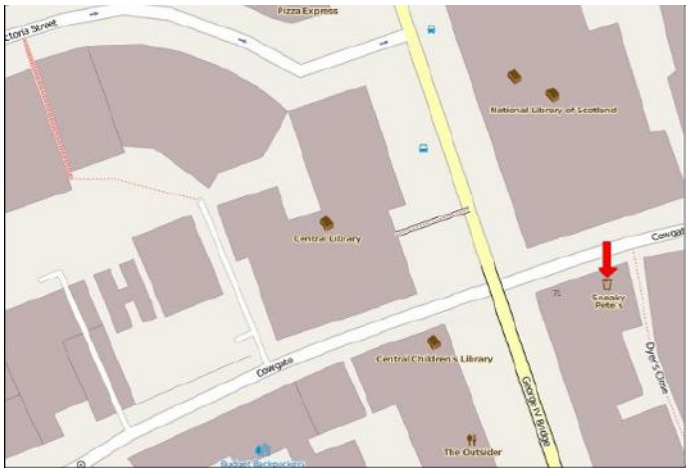


Figura 6.8: Mapa con nivel de zoom más detallado (2X).

Los modelos subyacentes de estos dos mapas son diferentes, pero están relacionados. El mapa con zoom 2X contiene menos elementos, pero con más propiedades debido al mayor nivel de detalle. Una ER para el target en el mapa 1X puede o no ser útil en el mapa 2X. Por ejemplo, la expresión referencial "el pub en Cowgate" es subespecificada en el mapa 1X, pero es minimal en el mapa 2X. Cambios de este tipo son comunes en aplicaciones interactivas: cambios en la

estructura del contexto de referencia durante una interacción, en el número de objetos y en las propiedades a las cuales hay que referirse. El desafío para los algoritmos de GER sería producir una descripción apropiada para el contexto modificado sin empezar desde cero. Estos temas se discuten en mayor detalle en el Capítulo 7.

6.2 Desempeño de otros algoritmos sobre el corpus

En esta sección se presentan los resultados del desempeño de un algoritmo simbólico conocido como el algoritmo incremental descripto en la Sección 2.2.1 y un algoritmo puramente estadístico basado en *support vector machines* (SVM). Estos algoritmos se evalúan sobre un fragmento del corpus ZOOM para poner en perspectiva los resultados de nuestros algoritmos que serán presentados en la Sección 6.3.

6.2.1 Algoritmos y procedimiento

Para esta evaluación publicada en [Altamirano et al., 2015] se utilizó un subconjunto de descripciones singulares de la parte portuguesa del corpus. Esto comprende 821 descripciones producidas por personas para 9 escenas. La evaluación se llevó a cabo mediante la comparación de las ERs del corpus con la salida del sistema para medir la exactitud global (es decir, el número de coincidencias exactas entre las dos descripciones), los coeficientes Dice [Dice, 1945] y MASI [Passonneau, 2006] (es decir, el grado de solapamiento entre la salida del algoritmo y la descripción del corpus correspondiente).

El algoritmo SVM utilizado se basa en un diseño propuesto en [Ferreira and Paraboni, 2014]. Es decir se construye un modelo de GER usando *clasificadores support vector machines* (SVM) con función de base kernel radial. Los clasificadores se entrenaron y se hicieron validaciones cruzadas con *6 folders*. Para conseguir los parámetros óptimos se realizó el procedimiento estándar de k-folders. Es decir, para cada validación cruzada, se reservó un conjunto para las pruebas, y los restantes $k-1$ conjuntos quedaron sujetos a un segundo procedimiento de validación cruzada en el que se probaban diferentes combinaciones de parámetros. Al parámetro C se le asignaron los valores 1, 10, 100 y 1000, y a γ se le asignaron los valores 1, 0.1, 0.001 y 0.0001. El conjunto de parámetros de mejor desempeño fue seleccionado para construir un clasificador entrenado con los conjuntos $k-1$, y probado en los datos de prueba. Este procedimiento se repitió para cada iteración del procedimiento principal de validación cruzada.

El modelo SVM consta de 12 clasificadores binarios que representan si los atributos individuales deben ser seleccionados para su inclusión en la ER de salida. Los clasificadores se corresponden con atributos atómicos del target y del primer landmark con los atributos (*tipo*, *nombre* y *otros*), y las relaciones. Los atributos de otros landmarks no se modelaron en este experimento. El valor múltiple de la relación *entre* también se ignoró, y la relación esquina que implica a dos puntos de referencia (por ejemplo, dos calles) se modeló como dos tareas de clasificación independientes. Dos características de aprendizaje fueron consideradas por cada clasificador:*landmarkCount*, que representa el número de objetos cercanos al target, y *DistractorCount*, que representa el número de objetos del mismo tipo que el target en el contexto considerado.

A partir de los resultados de los 12 clasificadores binarios, una descripción se construyó teniendo en cuenta los atributos atómicos del target en primer lugar. Por cada predicción positiva, se seleccionó el atributo atómico correspondiente para su inclusión en la descripción de salida. A continuación, se consideraron las relaciones. Si no se elije ninguna relación, el algoritmo termina devolviendo una descripción atómica del objeto target. Si la descripción incluye una relación, se selecciona el objeto landmark relacionado, y el algoritmo se llama de forma recursiva para describir a ese objeto.

Puesto que cada atributo que se corresponde con una predicción positiva será siempre seleccionado, el algoritmo no considera unicidad como una condición de parada. Como resultado, la descripción de salida puede transmitir una cierta cantidad de sobreespecificación o bien ser subespecificada.

6.2.2 Resultados de la generación

Tabla 6.3 resume los resultados obtenidos por el algoritmo de GER construido a partir de clasificadores SVM y los obtenidos mediante un sistema descripto en el Capítulo 2 que implementa una extensión relacional del algoritmo incremental [Dale and Reiter, 1995]. Se comparan exactitud, coeficientes Dice y MASI.

Algoritmo	Exactitud	Dice	MASI
SVM	0.15	0.51	0.28
Incremental	0.04	0.53	0.21

Tabla 6.3: Resultados de la GER.

En lo que sigue se comparan los resultados de exactitud obtenidos por cada par algoritmos mediante la prueba de chi cuadrado, y se compara Dice utilizando la prueba de rangos con signo *de Wilcoxon*.

En términos de exactitud global y también en cuanto a las puntuaciones MASI, el método SVM supera al algoritmo incremental. La diferencia con el segundo algoritmo de mejor rendimiento (es decir, el enfoque incremental) es altamente significativa (χ^2 = 79,87, df = 1, p< 0,0001). Sólo en cuanto a las puntuaciones de Dice se observa el efecto contrario (T = 137570.5, p=0,01413).

También se evaluó el desempeño de los clasificadores individuales. La Tabla 6.4 muestra estos resultados, medidos por la exactitud (E), cobertura (R), F1-medida (F_1)y el área bajo la curva ROC (AUC). F1 es la media armónica entre la exactitud y la cobertura. La curva ROC (AUC) es la representación de la proporción de verdaderos positivos (VPR = proporción de Verdaderos Positivos) frente a la proporción de falsos positivos (FPR = proporción de Falsos Positivos) también según se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo).

Clasificador	E	R	F_1	AUC
tg_type	0.95	1.00	0.98	0.25
tg_name	0.09	0.05	0.07	0.41
tg_other	0.00	0.00	0.00	0.05
lm_type	0.93	1.00	0.96	0.44
lm_name	0.97	1.00	0.98	0.35
lm_other	0.00	0.00	0.00	0.43
next-to	0.50	0.24	0.32	0.63
right-of	0.00	0.00	0.00	0.28
left-of	0.00	0.00	0.00	0.27
in-front-of	0.00	0.00	0.00	0.42
behind-of	0.00	0.00	0.00	0.17
in/on/at	0.60	0.60	0.60	0.61

Tabla 6.4: Resultados del clasificador a nivel atributo.

A partir de estos resultados se ve que algunos atributos (por ejemplo, el tipo del target y del landmark y el nombre del landmark) se clasificaron con gran exactitud, mientras que otros (por ejemplo, las relaciones) no se clasificaron con alta exactitud.

Cabe destacar que aunque en estas métricas, el algoritmo SVM supera al incremental, el SVM no puede asegurar que su salida sea una ER dado que no usa el modelo para verificar si

la descripción identifica unívocamente al target, mientras que el algoritmo incremental sí. El algoritmo SVM podría verse entonces como una posible implementación de la fase "generar" del modelo cognitivo de Keysar [Keysar et al., 1998] discutido en el Capítulo 2, a la que le falta la fase de "ajustar" para poder asegurar unicidad. Como tal genera ERs de una forma **egocéntrica** (en terminología de Keysar et al.) como lo haría un niño. Nuestro algoritmo en cambio, implementa las dos fases del modelo cognitivo de Keysar et al.

A continuación evaluamos el desempeño de nuestro algoritmo sobre mapas del corpus ZOOM.

6.3 Desempeño de nuestros algoritmos sobre el corpus

En esta sección presentamos tres casos de estudio detallados de evaluación de las ERs generadas con nuestros algoritmos para mapas del corpus ZOOM. En la Sección 6.3.1 analizaremos la generación de rankings de ERs singulares en el dominio del corpus ZOOM y comparamos el desempeño del algoritmo con datos del corpus. En la Sección 6.3.2 analizamos la generación de rankings de ERs en un mapa con mayor nivel de zoom que el analizado en la Sección 6.3.1. En la Sección 6.3.3 analizamos la generación de ERs plurales.

6.3.1 Generación de expresiones referenciales singulares

Consideremos el mapa de la Figura 6.9 que muestra una región de la ciudad de Madrid. El target, es decir, el objeto señalado por la flecha roja, es una iglesia, como se puede ver en la leyenda ubicada a la derecha del mapa.

Figura 6.9: Imagen del corpus ZOOM con target singular. Se muestran las probabilidades de uso de las palabras en la esquina inferior derecha.

En la parte inferior derecha de la imagen se muestran las probabilidades de uso, las cuales fueron aprendidas desde el corpus, como se describe en el Capítulo 4. Estas probabilidades no estaban en la imagen mostrada a los participantes que completaron el experimento de recolección de corpus, se agregaron a la imagen para ilustrar la discusión de esta sección.

En la Tabla 6.5 se muestran las 10 expresiones referenciales más frecuentes generadas por las personas que participaron en la recolección del corpus para el mapa de la Figura 6.9. Se indica el porcentaje de personas que dieron cada tipo de ER, es decir la cantidad de distintas personas que independientemente decidieron usar la misma ER[3] para referirse al target, dividido por la cantidad total de ERs del mapa (se da en porcentaje). Estas 10 ERs más frecuentes acumulan el 78.5% del total de las ERs del corpus para el mapa considerado. También se indica sin son subespecificadas (SubE) o sobreespecificadas (SobreE). Notar que las que no son ni subespecificadas, ni sobreespecificadas, son minimales. Para ejecutar el algoritmo necesitamos definir el vocabulario del modelo, es decir las palabras que vamos a usar al correr el algoritmo, la semántica del mismo. Estas palabras fueron sacadas de la anotación del corpus, el cual está anotado en inglés.

	ERs del corpus ZOOM para el target *church1*	%	SubE	SobreE
1	Iglesia	19.7%	X	
2	Iglesia en la calle de Santo Tomás	18.5%		
3	Iglesia en la calle de Santo Tomás frente al Ministerio de Asuntos Exteriores	13.6%		X
4	Iglesia frente al Ministerio de Asuntos Exteriores	13%		
5	Iglesia en la calle Santo Tomás cerca del Ministerio de Asuntos Exteriores	2.5%		X
6	Iglesia en la calle de Atocha	2.5%		
7	En calle de Santo Tomás	2.5%	X	
8	Iglesia que está en la calle de Santo Tomás y Atocha, frente al Ministerio de Asuntos Exteriores	2.5%		X
9	Iglesia que está en la calle de Santo Tomás y Atocha	2.5%		X
10	Iglesia cerca del Ministerio de Asuntos Exteriores	1.2%		

Tabla 6.5: Las 10 ER más frecuentes del corpus para el mapa de la Figura 6.9.

Para calcular las probabilidades de uso usamos el método presentado en el Capítulo 4, cuando se tiene un corpus disponible para la escena considerada. Se tuvieron en cuenta las ERs que dieron las personas, incluso las que eran subespecificadas, pero no las que no eran ciertas, un total de 63 ERs fueron las que cumplían con esas condiciones. Para cada palabra de las que aparecía en el corpus para el contexto seleccionado, las cuales son: {church, in, front, near, minAsExt, calleSantoTomas, calleAtocha} se contó la cantidad de veces que aparecía la palabra en el corpus, teniendo en cuenta que si aparecía 2 veces en la misma ER se contaba sólo 1 vez, a fin de tener siempre probabilidades entre 0 y 1, y luego se dividió sobre 63 que fueron la cantidad de ERs que eran válidas. Por ejemplo la palabra *church* apareció en 61 ERs, por lo tanto la probabilidad de *church* es $61/63 = 0.97$. Y así se calculó para las demás palabras del vocabulario. Las probabilidades de uso de las palabras se muestran en la esquina inferior derecha del mapa de la Figura 6.9. En la Figura 6.10 se muestra el modelo relacional que usamos como input del algoritmo. A fines ilustrativos, el modelo representa sólo la parte del mapa que está al sur de la calle Atocha.

La iglesia target se representa como *church1* y la iglesia distractora como *church2*. La iglesia target se encuentra *near* y *front* a el *minAsExt* el cual tiene identificador *min1*. Está *in calleAtocha* e *in calleSantoTomas*. El objeto *d*, representa la construcción no identificada en el mapa que está al frente de *church2*. Recordemos que el algoritmo es no-determinístico, así que si lo ejecutamos muchas veces, vamos a conseguir posiblemente diferentes ERs para el mismo input. Teniendo así como resultado un ranking de ERs. Ejecutamos el algoritmo 1000

[3]Por la misma ER nos referimos a una ER con la misma semántica. Posibles divergencias gramaticales o léxicas en la realización fueron unificadas.

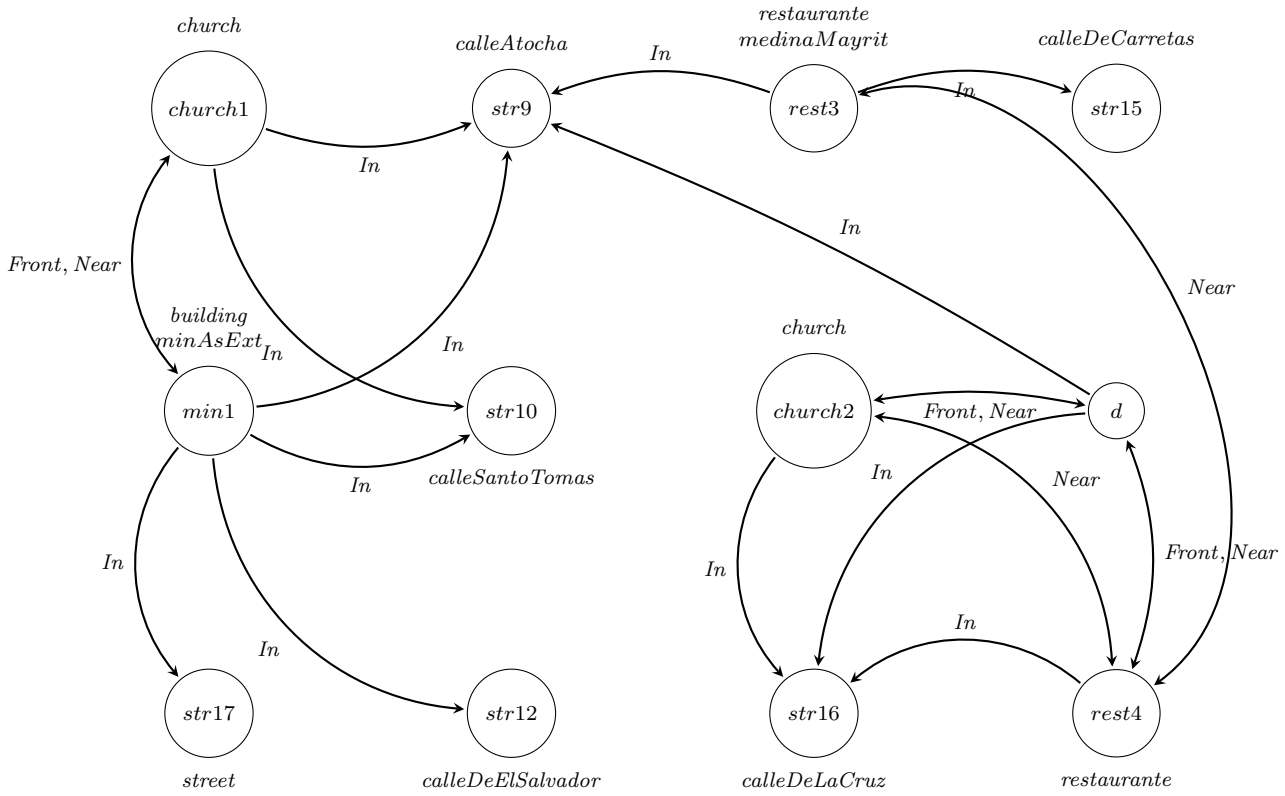


Figura 6.10: Modelo relacional del mapa de la Figura 6.9.

veces, con input el modelo mostrado en la Figura 6.10 y las probabilidades dadas en la esquina inferior derecha de la misma figura. Se muestran las posibles realizaciones de las 10 fórmulas más frecuentes generadas para el target *church1*, y la frecuencia dada por el algoritmo en la Tabla 6.6.

	ER generadas por el algoritmo para el target *church1*	#	%
1	Iglesia en la calle de Santo Tomás	308	30.8
2	La que está frente al Ministerio de Asuntos Exteriores	223	22.3
3	Iglesia en la calle de Atocha	87	8.7
4	La que está cerca al Ministerio de Asuntos Exteriores	94	9.4
5	Iglesia frente al Ministerio de Asuntos Exteriores	136	13.6
6	Iglesia que está en la calle de Santo Tomás y que tiene algo al frente	53	5.3
7	Iglesia que tiene cerca algo que está en la calle Santo Tomás	13	1.3
8	Iglesia que está al frente de algo que está en la calle Santo Tomás	11	1.1
9	Iglesia cerca del Ministerio de Asuntos Exteriores	6	0.6
10	En calle de Atocha y que está al frente del Ministerio de Asuntos Exteriores	1	0.1

Tabla 6.6: Posibles realizaciones de las 10 ER más frecuentes dadas por el algoritmo para el modelo de la Figura 6.10, para el target *church1*.

Notar que de los 10 tipos diferentes de ERs que más dieron las personas (que se muestran en la Tabla 6.5), 2 son subespecificadas. Al ser subespecificadas, el algoritmo no las generó.

La ER que identifica unívocamente más frecuente dada por las personas, también fue la más frecuente dada por el algoritmo como se muestra en la Tabla 6.7. Obtenemos una exactitud (E) total de 34.6% como se muestra en la Tabla 6.7.

	ERs para el target *church1*	Frec.	Frec-Alg	E
1	Iglesia	19.7%	-	-
2	Iglesia en la calle de Santo Tomás	18.5%	30.8%	18.5%
3	Iglesia en calle de Santo Tomás frente al Ministerio de Asuntos Exteriores	13.6%	0%	0%
4	Iglesia frente al Ministerio de Asuntos Exteriores	13%	13.6%	13%
5	Iglesia en calle Santo Tomás cerca del Ministerio de Asuntos Exteriores	2.5%	0%	0%
6	Iglesia en la calle de Atocha	2.5%	8.7%	2.5%
7	En calle de Santo Tomás	2.5%	-	-
8	Iglesia que está en la calle de Santo Tomás y Atocha, frente al Ministerio de Asuntos Exteriores	2.5%	0%	0%
9	La iglesia que está en la calle de Santo Tomás y Atocha	2.5%	0%	0%
10	Iglesia cerca del Ministerio de Asuntos Exteriores	1.2%	0.6%	0.6%
	Totales	69.5%	53.7%	34.6%

Tabla 6.7: Comparación de las 10 ERs más frecuentes del corpus y las dadas por el algoritmo.

Consideramos que el algoritmo no generó la ER *la iglesia que está en la calle de Santo Tomás y Atocha, frente al Ministerio de Asuntos Exteriores* por ser ésta una ER muy sobreespecificada. Nuestro algoritmo termina antes, consiguiendo una ER y no sobreespecifica lo suficiente como para dar esta ER (recordemos que sobreespecifica sólo en el primer ciclo). De forma similar, la ER *la iglesia en la calle de Santo Tomás frente al Ministerio de Asuntos Exteriores* se genera con baja frecuencia porque la probabilidad que tiene el algoritmo de elegir todas esas propiedades a la vez es muy baja. Otra ER que el algoritmo no generó es *la iglesia que está en la calle de Santo Tomás y Atocha*. Esta ER no está tan sobreespecificada pero *calleAtocha* tiene una probabilidad de uso muy baja de 0.06.

6.3.2 Generación de expresiones referenciales en mapas con zoom

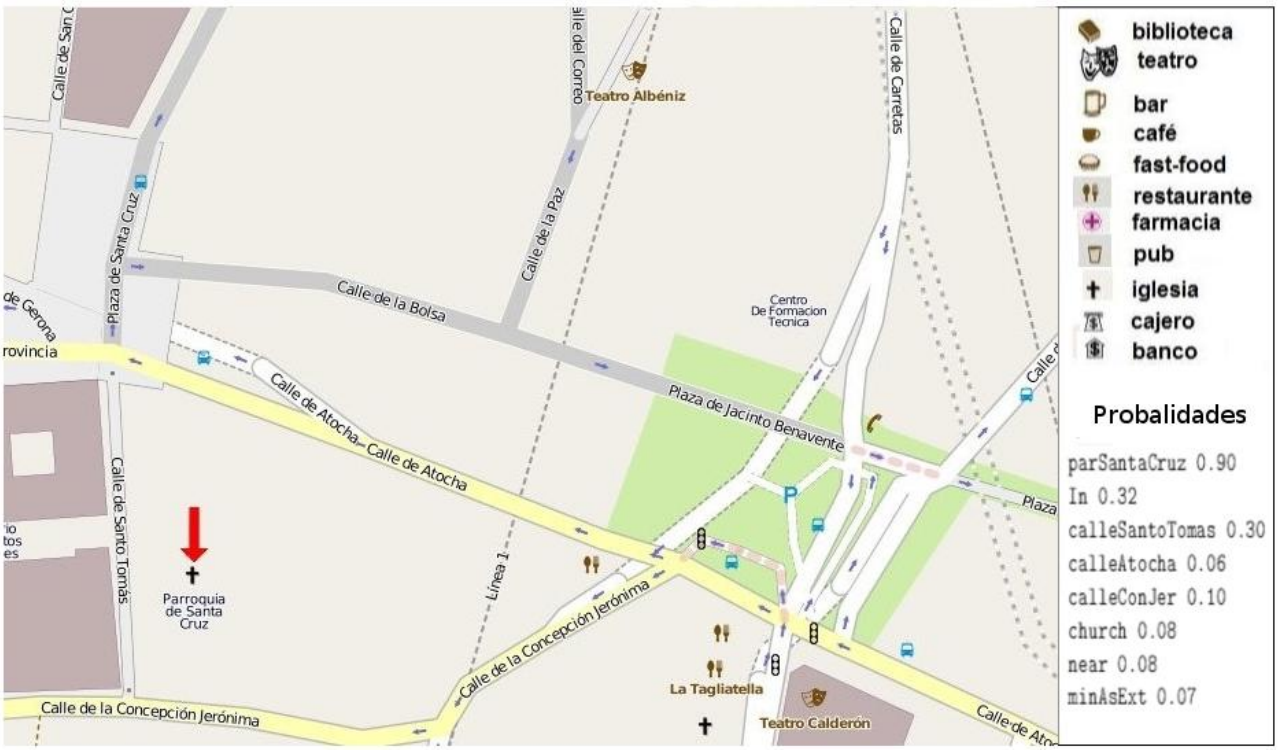


Figura 6.11: Imagen del corpus ZOOM con target singular. Este mapa muestra un fragmento de la zona de el mapa de la Figura 6.9 pero con mayor nivel de zoom. Se muestran las probabilidades de uso de las palabras en la esquina inferior derecha.

El mapa que analizamos en esta sección es el que se muestra en la Figura 6.11. Este mapa tiene un mayor nivel de zoom que el de la Figura 6.9, y por lo tanto muestra una porción más pequeña de la zona. Es decir por un lado, se ven menos objetos, pero por otro lado, lo que se ve tiene en general más detalle, por ejemplo, se ve el nombre de la iglesia: "Parroquia de Santa Cruz" que no se veía en la Figura 6.9. Otra cosa que se puede ver en el mapa con zoom es el nombre de la calle: "calle de la Concepción Jerónima" que está a la derecha de la Parroquia. En este mapa no se puede ver el nombre del Ministerio de Asuntos Exteriores porque queda cortado en la imagen, el mismo sí se veía en el mapa de la Figura 6.9. Lo único que se ve en la calle de Santo Tomás es la iglesia target. Estas diferencias se van a ver reflejadas en el modelo que toma como input el algoritmo. En la Tabla 6.8 se muestran las 12 ERs distintas dadas por los participantes que completaron el experimento en la recolección del corpus, y la cantidad de ocurrencias en el corpus. El porcentaje que representa, y si es subespecificada (SubE) o sobreespecificada (SobreE). Como dijimos anteriormente las que no son ni subespecificadas ni sobreespecificadas son minimales. Las ERs mostradas suman un total de 72, el corpus contaba con 80 ERs para esa imagen, pero 8 fueron descartadas por estar mal formadas, o no ser verdaderas para el target. En la Tabla 6.8 se puede ver que 44 de 80 ERs son *Parroquia de Santa Cruz*. De las 12 ERs diferentes dadas por los las personas, 9 contienen "Parroquia de Santa Cruz", y acumulan un total de 69 ERs de 80. Las 3 restantes son: *Iglesia* con frecuencia 1 de 80, la cual era mucho más frecuente para el mapa sin zoom. En este caso como en el mapa sin zoom, también es subespecificada. *Iglesia cerca del Ministerio de Asuntos Exteriores*, la cual se dio 1 vez en el mapa con zoom, es decir también era más frecuente en el mapa sin zoom y *En calle de Santo Tomás* la cual se dio sólo 1 vez en el mapa con zoom, también mucho más frecuente en el mapa sin zoom. Esta última es una ER que identifica al target por ser el único objeto que aparece en esa calle.

	ERs del corpus ZOOM para el target *church1*	Cantidad	Frecuencia	SubE	SobreE
1	Parroquia de Santa Cruz	44	61.1%		
2	Parroquia de Santa Cruz en calle de Santo Tomás	9	12.5%		X
3	Parroquia de Santa Cruz en calle de Santo Tomás y De La Concepción Jerónima	4	5.6%		X
4	Parroquia de Santa Cruz en calle de Santo Tomás cerca de Ministerio de Asuntos Exteriores	3	4.2%		X
5	Parroquia de Santa Cruz en calle de Santo Tomás entre calle de Atocha y de La Concepción Jerónima	3	4.2%		X
6	Iglesia de la Parroquia de Santa Cruz	2	2.8%		X
7	Iglesia de la Parroquia de Santa Cruz en calle de Santo Tomás	2	2.8%		X
8	Iglesia cerca del Ministerio de Asuntos Exteriores	1	1.4		
9	En calle de Santo Tomás	1	1.4%	X	
10	Parroquia de Santa Cruz cerca de Ministerio de Asuntos Exteriores	1	1.4%		X
11	Parroquia de Santa Cruz en calle de Atocha cerca de Ministerio de Asuntos Exteriores	1	1.4%		X
12	Iglesia	1	1.4%	X	

Tabla 6.8: ERs del corpus dadas por las personas para el mapa de la Figura 6.11.

Algunas personas incluyeron en la descripción al "Ministerio de Asuntos Exteriores" (exactamente 6 personas), siendo que el nombre del Ministerio no se ve en el mapa, la explicación que damos a esto es que la personas vieron el mapa sin zoom antes de ver este mapa. Como explicamos en la Sección 6.1.2 los mapas se dieron de manera aleatoria, esa aleatoriedad hizo que algunas personas recuerden lo que vieron en otros mapas. Esto es una característica de la metodología de recolección del corpus, que hace que las personas digan cosas que no están

en el modelo. El vocabulario y las probabilidades de uso sacadas del corpus, para el mapa de la Figura 6.11 se ven en el mismo mapa, en la esquina inferior derecha. Así como en el mapa sin zoom, las probabilidades no eran parte del mapa que se mostró a los participantes que completaron las ERs del corpus.

En el modelo de la figura que se muestra en el Apéndice A (con zoom) se ven varias diferencias con respecto al modelo de la Figura 6.9 (sin zoom) y esto se debe a que al acercar el zoom del mapa, algunas cosas aparecen y otras desaparecen. Se ve que la iglesia target no está en la calle de Atocha como parecía en el mapa mostrado en la Figura 6.9. Se agrega el nombre de la calle de La Concepción Jerónima que no se veía en el mapa anterior. Como ya mencionamos, no se ve el nombre del Ministerio de Asuntos Exteriores. Se agrega el nombre *Parroquia de Santa Cruz* que no se veía en el otro mapa. Aparece una nueva calle que está en la parte superior de la manzana de la iglesia, de la cual no sabemos el nombre, pero le asignamos el identificador *str25*. Aparece el nombre de la *calle de Carretas*, cuyo identificador es *str15* en la que está el restaurant *Medina Mayrit*, pero desaparece el nombre del restaurante. El edificio con identificador `d`, en este mapa se ve que es un teatro, el *Teatro Calderón*, también aparece en el mapa el restaurante *La Tagliatela* al lado de un restaurante sin nombre, uno de esos restaurantes se podía ver en el mapa sin zoom. Vamos a suponer que el que se podía ver en el mapa sin zoom era *La Tagliatela*. Notar que dejamos de ver el nombre de la calle en la cual se ubica el teatro y los restaurantes recién nombrados.

Se ejecutó el algoritmo 1000 veces con las probabilidades de uso que se muestran en la esquina inferior derecha del mapa de la Figura 6.11, con el modelo que se muestra en el Apéndice A. Las ERs obtenidas se muestran en la Tabla 6.9.

Como resultado podemos ver que el algoritmo generó *Parroquia de Santa Cruz* en 80% de las ejecuciones, y esto es muy cercano al porcentaje que fue generada por las personas en el corpus (61.1%). La segunda ER más generada por las personas también está en la segunda posición del top 10 ranking del algoritmo con una frecuencia del 8.8%, las personas la generaron un 12.5%. Varias ERs dadas por las personas en el corpus, son sobreespecificadas y consideramos que justamente por eso el algoritmo no las generó, la sobreespecificación que permite el algoritmo es limitada, entonces, no agrega tanta información redundante. *Iglesia de la Parroquia de Santa Cruz* fue generada por las personas en un 2.78% y por el algoritmo en 6.5%. Recordemos el *Ministerio de Asuntos Exteriores* no pertenecía al modelo, por lo tanto, el algoritmo no podía generar las ERs que lo incluyeran.

	ERs generadas por el algoritmo para el target *church1*	#	%
1	Parroquia de Santa Cruz	801	80.1
2	Parroquia de Santa Cruz en Calle de Santo Tomás	88	8.8
3	Iglesia de la Parroquia de Santa Cruz	65	6.5
4	Iglesia en Calle de Santo Tomás	16	1.6
5	Parroquia de Santa Cruz en Calle de la Concepción Jerónima	13	1.3
6	Iglesia en Calle de la Concepción Jerónima	10	1

Tabla 6.9: Posibles realizaciones de las fórmulas que dio el algoritmo para el modelo del Apéndice A. Se muestran las que tienen frecuencia mayor a 10.

Por los resultados obtenidos vemos que las probabilidades de uso, son una buena guía para el algoritmo para conseguir un ranking de ERs que se acerca bastante al dado por las personas. Consiguiendo en este caso una exactitud total de 72.7% como se ve en la Tabla 6.10. En la tabla se muestra una comparación entre el porcentaje de las ERs en el corpus, el porcentaje con lo que la generó el algoritmo. En la columna (E) se muestra la exactitud que es el mínimo entre las 2 columnas anteriores.

	ERs para el target *church1*	Frec.	Frec.-Alg	E
1	Parroquia de Santa Cruz	61.1%	80.1%	61.1%
2	Parroquia de Santa Cruz en calle de Santo Tomás	12.5%	8.8%	8.8%
3	Parroquia de Santa Cruz en calle de Santo Tomás y de La Concepción Jerónima	5.6%	0%	0%
4	Parroquia de Santa Cruz en calle de Santo Tomás cerca de Ministerio de Asuntos Exteriores	4.2%	0%	0%
5	Parroquia de Santa Cruz en calle de Santo Tomás entre calle de Atocha y de La Concepción Jerónima	3	0%	0%
6	Iglesia de la Parroquia de Santa Cruz	2.8%	6.5%	2.8%
7	Iglesia de la Parroquia de Santa Cruz en calle de Santo Tomás	2.8%	0%	0%
8	Iglesia cerca del Ministerio de Asuntos Exteriores	1.5%	0%	0%
9	En calle de Santo Tomás	1.4%	0%	0%
10	Parroquia de Santa Cruz cerca de Ministerio de Asuntos Exteriores	1.4%	0%	0%
11	Parroquia de Santa Cruz en calle de Atocha cerca de Ministerio de Asuntos Exteriores	1.4%	0%	0%
12	Iglesia	1.4%	0%	0%
	Totales	100 %	95.4%	72.7%

Tabla 6.10: Comparación de las ERs del corpus con las generadas por el algoritmo para la Figura 6.11.

6.3.3 Generación de expresiones referenciales plurales

En esta sección analizamos el desempeño de nuestro algoritmo de GER en un mapa con target plural.

Figura 6.12: Imagen sin zoom, target plural. Las probabilidades de uso de las palabras se muestran en la esquina inferior derecha.

Como podemos ver en el mapa de la Figura 6.12 el conjunto target está compuesto por 2 restaurantes, uno que tiene nombre *Restaurante Medina Mayrit* y el otro está ubicado cerca (a unos 100 metros), al lado de una iglesia.

Las ERs del corpus se muestran en la Tabla 6.11 se indica la frecuencia de ocurrencia en el corpus, si es parcial, si es sobreespecificada (SobreE) o subespecificada (SubE). Recordemos que en el Capítulo 2 dimos la definición de ER parcial: es aquella que siendo plural, identifica a un subconjunto estricto del target. Es decir, una ER parcial identifica a una parte del target pero no refiere a todos los elementos que forman parte del conjunto target.

De las ERs del corpus que se ven en la tabla, 55% son parciales, 25% son sobreespecificadas. Cabe notar que el algoritmo no va a generar ERs parciales, ni subespecificadas.

	ERs del corpus ZOOM para el target	Cantidad	%	Parcial	SobreE	SubE
1	Medina Mayrit y el que está cerca	11	18.3%			
2	Medina Mayrit	11	18.3%	X		
3	Restaurantes de calle de Atocha	8	13.3%			X
4	Medina Mayrit y el que está en calle de Atocha	5	8.3%			
5	el de la calle Atocha y De La Cruz	3	5%	X		
6	Medina Mayrit y el que está cerca cercano a la iglesia	3	5%		X	
7	Medina Mayrit en calle de Atocha y el que está cerca cerca de la iglesia	3	5%		X	
8	calle de Atocha	2	3.3%			X
9	Medina Mayrit y el que está cerca en calle de La Cruz	2	3.3%		X	
10	Medina Mayrit y el que está cerca de la iglesia	2	3.3%		X	
11	El que está en calle de Atocha y Carretas	2	3.3%	X		
12	Medina Mayrit y el que está cerca en la calle Atocha y De La Cruz	2	3.3%		X	
13	Medina Mayrit en calle de Atocha y el que está cerca en calle de La Cruz	1	1.7%		X	
14	Medina Mayrit y el que está cerca en calle de Atocha	1	1.6%		X	
15	El que está en calle de Atocha y Carretas, y el de Atocha y de la Cruz	1	1.6%			
16	próximo a la iglesia	1	1.6%	X		
17	Medina Mayrit en calle de Atocha y el que está cerca calle de Atocha y calle de La Cruz	1 1	1.6% 1.7%	 X	X	

Tabla 6.11: ERs del corpus dadas por humanos para el mapa de la Figura 6.12.

	ERs generadas por el algoritmo para el target	Frec.	%
1	El Medina Mayrit y el que está cerca	303	30.3
2	El Medina Mayrit y el que está cerca en la calle de Atocha	83	8.3
3	Los restaurantes que están cerca, uno en calle de Carretas y otro en la calle de la Cruz	20	2
4	El Medina Mayrit de la calle de Atocha y el que está cerca en la calle de la Cruz	13	1.3
5	El que está en la calle de Carretas y el que está cerca	9	0.9
6	El Medina Mayrit en calle de Atocha y el que está cercano a una iglesia	9	0.9

Tabla 6.12: ERs generadas por el algoritmo para el mapa de la Figura 6.12.

Ejecutamos el algoritmo 1000 veces, con input el modelo que se mostró en la Figura 6.10, ya que coincide con el modelo para el ejemplo singular sin zoom. Las probabilidades de uso de las palabras se calcularon como se explica en el Capítulo 4 y se muestran en la esquina

inferior derecha de la Figura 6.12. En el Apéndice B, se muestran las fórmulas generadas por el algoritmo. La exactitud que conseguimos en este caso es de 22.1%, como se ve en la Tabla 6.13, la tabla muestra sólo las ERs del corpus que también fueron generadas por el algoritmo.

	ERs para el target *rest3* y *rest4*	Frec.	Frec.-Alg	E
1	Medina Mayrit y el que está cerca	18.3%	30.3%	18.3%
2	Medina Mayrit en calle de Atocha y el que está cerca cerca de la iglesia	5%	0.9%	0.9%
3	Medina Mayrit en calle de Atocha y el que está cerca en calle de La Cruz	1.7%	1.3%	1.3%
4	Medina Mayrit y el que está cerca en calle de Atocha	1.6%	8.3%	1.6%
	Total			22.1%

Tabla 6.13: ERs en la intersección del corpus y las generadas por el algoritmo para la Figura 6.12.

6.4 Notas finales y linkeo del capítulo

En este capítulo explicamos cómo se recolectó el corpus ZOOM, los materiales que se usaron para la recolección están en el Apéndice A. Discutimos el proceso de anotación y presentamos una descripción detallada. Este corpus se creó viendo la necesidad de existencia de recursos en dominios más naturales. Se recolectó mediante un experimento on-line en el que las personas veían un mapa el cual contenía flechas indicando el target y completaban una ER dirigida a un amigo en la cual tenían que describir los objetos señalados. Cada persona dio 20 ERs, 10 singulares y 10 plurales, de esas 10 singulares o plurales, había 5 con alto nivel de zoom y 5 con bajo nivel de zoom. El corpus tiene en cuenta un dominio que es cercano a aplicaciones del mundo real, y que aborda situaciones más complejas de referencia que los recursos existentes de este tipo. En particular, el dominio de zoom hace uso de contextos con diferentes niveles de detalle, y contiene ejemplos de referencia singular y plural producido por un número relativamente grande de hablantes en ambos idiomas español y portugués. En este corpus podemos encontrar distintos tipos de ERs como las nombradas en el Capítulo 2, tenemos varias ERs por mapa con lo cual podremos luego calcular las probabilidades de uso de las palabras como se explicó en el Capítulo 4. Dadas las características de los mapas usados para la recolección, el corpus tiene ERs relacionales, proposicionales, plurales, parciales, sobreespecificadas y también subespecificadas. Se presentó una evaluación basada en SVM con respecto al corpus ZOOM. Esta evaluación pone en evidencia la complejidad real de la tarea de GER en este dominio. En primer lugar, nos damos cuenta de que un enfoque basado en SVM similar [Ferreira and Paraboni, 2014] en datos de GRE3D3 y GRE3D7 ha obtenido considerablemente mayor exactitud media (0,46 y 0,64 respectivamente) que la obtenida en el dominio ZOOM. Esto, en primer lugar, se explica por el aumento de la complejidad del dominio ZOOM (por ejemplo, en el número de posibles objetos y propiedades atómicas y relacionales etc.). En segundo lugar, las descripciones del ZOOM son propensas a transmitir las relaciones entre el target y varios otros objetos del modelo, como en 'el restaurante en la calle 5, cerca de la biblioteca'. Aunque es común en el uso del lenguaje, el uso de múltiples propiedades relacionales de esta manera ha sido poco investigado, tal vez debido a que puede dar lugar a descripciones ambiguas, como en 'el restaurante cerca de la iglesia en la calle Jackson y el bar en la calle 5' para un restaurante situado cerca del cruce de estas dos calles. Se realizó un caso de estudio para 3 mapas del ZOOM corpus, y se vio que el algoritmo es capaz de conseguir ERs que se encuentran en corpora, y aún más cuando tenemos entornos con target plural, el algoritmo siempre da una

ER, teniendo una ventaja significativa con respecto a las personas, las cuales dieron en más del 50% ERs parciales o que contenían cosas que no eran ciertas en la imagen considerada. A pesar de las ventajas del algoritmo vemos que hay cosas que todavía quedan por pulir como por ejemplo tautologías encontradas en las fórmulas, o propiedades circulares que se podrían sacar, o directamente no permitir.

Capítulo 7

Conclusiones y trabajo futuro

El concepto teórico y filosófico de referencia a un objeto del mundo real ha sido y continúa siendo objetivo de amplio debate [Searle, 1969; Abbott, 2016]. Dada su relevancia para el funcionamiento del lenguaje natural, el concepto práctico y aplicado de referencia es uno de los temas más estudiados en el área de generación automática de lenguaje natural. Esta tesis se enmarca en el área de generación automática de expresiones referenciales (GER), que es un área más reciente que su contraparte filosófica pero también prolífera. [Krahmer and van Deemter, 2012, pag. 174] definen como sigue la tarea en la que se enmarca esta tesis de la siguiente manera.

La tarea de selección de contenido (para la generación automática de expresiones referenciales) es un proceso de balanceo complejo: Debemos decir lo suficiente como para permitir la identificación del referente, pero no demasiado. Una selección de información debe hacerse, y debe hacerse rápido.

Esta definición de la tarea tiene en cuenta dos aspectos centrales considerados en esta tesis. Primero, reconoce que la tarea involucra una selección de información de forma de que el referente se identifique unívocamente. Esta tesis aborda esta cuestión con una solución que combina probabilidades aprendidas del dominio que funcionan como heurística a la hora de seleccionar la información y la noción lógica de similaridad para asegurar unicidad. Segundo, la definición dice que la selección de información debe hacerse rápido. Esta tesis aborda esta cuestión a través de la noción lógica de simulaciones y los algoritmos conocidos eficientes para resolver esta tarea para lenguajes lógicos de distintas expresividades.

El área de GER comienza a activarse en los años 80 con el creciente interés en el Procesamiento Automático de Lenguaje Natural [Appelt, 1985; Appelt and Kronfeld, 1987]. Desde entonces ha avanzado a través de investigaciones independientes y de desafíos de evaluación como el TUNA [Gatt et al., 2008; Gatt et al., 2009], GREC [Belz et al., 2008; Belz et al., 2010], y GIVE [Koller et al., 2010a; Koller et al., 2010b]. Estos desafíos han permitido la comparación de algoritmos del estado del arte, pero se restringen a dominios simplificados y poco realistas como se argumenta en [Krahmer and van Deemter, 2012]. Esta tesis intenta avanzar el estado del arte ante el desafío de dominios más complejos y realistas que necesariamente traen consigo nuevas fuentes de incertidumbre.

A continuación se resumen los principales aportes de esta tesis al área de Generación Automática de Expresiones Referenciales en contextos con incertidumbre. Y luego se discuten nuevas áreas de investigación que desafían los límites de los enfoques actuales.

7.1 Conclusiones

En esta tesis investigamos la generación automática de rankings de expresiones referenciales en contextos con incertidumbre. Las posibles aplicaciones de la generación de expresiones referenciales que deben referirse al mundo real (por ejemplo, software para robots, sistemas gps, etc) sufren de incertidumbre por datos ruidosos de sensores y modelos incompletos de la realidad. Extendimos técnicas y algoritmos de teoría de modelos y simulaciones integrando una distribución finita de probabilidades que representa esta incertidumbre. Nuestro objetivo es generar un ranking de las expresiones referenciales ordenado por la probabilidad de ser correctamente interpretada en el contexto.

A continuación resumimos nuestros aportes organizados en 4 temas. En primer lugar, describimos nuestras contribuciones con respecto al desarrollo de técnicas y de algoritmos de generación de expresiones referenciales que extienden algoritmos clásicos de minimización de autómatas aplicados a la caracterización de modelos de primer orden. Luego, reflexionamos sobre cómo dichos algoritmos fueron extendidos usando probabilidades aprendidas de corpora con técnicas de aprendizaje automático y mencionamos las limitaciones de este enfoque. Además, resumimos lo que aprendimos al evaluar los algoritmos resultantes usando técnicas automáticas y evaluaciones de jueces humanos sobre datos de benchmarks del área. Finalmente, explicamos nuestros resultados en el proceso de recolección y evaluación de un nuevo corpus de expresiones referenciales de puntos de interés en mapas de ciudades con distintos niveles de zoom, el cual es relevante a aplicaciones sobre mapas del mundo real.

7.1.1 Extendiendo algoritmos de simulaciones con probabilidades

Entre las aproximaciones a la GER está la de [Areces et al., 2008], la cual decidimos tomar como punto de partida por la generalidad que nos da la teoría de modelos permitiéndonos elegir un buen balance entre la expresividad del lenguaje lógico a utilizar y la complejidad computacional de los algoritmos de GER. Siguiendo este trabajo previo usamos teoría de modelos, y lenguajes lógicos para generar ERs, y para representar el modelo usamos estructuras de Kripke.

El enfoque basado en teoría de modelos de esta tesis trabaja a nivel semántico y no sintáctico. Esto permite proponer algoritmos genéricos para la GER que son parametrizables para distintos lenguajes lógicos con distinta expresividad. En este sentido los algoritmos propuestos son independientes del lenguaje lógico elegido, es decir funcionan con el lenguaje lógico que queramos. En esta tesis describimos las distintas lógicas usadas para la GER, y analizamos qué clases de fórmulas nos ayudarían a generar ERs con la expresividad que las personas las generan.

El algoritmo original propuesto por [Areces et al., 2008], al igual que otros algoritmos del área, da una sola ER de salida y tiene una lista de preferencias de propiedades ordenada. En esta tesis propusimos cómo reemplazar esa lista de preferencias por una distribución finita de probabilidades, las cuales guían el proceso de inclusión de propiedades unarias y binarias. Agregamos al algoritmo un componente aleatorio, que nos permite ejecutar el algoritmo muchas veces y obtener no una sino un ranking de ERs generadas a partir del modelo dado y la distribución de probabilidades de las palabras del vocabulario.

El algoritmo va particionando el modelo guiado por la distribución de probabilidades, y refinando las clases, una vez que no puede refinar más termina dando como resultado ERs para todos los objetos del modelo, si es posible en la expresividad del lenguaje elegido y con el modelo que le hemos dado como input. El algoritmo es muy eficiente y garantiza encontrar una ER para cada elemento si existe, en tiempo polinomial para $\mathcal{EL}$ y $\mathcal{ALC}$.

7.1.2 Probabilidades de uso y sobreespecificación

Un segundo tema que vale la pena discutir es cómo nuestro algoritmo con sobreespecificación funciona. Como describimos en la Sección 4.4.2 la generación de ERs sobreespecificadas se realiza en dos pasos. En la primera iteración, la probabilidad de incluir una propiedad en la ER depende sólo de su probabilidad de uso. No importa si la propiedad en realidad elimina cualquier distractor. Por lo tanto, la ER resultante puede ser sobreespecificada. Después de que todas las propiedades tuvieron la oportunidad de ser incluidas de esta manera, si la ER resultante no identifica al target unívocamente, entonces el algoritmo entra en una segunda fase en la que se asegura que la ER identifica al target unívocamente. Este modelo está inspirado en la obra de [Keysar et al., 1998] de producción de lenguaje natural y egocentrismo. Keysar et al argumentan que, al producir el lenguaje, el hecho de tener en cuenta el punto de vista de los oyentes no se hace desde el principio, es más bien un ajuste de último momento [Keysar et al., 1998]. En su modelo cognitivo de producción del lenguaje los hablantes adultos producen ERs egocéntricamente, al igual que hacen los niños, pero luego ajustan las ERs para que el destinatario sea capaz de identificar al target de forma unívoca. El primer paso es egocéntrico, es un proceso heurístico basado en un modelo de la **prominencia** de las propiedades de la escena que contiene el target.

Nuestra definición de probabilidad de uso p_{use} pretende capturar las **prominencias**[1] de las propiedades de diferentes escenas y objetivos. El p_{use} de una propiedad cambia de acuerdo a la escena como discutimos en la Sección 4.2. Esto está en contraste con el trabajo anterior, donde el prominencia de una propiedad se asume constante en un dominio. Keysar et al argumentan que la razón del procedimiento de generar y ajustar puede tener que ver con las limitaciones de procesamiento de información de la mente. Si la heurística que guía la fase egocéntrica está bien sintonizada, tiene éxito en el primer intento, siendo la primer ER adecuada en la mayoría de los casos y rara vez requiere ajustes. Es interesante observar que un comportamiento similar se obtiene con nuestro algoritmo: cuando los valores de p_{use} se aprenden del dominio en que se van a utilizar, el algoritmo es mucho más preciso y mucho más rápido.

7.1.3 Desafíos en la evaluación de sistemas de generación

En esta tesis se estudió la generación automática de expresiones referenciales teniendo como meta generar expresiones referenciales como las personas lo harían. Para eso se estudió la generación humana de expresiones referenciales desde una perspectiva psicolingüística concluyendo que el no-determinismo y la sobreespecificación son puntos claves a tener en cuenta. También aprendimos que no siempre las personas incluyen relaciones cuando es estrictamente necesario, es decir cuando no se puede identificar al target sólo con propiedades proposicionales. Vimos el rol importante de la existencia de corpora, que nos permite varias cosas: por un lado aprender qué dirían las personas en distintas situaciones, y usar esos datos para intentar imitar al corpus, y por otro nos permite comparar la salida de los algoritmos con ERs de las personas. Mostramos especificaciones de corpora existente, los clasificamos según el tipo de ERs que contiene, y los comparamos entre ellos.

Dimos definiciones y vocabulario específico del área, y estudiamos las diferentes aproximaciones a la solución. En este proceso vimos que hay varias cosas a tener en cuenta para generar ERs automáticamente: primero es cómo y qué representar del modelo, y esto es muy importante ya que puede hacer que nuestro algoritmo genere cosas no deseadas. En segundo lugar, cómo guiar al algoritmo para que de una ERs adecuada. Algunos algoritmos dan una ER minimal, pero aún así pueden generar varias ERs minimales, entonces, cuál de ellas elegir?. Vimos que muchos algoritmos usan una lista de preferencias fija de propiedades, otros una función de costo

[1] Del inglés *saliency*.

a minimizar, es decir devuelven la ER de menor costo. En cualquiera de los casos el algoritmo da como mucho una ER, y no tenemos información de qué tan buena es esa ER. Concluimos que la generación de expresiones referenciales es una tarea en la cual hay incertidumbre.

Evaluamos nuestra propuesta sobre datos de benchmarks del área, comparando con corpora en la medida de lo posible. Probamos el algoritmo propuesto en el corpus GRE3D7 y se encontró que es capaz de generar una gran proporción de las ERs sobreespecificadas que se encuentran en el corpus sin generar expresiones referenciales trivialmente redundantes.

[Viethen, 2011] entrenó árboles de decisión que son capaces de lograr una precisión media del 65% en el corpus GRE3D7. El enfoque basado en árboles de decisión es capaz de generar descripciones relacionales sobreespecificadas, pero que podrían no ser expresiones referenciales. De hecho, como los árboles de decisión no verifican la extensión de la expresión generada en el modelo de la escena, la descripción generada podría no identificar de forma única el objetivo. Como ya hemos comentado, nuestro algoritmo asegura la terminación y siempre encuentra una expresión referencial, si existe. Por otra parte, logra un promedio de 75,03 % de precisión en las escenas utilizadas en nuestras pruebas.

Algoritmos diferentes para la generación de expresiones referenciales sobreespecificadas se han propuesto en los últimos años (ver, por ejemplo, [de Lucena and Paraboni, 2008; Ruud et al., 2012]). Pero, hasta donde sabemos, no han sido evaluados en el GRE3D7 corpus y, por lo tanto, la comparación es difícil. Los algoritmos de [de Lucena and Paraboni, 2008] y [Ruud et al., 2012] han sido evaluados en el corpus TUNA-AR [Gatt et al., 2008] logrando una precisión de 33 % y 40 % respectivamente. Como el corpus TUNA-AR sólo incluye ERs proposicionales, sería interesante evaluar la performance de estos algoritmos en corpora con ERs relacionales como el GRE3D7.

7.1.4 Generación sobre un contexto del mundo real

Se creó, conjuntamente con la Universidad de São Paulo, el ZOOM corpus de expresiones referenciales de puntos de interés en mapas. Este corpus está en proceso de revisión y será liberado para investigación en los próximos meses. El proceso de recolección de este corpus demostró ser complejo, esto se vio por ejemplo ya que muchas de las descripciones recolectadas debieron ser descartadas por contener información incorrecta. Además el corpus mostró tener un porcentaje de expresiones subespecificadas mucho mayor que corpus anteriores recolectados sobre dominios más simples.

Sobre mapas de este corpus se realizaron 3 casos de estudio que se describen en el Capítulo 6. Estos casos de estudio, si bien muestran que el algoritmo es capaz de generar ERs en dominios naturales de la vida real, como son los mapas de ciudades, también dejó al descubierto algunas cuestiones lógicas que podrían ser mejoradas en trabajo futuro. En el caso particular de la GER con target plural, nos muestra que todavía queda mucho por hacer.

7.2 Trabajo futuro

En esta sección describimos cuatro líneas de trabajo claras en las que los aportes de esta tesis podrían ser continuados. Algunas de estas áreas han sido previamente estudiadas pero aún plantean muchos desafíos, como la generación de ERs plurales y generación de ERs de forma incremental en entornos interactivos. Otros temas como el uso de lenguajes lógicos modales más expresivos y la inclusión de una teoría del dominio son extensiones naturales del enfoque a la GER adoptado en esta tesis que no han sido estudiados previamente.

7.2.1 Interactividad e incrementalidad

Estudios previos [Jordan and Walker, 2005; Gupta and Stent, 2005] sugieren que los algoritmos del estado del arte de GER que usan un orden fijo de las propiedades del contexto no se pueden aplicar de forma directa en aplicaciones interactivas. Estos estudios encontraron que en este tipo de aplicaciones los algoritmos del estado del arte se desempeñan peor que estrategias basadas en patrones sencillos y que prestan atención al historial de la interacción. Un experimento realizado sobre el corpus iMap [Viethen et al., 2011] confirmó la importancia de modelar el terreno común de una interacción para producir ERs útiles. El trabajo publicado en [Stoia et al., 2006] de GER en entornos de interacción situada en un videojuego hace evidente la necesidad de variar el orden de las propiedades utilizado por los algoritmos dependiendo de características del contexto que se modifican continuamente (como la distancia a los objetos target y distractores). Nuestro algoritmo, al proveer una distribución finita de probabilidades asociada a cada propiedad en lugar de requerir un orden fijo, permite un mayor control y dinamismo a la hora de elegir propiedades para una ER en un contexto cambiante. Sería interesante evaluar su desempeño en este tipo de aplicaciones interactivas.

Modelos que cambian parcialmente como sucede con los mapas con distintos niveles de ZOOM en el corpus son comunes en aplicaciones interactivas donde el usuario está navegando la aplicación. En estos dominios el contexto de referencia puede cambiar en su estructura o en el número de objetos y propiedades, mientras algunas características del contexto se mantienen. El desafío de los algoritmos de GER sería producir ERs sin empezar desde cero cada vez que el modelo cambia. Algoritmos como el nuestro que se basan en particiones locales del contexto podrían tener una ventaja en este sentido, esta es una línea interesante de trabajo futuro.

Como trabajo futuro también tenemos la intención de evaluar nuestro algoritmo de dominios más complejos como los de dominio abiertos proporcionados en Folksonimies [Pacheco et al., 2012]. Los corpus Stars/2 recientemente liberados, serían interesantes para evaluar el algoritmo, ya que poseen situaciones complejas de referencia que hacen amplio uso de relaciones y expresiones de hasta tres landmarks. También planeamos explorar corpus que se obtienen de la interacción humana, como el GIVE Corpus [Gargett et al., 2010] donde es común observar ERs en varios intentos, es decir dando expresiones parciales en cada intento. Bajo presión de tiempo los hablantes producen primero una expresión subespecificada que incluye las propiedades más destacadas del target (por ejemplo, "el botón rojo"). Y luego, en un siguiente enunciado, añaden propiedades adicionales (por ejemplo, "a la izquierda de la lámpara") para que la expresión sea una ER adecuada para identificar de manera única al target. Algoritmos para generar este tipo de ERs han sido propuestos en [Denis, 2010].

7.2.2 Plurales: de targets singleton a conjuntos

Un potencial interesante de nuestros algoritmos que se basan en dividir el modelo a través de la semántica, es que, en principio puede dar ERs para un conjunto target que no sea singleton. En el Capítulo 6.3 se presenta un caso de estudio de la generación de expresiones referenciales plurales pero aún no hemos abordado varias cuestiones relacionadas a este tema señaladas por trabajo previo.

En particular, las expresiones referenciales generadas por nuestro algoritmo tienden a ser largas y conjuntivas cuando en realidad existen expresiones plurales colectivas que serían más efectivas. En trabajo previo se ha sugerido el uso de disyunción en el lenguaje lógico usado para la generación [Gardent, 2002; Gatt, 2007; Khan et al., 2008].

7.2.3 Lenguajes más expresivos: negación y probabilidades

En esta tesis utilizamos principalmente el lenguaje lógico $\mathcal{EL}$. Es decir sin ningún tipo de negación. Esto mostró ser apropiado para dominios estáticos y no agentivos en los que trabajamos. Sería interesante investigar si para otros dominios sería útil tener algún tipo de negación en el lenguaje.

En esta tesis la combinación de lógica y probabilidades se realiza a nivel algorítmico. Recientemente se han comenzado a investigar semántica probabilística para lógica modal [Lando, 2012]. Sería interesante investigar si incluir probabilidades en la misma semántica de la lógica mejoraría el desempeño de la GER.

7.2.4 Razonando con una teoría del dominio

En el Capítulo 1 discutimos que todos los modelos son incompletos, especificarlos y tratar de completarlos todo lo posible es un trabajo muy laborioso. Esto puede hacerse más fácil si fuera posible escribir axiomas del dominio que ayuden a extender los modelos automáticamente. Por ejemplo, si sabemos que el objeto *e1* está a la derecha del objeto *e2* y el objeto *e2* está a la derecha del objeto *e3*, se puede especificar un axioma en algún framework de especificación de conocimiento y razonamiento que automáticamente agregue que *e1* está a la derecha del objeto *e3* simplemente diciendo que la relación a la derecha de es transitiva.

Existen diversos frameworks para especificar teorías de dominios de este tipo. Uno de los más difundidos son las lógicas de descripción [Baader et al., 2003]. Nuestro enfoque a la GER se presta especialmente para la integración de teorías de dominio dado que se basa en este tipo de lógicas. En particular, nuestros algoritmos generan expresiones referenciales estructuralmente complejas como las que se ilustran en el Capítulo 6 y en el Apéndice B. Sería interesante evaluar empíricamente si la complejidad de estas fórmulas es apropiada para la tarea de GER y definir así cómo es apropiado restringir el lenguaje lógico a utilizar en esta tarea.

Bibliografía

[Abbott, 2016] Abbott, B. (2016). *Reference*, chapter 12. Oxford University Press, Oxford, United Kingdom.

[Altamirano et al., 2012] Altamirano, R., Areces, C., and Benotti, L. (2012). Probabilistic refinement algorithms for the generation of referring expressions. In *Proceedings of the 24th International Conference on Computational Linguistics 2012: Posters*, pages 53–62, Mumbai, India. The COLING 2012 Organizing Committee.

[Altamirano et al., 2015] Altamirano, R., Ferreira, T. C., Paraboni, I., and Benotti, L. (2015). Zoom: a corpus of natural language descriptions of map locations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 69–75.

[Appelt and Kronfeld, 1987] Appelt, D. and Kronfeld, A. (1987). A computational model of referring. In McDermott, J., editor, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 640–647, Los Altos, California. Morgan Kaufmann.

[Appelt, 1985] Appelt, D. E. (1985). Planning english referring expressions. *Artificial Intelligence*, 26(1):1–33.

[Areces et al., 2011] Areces, C., Figueira, S., and Gorín, D. (2011). Using logic in the generation of referring expressions. In Pogodalla, S. and Prost, J., editors, *Proceedings of the 6th International Conference on Logical Aspects of Computational Linguistics (LACL 2011)*, volume 6736 of *Lecture Notes in Computer Science*, pages 17–32. Springer, Montpelier.

[Areces et al., 2008] Areces, C., Koller, A., and Striegnitz, K. (2008). Referring expressions as formulas of description logic. In *Proceedings of the Fifth International Natural Language Generation Conference*, INLG '08, pages 42–49, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Areces and ten Cate, 2006] Areces, C. and ten Cate, B. (2006). Hybrid logics. In Blackburn, P., Wolter, F., and van Benthem, J., editors, *Handbook of Modal Logics*. Elsevier.

[Arts et al., 2011] Arts, A., Maes, A., Noordman, L. G. M., and Jansen, C. (2011). Overspecification facilitates object identification. *Journal of Pragmatics*, 43(1):361–374.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA.

[Belz and Gatt, 2008] Belz, A. and Gatt, A. (2008). Intrinsic vs. extrinsic evaluation measures for referring expression generation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 197–200, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Belz et al., 2008] Belz, A., Kow, E., Viethen, J., and Gatt, A. (2008). The grec challenge: Overview and evaluation results. In *Proceedings of the Fifth International Natural Language Generation Conference*, INLG '08, pages 183–193, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Belz et al., 2010] Belz, A., Kow, E., Viethen, J., and Gatt, A. (2010). Generating referring expressions in context: The grec task evaluation challenges. In Krahmer, E. and Theune, M., editors, *Empirical Methods in Natural Language Generation: Data-oriented Methods and Empirical Evaluation*, pages 294–327. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Benotti and Altamirano, 2013] Benotti, L. and Altamirano, R. (2013). Evaluation of a refinement algorithm for the generation of referring expressions. In Brézillon, P., Blackburn, P., and Dapoigny, R., editors, *Modeling and Using Context: 8th International and Interdisciplinary Conference, CONTEXT 2013, Annecy, France, October 28 -31, 2013, Proceedings*, pages 31–44. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Blackburn et al., 2001] Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*. Cambridge University Press.

[Clark, 1992] Clark, H. H. (1992). *Arenas of language use*. University of Chicago Press.

[Clark, 1996] Clark, H. H. (1996). *Using language*. Cambridge university press, Cambridge.

[Clark and Marshall, 1981] Clark, H. H. and Marshall, C. R. (1981). Definite Reference and Mutual Knowledge. In Webber, B. L., Joshi, A. K., and Sag, I. A., editors, *Elements of Discourse Processing*, pages 10–63. Cambridge University Press, New York.

[Clark and Wilkes-Gibbs, 1986] Clark, H. H. and Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition*, 22(1):1–39.

[Cohen, 1960] Cohen, J. (1960). A coeficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

[Dale, 1989] Dale, R. (1989). Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*, ACL '89, pages 68–75, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Dale, 2002] Dale, R. (2002). Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 68–75.

[Dale and Haddock, 1991a] Dale, R. and Haddock, N. (1991a). Generating referring expressions involving relations. In *Proceedings of the 5th conference of the European chapter of the Association for Computational Linguistics*, EACL '91, pages 161–166.

[Dale and Haddock, 1991b] Dale, R. and Haddock, N. J. (1991b). Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4):252–265.

[Dale and Reiter, 1995] Dale, R. and Reiter, E. (1995). Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

[Dale and Viethen, 2009] Dale, R. and Viethen, J. (2009). Referring expression generation through attribute-based heuristics. In *Proceedings of ENLG-2009*, pages 58–65.

[de Lucena and Paraboni, 2008] de Lucena, D. J. and Paraboni, I. (2008). USP-EACH frequency-based greedy attribute selection for referring expressions generation. In *Proceedings of the 5th International Conference on Natural Language Generation 2008*, ENLG '08, pages 219–220. Association for Computational Linguistics.

[Denis, 2010] Denis, A. (2010). Generating referring expressions with reference domain theory. In *INLG 2010 - Proceedings of the Sixth International Natural Language Generation Conference, July 7-9*, pages 27–35, Dublin, Ireland.

[Dice, 1945] Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

[Dovier et al., 2004] Dovier, A., Piazza, C., and Policriti, A. (2004). An efficient algorithm for computing bisimulation equivalence. *Theoretical Computer Science*, 311(1–3).

[Dräger and Koller, 2012] Dräger, M. and Koller, A. (2012). Generation of landmark-based navigation instructions from open-source data. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 757–766, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ebbinghaus et al., 1996] Ebbinghaus, H., Flum, J., and Thomas, W. (1996). *Mathematical Logic*. Springer.

[Engelhardt et al., 2006] Engelhardt, P., Bailey, K., and Ferreira, F. (2006). Do speakers and listeners observe the gricean maxim of quantity? *Journal of Memory and Language*, 54(4):554–573.

[Fabbrizio et al., 2008] Fabbrizio, G. D., Stent, A. J., and Bangalore, S. (2008). Trainable speaker-based referring expression generation. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, pages 151–158, Stroudsburg, PA, USA.

[Ferreira and Paraboni, 2014] Ferreira, T. C. and Paraboni, I. (2014). Classification-based referring expression generation. *Lecture Notes in Computer Science*, 8403:481–491.

[Foster et al., 2009] Foster, M. E., Giuliani, M., Isard, A., Matheson, C., Oberlander, J., and Knoll, A. (2009). Evaluating description and reference strategies in a cooperative human-robot dialogue system. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, Pasadena, California.

[Gardent, 2002] Gardent, C. (2002). Generating minimal definite descriptions. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[Garey and Johnson, 1979] Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. Freeman.

[Gargett et al., 2010] Gargett, A., Garoufi, K., Koller, A., and Striegnitz, K. (2010). The GIVE-2 corpus of giving instructions in virtual environments. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*, Malta.

[Gatt, 2007] Gatt, A. (2007). *Generating coherent references to multiple entities*. PhD thesis, Department of Computer Science, University of Aberdeen.

[Gatt et al., 2008] Gatt, A., Belz, A., and Kow, E. (2008). The TUNA challenge 2008: Overview and evaluation results. In *Proceedings of the 5th International Conference on Natural Language Generation*, ENLG '09, pages 198–206. Association for Computational Linguistics.

[Gatt et al., 2009] Gatt, A., Belz, A., and Kow, E. (2009). The TUNA-REG challenge 2009: Overview and evaluation results. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 174–182, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Gatt et al., 2007] Gatt, A., van der Sluis, I., and van Deemter, K. (2007). Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of ENLG-07*.

[Gatt et al., 2011] Gatt, A., van Gompel, R., Krahmer, E., and van Deemter, K. (2011). Non-deterministic attribute selection in reference production. In *Workshop on the Production of Referring Expressions*, PRE-CogSci '11, pages 1–7.

[Gupta and Stent, 2005] Gupta, S. and Stent, A. (2005). Automatic evaluation of referring expression generation using corpora. In *Proceedings of the 1st Workshop on Using Corpora in Natural Language Generation (UCNLG)*, pages 1–6.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

[Henzinger et al., 1995] Henzinger, M. R., Henzinger, T. A., and Kopke, P. W. (1995). Computing simulations on finite and infinite graphs. In *Proceedings of 36th Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press.

[Hopcroft, 1971] Hopcroft, J. (1971). An $n \log(n)$ algorithm for minimizing states in a finite automaton. In Kohave, Z., editor, *Theory of Machines and computations*. Academic Press.

[Horacek, 1997] Horacek, H. (1997). An algorithm for generating referential descriptions with flexible interfaces. In *Proceedings of the 35th ACL*.

[Jordan and Walker, 2005] Jordan, P. W. and Walker, M. A. (2005). Learning content selection rules for generating object descriptions in dialogue. *Journal of Artificial Intelligence Research (JAIR)*, 24:157–194.

[Jurafsky and Martin, 2008] Jurafsky, D. and Martin, J. (2008). *Speech and Language Processing*. Pearson Prentice Hall, second edition.

[Kelleher and Kruijff, 2006] Kelleher, J. and Kruijff, G.-J. (2006). Incremental generation of spatial referring expressions in situated dialog. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 1041–1048.

[Keysar et al., 1998] Keysar, B., Barr, D. J., and Horton, W. S. (1998). The Egocentric Basis of Language Use. *Current Directions in Psychological Science*, 7(2):46–49.

[Khan et al., 2008] Khan, I. H., van Deemter, K., and Ritchie, G. (2008). Generation of referring expressions: Managing structural ambiguities. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 433–440, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Koller et al., 2010a] Koller, A., Striegnitz, K., Byron, D., Cassell, J., Dale, R., Moore, J., and Oberlander, J. (2010a). The first challenge on generating instructions in virtual environments. In Krahmer, E. and Theune, M., editors, *Empirical Methods in Natural Language Generation*, volume 5790 of *LNAI*. Springer.

[Koller et al., 2010b] Koller, A., Striegnitz, K., Gargett, A., Byron, D., Cassell, J., Dale, R., Moore, J., and Oberlander, J. (2010b). Report on the Second NLG Challenge on Generating Instructions in Virtual Environments (GIVE-2). In *Proceedings of the Sixth International Natural Language Generation Conference (Special session on Generation Challenges)*, Dublin.

[Krahmer and Theune, 2002] Krahmer, E. and Theune, M. (2002). Efficient context-sensitive generation of referring expressions. In van Deemter, K. and Kibble, R., editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pages 223–264. CSLI Publications, Stanford, CA.

[Krahmer and Theune, 2010] Krahmer, E. and Theune, M., editors (2010). *Empirical Methods in Natural Language Generation: Data-oriented Methods and Empirical Evaluation*. Springer-Verlag, Berlin, Heidelberg.

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

[Krahmer et al., 2003] Krahmer, E., van Erk, S., and Verleg, A. (2003). Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

[Krahmer et al., 2008] Krahmer, E. J., Theune, M., Viethen, J., and Hendrickx, I. (2008). Graph: The costs of redundancy in referring expressions. In *Proceedings of the 5th International Natural Language Generation Conference, Salt Fork, Ohio, USA*, INLG '08, pages 227–229, USA. The Association for Computational Linguistics.

[Kurtonina and de Rijke, 1998] Kurtonina, N. and de Rijke, M. (1998). Expressiveness of concept expressions in first-order description logics. *Artificial Intelligence*, 107.

[Lando, 2012] Lando, T. A. (2012). *Probabilisitic Sematics for Modal Logics*. PhD thesis, University of California, Berkley.

[Luccioni et al., 2015] Luccioni, A., Benotti, L., and Landragin, F. (2015). Overspecified references: An experiment on lexical acquisition in a virtual environment. *Computers in Human Behavior*, 49:94–101.

[Mellish and Evans, 2004] Mellish, C. and Evans, R. (2004). Implementation architectures for natural language generation. *Natural Language Engineering*, 10(3/4).

[Pacheco et al., 2012] Pacheco, F., Duboue, P., and Domínguez, M. (2012). On the feasibility of open domain referring expression generation using large scale folksonomies. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 641–645, Montréal, Canada. Association for Computational Linguistics.

[Paige and Tarjan, 1987] Paige, R. and Tarjan, R. (1987). Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989.

[Paraboni et al., 2015] Paraboni, I., Galindo, M. R., and Iacovelli, D. (2015). Generating overspecified referring expressions: the role of discrimination. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 76–82.

[Paraboni et al., 2016] Paraboni, I., Galindo, M. R., and Iacovelli, D. (2016). Stars2: a corpus of object descriptions in a visual domain. *Language Resources and Evaluation*, pages 1–24.

[Paraboni et al., 2007] Paraboni, I., van Deemter, K., and Masthoff, J. (2007). Generating referring expressions: Making referents easy to identify. *Computational Linguistics*, 33(2):229–254.

[Passonneau, 2006] Passonneau, R. (2006). Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. In *In Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

[Pechman, 1989] Pechman, T. (1989). Incremental speech production and referential overspecification. *Linguistics*, 27:89–110.

[Reiter and Dale, 1992] Reiter, E. and Dale, R. (1992). A fast algorithm for the generation of referring expressions. In *In proceedings of The 15th International Conference on Computational Linguistics*.

[Reiter and Dale, 2000] Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

[Ruud et al., 2012] Ruud, K., Emiel, K., and Mariët, T. (2012). Learning preferences for referring expression generation: Effects of domain, language and algorithm. In *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*, pages 3–11, Utica, IL. Association for Computational Linguistics.

[Searle, 1969] Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge Univesity Press, Cambridge, England.

[Stoia et al., 2006] Stoia, L., Shockley, D. M., Byron, D. K., and Fosler-Lussier, E. (2006). Noun phrase generation for situated dialogs. In *Proceedings of the Fourth International Natural Language Generation Conference*, INLG '06, pages 81–88, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Stone and Webber, 1998] Stone, M. and Webber, B. (1998). Textual economy through close coupling of syntax and semantics. In *Proceedings of the 9th International Workshop on Natural Language Generation*, INLG '98, pages 178–187.

[Turner et al., 2009] Turner, R., Sripada, Y., and Reiter, E. (2009). Generating approximate geographic descriptions. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 42–49, Athens, Greece. Association for Computational Linguistics.

[van Deemter, 2002] van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1).

[van Deemter, 2016] van Deemter, K. (2016). *Computational Model of Referring: a Study in Cognitive Science*. MIT Press.

[van Deemter et al., 2006] van Deemter, K., van der Sluis, I., and Gatt, A. (2006). Building a semantically transparent corpus for the generation of referring expressions. In *Proceedings of the 4th International Natural Language Generation Conference*, INLG '06.

[van der Sluis et al., 2007] van der Sluis, I., Gatt, A., and van Deemter., K. (2007). Evaluating algorithms for the generation of referring expressions: Going beyond toy domains. In *Proceedings of Recent Advances in Natural Language Processing*.

[Viethen, 2011] Viethen, H. A. E. (2011). *The Generation of Natural Descriptions: Corpus-Based Investigations of Referring Expressions in Visual Domains*. PhD thesis, Macquarie University, Sydney, Australia.

[Viethen and Dale, 2006] Viethen, J. and Dale, R. (2006). Algorithms for generating referring expressions: Do they do what people do? In *Proceedings of the 4th International Natural Language Generation Conference*, INLG '06, pages 63–70.

[Viethen and Dale, 2011] Viethen, J. and Dale, R. (2011). Gre3d7: A corpus of distinguishing descriptions for objects in visual scenes. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, pages 12–22, Edinburgh, Scotland. Association for Computational Linguistics.

[Viethen et al., 2011] Viethen, J., Dale, R., and Guhe, M. (2011). Generating subsequent reference in shared visual scenes: Computation vs re-use. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1158–1167, Edinburgh, Scotland, UK. Association for Computational Linguistics.

[White et al., 2010] White, M., Clark, R. A., and Moore, J. D. (2010). Generating tailored, comparative descriptions with contextually appropriate intonation. *Computational Linguistics*, 36(2):159–201.

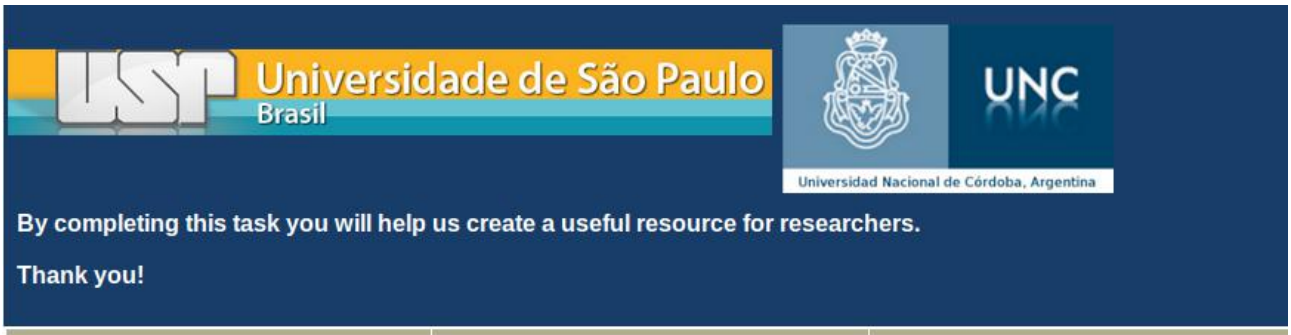
Apéndice A

Materiales para la recolección del corpus ZOOM

En este apéndice se muestran los materiales que se usaron para la recolección del corpus ZOOM. Para ver en detalle información del corpus, su recolección y anotación vea la Sección 6.

A.1 Instrucciones para la obtención del corpus

En la Figura A.1 se muestra en link que se envió por correo a los participantes que completaron el experimento para la recolección del corpus ZOOM. En la página de ven las instrucciones en tres idiomas, la persona tenía que seleccionar el idioma en el que quería completar el experimento. Esta página contenía las instrucciones en cada uno de los tres idiomas.



ENGLISH	PORTUGUES	SPANISH
Please read carefully the following instructions Your task is to describe the place or places (restaurants, churches etc.) indicated by one or two red arrows on a series of maps, completing the sentence "It would be interesting to visit...", as if you were giving directions to a friend. Please try to be precise in your answers, not simply writting "the restaurant" or "the churches", but don't worry about writing an overly long description: just follow your first intuition. The task will take about 20 minits to be completed. Please avoid interrumptions until you finish. For the English version of the task click HERE	Por favor siga as instruções cuidadosamente. Sua tarefa é descrever os locais (restaurantes, igrejas etc.) indicados pelas setas vermelhas em uma série de mapas, completando a frase "Seria interessante conhecer..." como se você estivesse dando instruções a um amigo. Tente ser preciso nas suas respostas, não escrevendo apenas "o restaurante" ou "as igrejas", mas não se preocupe em escrever uma descrição mais longa do que o necessário: simplesmente siga sua primeira intuição. A tarefa leva cerca de 20 minutos. Por favor evite interrupções até concluí-la. Para a versão em Português da tarefa, clique AQUI	Por favor, lea atentamente las siguientes instrucciones Su tarea es describir el o los lugares (restaurantes, iglesias, etc) indicados por una o dos flechas rojas en una serie de mapas, completando la frase "Es interesante visitar..." como si estuviera dando indicaciones a un amigo. Intente ser preciso en sus respuestas no escribiendo solamente "el restaurante" o "las iglesias", pero no se preocupe en dar una descripción demasiado detallada: sólo tiene que seguir su primera intuición. Esta tarea le llevará alrededor de 20 minutos, por favor evite interrupciones. Para la versión en Español de la tarea, click AQUI

Figura A.1: Primera página del experimento.

Una vez seleccionado el idioma del experimento, se solicitaba información demográfica de los participantes, como se muestra en la Figura A.2. También se solicitaba que se acepten los

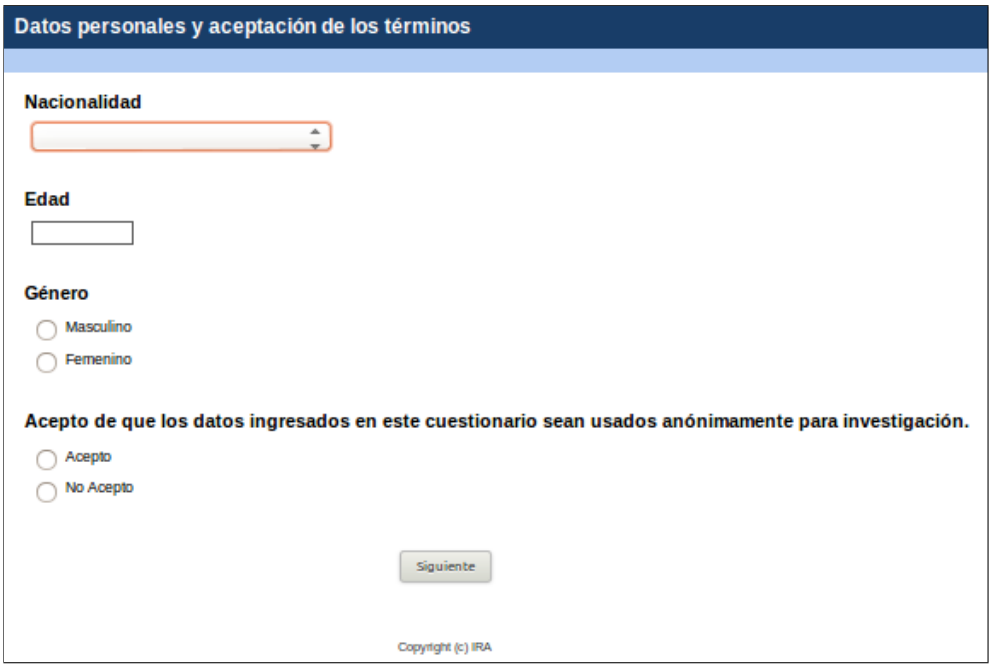


Figura A.2: Datos requeridos.

términos y condiciones que decían que los datos ingresados serán usados para investigación.

En la Figura A.3 se ve una imagen estímulo, un mensaje "Es interesante visitar..." que es la oración que el participantente tenía que completar con una expresión referencial del o los objetos apuntados por las flechas. También se veía una barra indicadora de progreso, y las instrucciones.

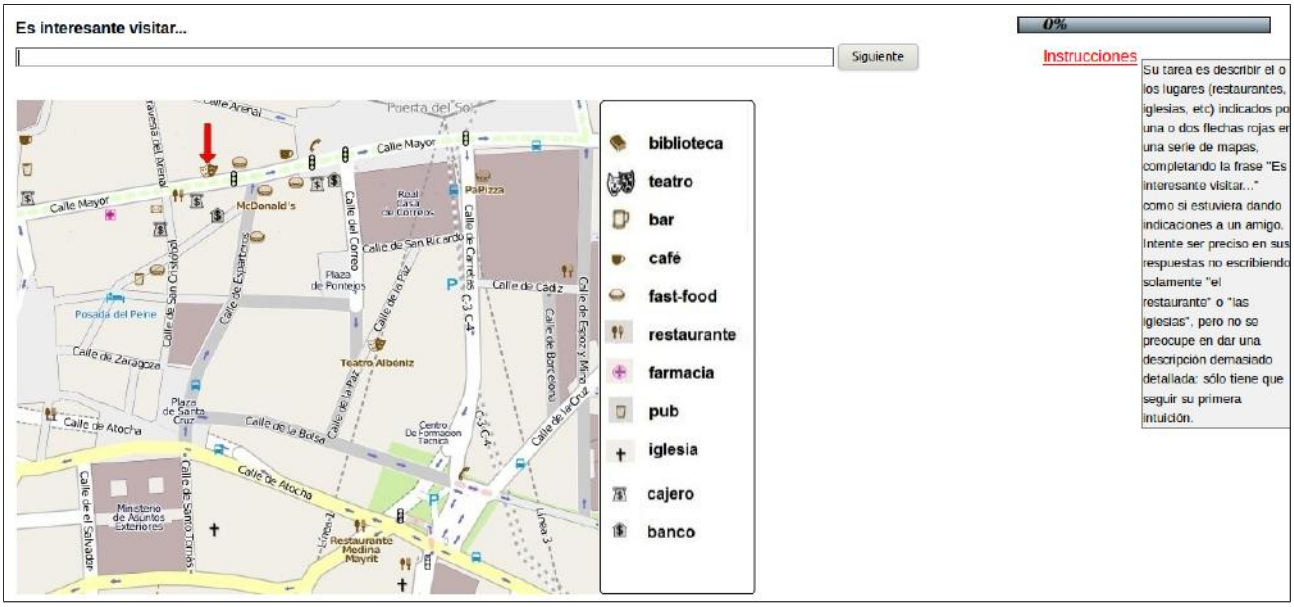


Figura A.3: Interface del experimento.

A.2 Imágenes estímulo del corpus

Aquí se muestran las imágenes que se mostraron a los participantes que completaron la tarea de recolección del corpus ZOOM para el español. El corpus cuenta con imágenes con y sin zoom, para targets singulares y plurales.

A.2.1 Estímulos singulares

Se muestran mapas con targets singulares. Las imágenes (a) muestran targets singleton. La imagen (b) correspondiente muestra el mismo target pero con zoom.

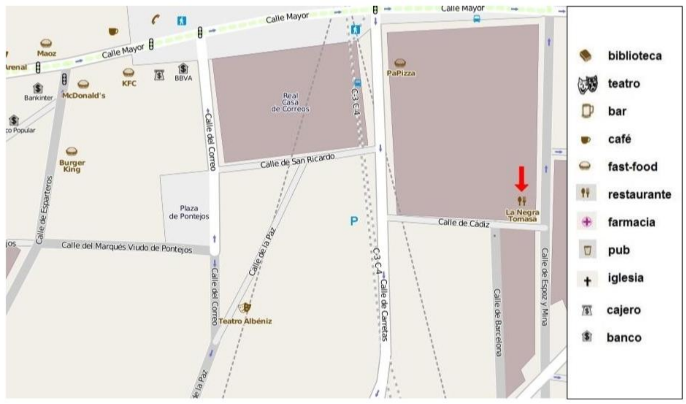
(a)

(b)

(a)

(b)

(a)



(b)

(a)

(b)

(a)

(b)

A.2.2 Estímulos plurales

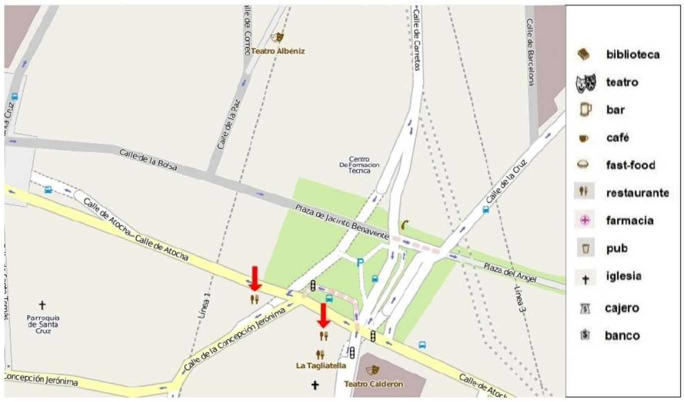
Se muestran mapas con targets plurales. Las imágenes (a) muestran targets con 2 elementos. La imagen (b) correspondiente muestra el mismo target pero con zoom.

(a)

(b)


Puerta del Sol
Calle Mayor
McDonald's
PaPizza
Plaza de Pontejos
Posada del Peine
Calle de Zaragoza
Teatro Albéniz
Calle de Atocha
Calle de la Bolsa
Calle de Cádiz
Centro De Formacion Tecnica
Ministerio de Asuntos Exteriores
Restaurante Medina Mayrit
Línea 3
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(a)


Teatro Albéniz
Calle de la Bolsa
Centro De Formacion Tecnica
Plaza de Jacinto Benavente
Calle de Atocha
Calle de la Cruz
Plaza del Angel
Parroquia de Santa Cruz
Calle de la Concepción Jerónima
La Tagliatella
Teatro Calderón
Línea 3
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(b)


Puerta del Sol
Calle Mayor
McDonald's
PaPizza
Plaza de Pontejos
Posada del Peine
Calle de Zaragoza
Teatro Albéniz
Calle de Atocha
Calle de la Bolsa
Calle de Cádiz
Centro De Formacion Tecnica
Ministerio de Asuntos Exteriores
Restaurante Medina Mayrit
Línea 3
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(a)


La Negra Tomasa
Calle de Cádiz
Calle del Marqués Viudo de Pontejos
Teatro Albéniz
Calle de la Bolsa
Centro De Formacion Tecnica
Plaza de Jacinto Benavente
Calle de Atocha
Plaza del Angel
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(b)


Puerta del Sol
Calle Mayor
McDonald's
PaPizza
Plaza de Pontejos
Posada del Peine
Calle de Zaragoza
Teatro Albéniz
Calle de Atocha
Calle de la Bolsa
Calle de Cádiz
Centro De Formacion Tecnica
Ministerio de Asuntos Exteriores
Restaurante Medina Mayrit
Línea 3
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(a)


Calle de Zaragoza
Teatro Albéniz
Calle de la Bolsa
Plaza de la Provincia
Calle de Atocha
Ministerio de Asuntos Exteriores
Parroquia de Santa Cruz
Calle de la Concepción Jerónima
La Tagliatella
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(b)


Puerta del Sol
Calle Mayor
McDonald's
PaPizza
Plaza de Pontejos
Posada del Peine
Calle de Zaragoza
Teatro Albéniz
Calle de Atocha
Calle de la Bolsa
Calle de Cádiz
Centro De Formacion Tecnica
Ministerio de Asuntos Exteriores
Restaurante Medina Mayrit
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(a)


Calle Mayor
Teatro Arenal
Mooz
McDonald's
KFC
Real Casa de Correos
PaPizza
Burger King
Calle de San Ricardo
Plaza de Pontejos
Calle del Marqués Viudo de Pontejos
Calle de Cádiz
Teatro Albéniz
biblioteca
teatro
bar
café
fast-food
restaurante
farmacia
pub
iglesia
cajero
banco

(b)

A.3 Modelo relacional que representa un mapa del corpus

Este modelo relacional representa los casos de estudios de las secciones 6.3.2 y 6.3.3 y representa el mapa de la Figura 6.11.

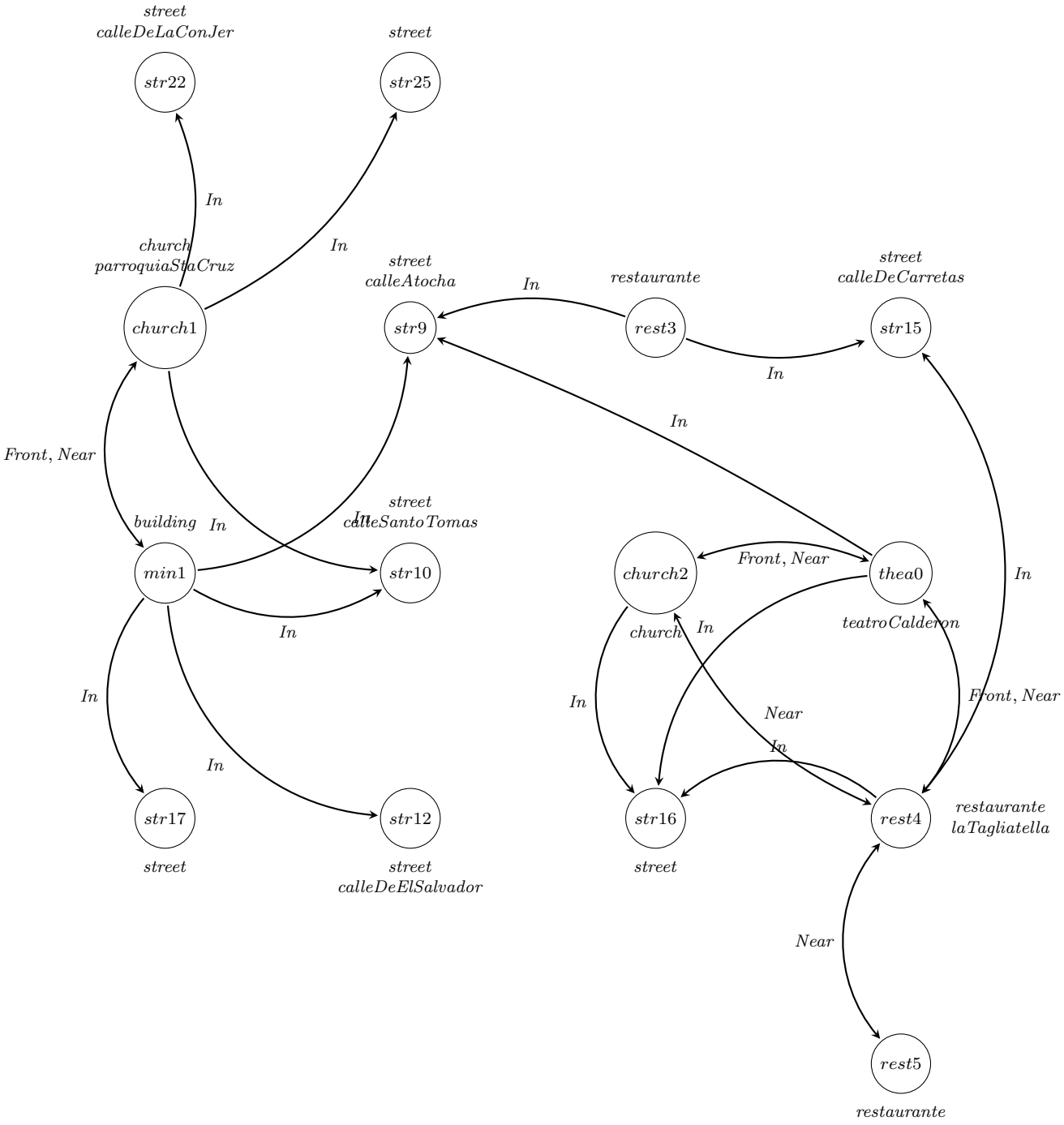


Figura A.4: Modelo relacional del mapa de la Figura 6.11.

Apéndice B

Fórmulas generadas por nuestro algoritmo

En este apéndice mostramos fórmulas que representan expresiones referenciales generadas usando el algoritmo presentado en el Capítulo 4 para los corpora GRE3D7 y ZOOM.

B.1 Generación sobre el corpus GRE3D7

En esta sección mostramos fórmulas que representan expresiones referenciales para los estímulos del corpus GRE3D7 generadas usando el algoritmo presentado en el Capítulo 4. También mostramos las imágenes estímulo del corpus que usamos en esta tesis para la evaluación presentada en el Capítulo 5.

B.1.1 Ejemplos de fórmulas

En la Sección 4.4 se mostró un ejemplo de ejecución del algoritmo para la Figura 1.2a, aquí mostramos todas las fórmulas que dió el algoritmo para el target señalado por la flecha, en 10000 ejecuciones.

	Fórmula	#
1	$\exists ball \wedge \exists large \wedge \exists red$	6003
2	$\exists ball \wedge \exists red$	3672
3	$\exists ball \wedge \exists large$	53
4	$\exists ball \wedge \exists red \wedge \exists rightof$	248
5	$\exists ball \wedge \exists rightof$	15
6	$\exists ball \wedge \exists rightof(\exists cube)$	7
7	$\exists ball \wedge \exists rightof(\exists cube \wedge \exists large)$	1
8	$\exists ball \wedge \exists rightof(\exists cube \wedge \exists red)$	1

Tabla B.1: Fórmulas dadas por el algoritmo para el modelo de la Figura 4.3.

B.1.2 Imágenes verde-azules del corpus

En la Figura B.1 se muestran las escenas mostradas a los participantes que completaron el experimento para la recolección del corpus GRE3D7 introducido en la Sección 2.1.3, esta parte incluye sólo las escenas verde y azules. El corpus contiene otra parte similar con colores rojo y amarillo.

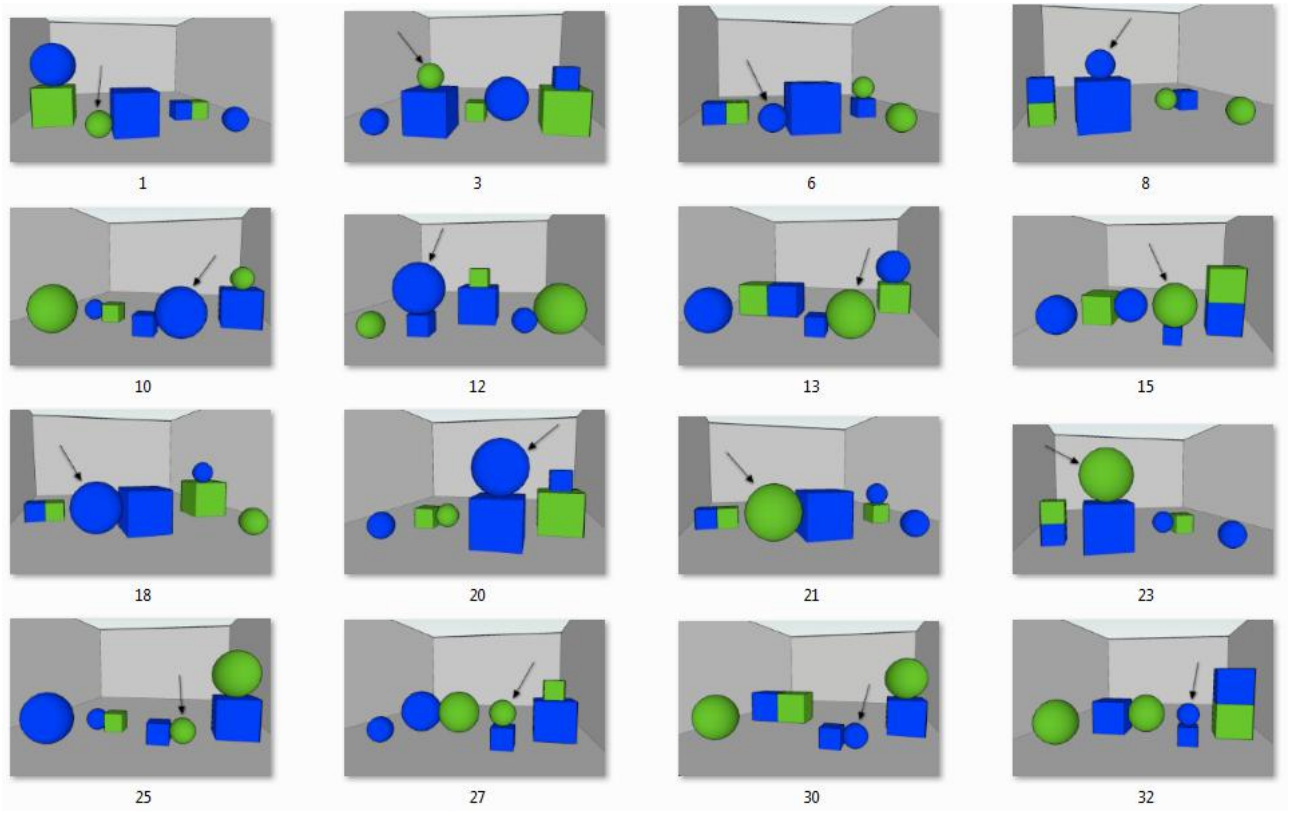


Figura B.1: Imágenes del GRE3D7 parte azul y verde.

B.2 Generación sobre el corpus ZOOM

En la Tabla B.2 se muestran las fórmulas generadas por el algoritmo para el caso de estudio mostrado en la Sección 6.3.2, ejecutando el algoritmo 1000 veces.

	Fórmula	#
1	$\exists parroquiaStaCruz$	570
2	$\exists parroquiaStaCruz.\exists In.(T)$	174
3	$\exists parroquiaStaCruz.\exists In.(\exists calleSantoTomas)$	72
4	$\exists parroquiaStaCruz \wedge \exists Near.(T)$	57
5	$\exists church \wedge \exists parroquiaStaCruz$	48
6	$\exists church \wedge \exists parroquiaStaCruz.\exists In.(T)$	17
7	$\exists parroquiaStaCruz \wedge \exists.In(\exists calleDeLaConJer)$	13
8	$\exists church \wedge \wedge \exists In.(\exists calleDeLaConJer)$	10
9	$\exists In.(\exists calleSantoTomas) \wedge \exists church$	9
10	$\exists church \wedge \exists In.(\exists calleSantoTomas))$	7

Tabla B.2: Fórmulas dadas por el algoritmo para el modelo de la Figura A.4.

Las fórmulas mostradas en la Tabla B.3, fueron generadas por el algoritmo en el caso de estudio presentado en la Sección 6.3.1. Ejecutando el algoritmo 1000 veces.

En la Tabla B.4 se muestran las fórmulas dadas por el algoritmo en 1000 ejecuciones para la Figura 6.12, mostrado en la Sección 6.3.3.

	Fórmula	#
1	$\exists church.(T) \wedge \exists In.(\exists calleSantoTomas))$	234
2	$\exists front.(\exists minAsExt)$	202
3	$\exists church.(T) \wedge \exists In.(\exists calleAtocha)$	87
4	$\exists near.(\exists minAsExt.(T))$	79
5	$\exists front.(\exists front.(\exists church.(T))) \wedge \exists In.(\exists calleSantoTomas))$	74
6	$\exists church.(T) \wedge \exists front.(\exists minAsExt))$	71
7	$\exists church.(T) \wedge \exists front.(\exists minAsExt) \wedge \exists In.(T))$	65
8	$\exists front.(\exists In.(\exists calleSantoTomas))) \wedge \exists church.(T) \wedge \exists In.(T)$	53
9	$\exists front.(\exists minAsExt) \wedge \exists In.(T))$	21
10	$\exists near.(\exists minAsExt.(T) \wedge \exists In.(T))$	15
11	$\exists near.(\exists In.(\exists calleSantoTomas))) \wedge \exists church.(T)$	13
12	$\exists near.(\exists front.(\exists church.(T))) \wedge \exists In.(\exists calleAtocha)$	11
13	$\exists front.(\exists In.(\exists calleSantoTomas))) \wedge \exists church.(T)$	11
14	$\exists church.(T) \wedge \exists In.(\exists calleAtocha) \wedge \exists front.(\exists front.(\exists church.(T)))$	10
15	$\exists near.(\exists near.(\exists church.(T))) \wedge \exists In.(\exists calleSantoTomas))$	8
16	$\exists near.(\exists In.(\exists calleSantoTomas))) \wedge \exists church.(T) \wedge \exists In.(T)$	6
17	$\exists church.(T) \wedge \exists near.(\exists minAsExt.(T) \wedge \exists In.(T))$	6
18	$\exists near.(\exists near.(\exists church.(T) \wedge \exists In.(T))) \wedge \exists In.(\exists calleSantoTomas))$	5
19	$\exists front.(\exists front.(\exists church.(T) \wedge \exists In.(T))) \wedge \exists In.(\exists calleSantoTomas))$	5
20	$\exists In.(\exists calleSantoTomas)) \wedge \exists church.(T)$	4
21	$\exists church.(T) \wedge \exists near.(\exists minAsExt.(T))$	4
22	$\exists near.(\exists front.(\exists church.(T))) \wedge \exists In.(\exists calleSantoTomas))$	4
23	$\exists front.(\exists near.(\exists church.(T))) \wedge \exists In.(\exists calleSantoTomas))$	3
24	$\exists church.(T) \wedge \exists near.(\exists near.(\exists church.(T))) \wedge \exists In.(\exists calleAtocha)$	2
25	$\exists near.(\exists front.(\exists church.(T) \wedge \exists In.(T))) \wedge \exists In.(\exists calleAtocha)$	2
26	$\exists church.(T) \wedge \exists In.(\exists calleAtocha) \wedge \exists near.(\exists near.(\exists church.(T)))$	2
27	$\exists front.(\exists In.(\exists calleSantoTomas))) \wedge \exists near.(\exists near.(\exists church.(T)))$	1
28	$\exists front.(\exists minAsExt) \wedge \exists In.(\exists calleAtocha))$	1
29	$\exists In.(\exists calleAtocha) \wedge \exists church.(T)$	1

Tabla B.3: Fórmulas dadas por el algoritmo para el modelo de la Figura 6.10.

	Fórmula	#
1	$rest_3 = \exists medinaMayrit$ $rest_4 = \exists near.(\exists medinaMayrit)$	303
2	$rest_3 = \exists medinaMayrit$ $rest_4 = \exists near.(\exists near.(\exists medinaMayrit)) \wedge \exists near.(\exists medinaMayrit)$	285
3	$rest_3 = \exists medinaMayrit \wedge \exists In.(T)$ $rest_4 = \exists near.(\exists medinaMayrit \wedge \exists In.(T))$	105
4	$rest_3 = \exists medinaMayrit \wedge \exists In.(\exists calleAtocha)$ $rest_4 = \exists near.(\exists medinaMayrit \wedge \exists In.(\exists calleAtocha))$	83
5	$rest_3 = \exists medinaMayrit \wedge \exists In.(\exists calleAtocha)$ $\wedge \exists near.(\exists near.(\exists medinaMayrit))$ $rest_4 = \exists near.(\exists medinaMayrit)$	77
6	$rest_3 = \exists near.(\exists In.(\exists calleDeLaCruz)) \wedge \exists In.(\exists calleDeCarretas) \wedge$ $\exists In.(\exists calleDeLaCruz) \wedge \exists near.(\exists In.(\exists calleDeCarretas))$	20
7	$rest_3 = \exists medinaMayrit$ $\wedge \exists near.(\exists In.(\exists calleDeLaCruz)) \wedge \exists In.(\exists calleAtocha)$ $rest_4 = \exists near.(\exists medinaMayrit)$	13
8	$rest_3 = \exists In.(\exists calleDeCarretas) \wedge \exists near.(\exists near.$ $(\exists near.(\exists near.(\exists$ $near.(\exists In.(\exists calleDeCarretas))))))$ $rest_4 = \exists near.(\exists In.(\exists calleDeCarretas)$ $\wedge \exists near.(\exists near.(\exists near.(\exists near.$ $(\exists In.(\exists calleDeCarretas)))))))$	9
9	$rest_3 = \exists medinaMayrit \wedge \exists In.(\exists calleAtocha)$ $rest_4 = \exists near.(\exists In.(\exists calleAtocha)$ $\wedge \exists near.(\exists near.(\exists church.(T) \wedge \exists near.(\exists In.(\exists calleAtocha)$ $\wedge \exists near.(\exists near.(\exists church.(T) \wedge \exists In.(T)))))))$	9
10	$rest_3 = \exists near.(\exists In.(\exists calleAtocha) \wedge \exists near.(\exists near.(\exists In.(\exists calleAtocha)$ $\wedge \exists near.(\exists near.(\exists In.(\exists calleDeCarretas))))))) \wedge \exists In.(\exists calleDeCarretas)$ $rest_4 = \exists near.(\exists near.(\exists In.(\exists calleAtocha) \wedge \exists near.(\exists near.$ $(\exists In.(\exists calleAtocha) \wedge \exists near.(\exists near.(\exists In.(\exists calleDeCarretas))))))$ $\wedge \exists In.(\exists calleDeCarretas))$	9

Tabla B.4: Las 10 fórmulas más frecuentes dadas por el algoritmo para el modelo de la Figura 6.10. Tomando como target 2 restaurantes.