Aplicación de técnicas de lenguajes naturales a un sistema de ficción interactiva

Adrián Fraga Fernández

Ultima modificación: 15 de noviembre de 2007

Índice

1. Introducción

Un sistema de ficcion interactiva permite la creación de un mundo virtual sobre el que el usuario interactua utilizando sentencias de lenguaje natural.

Aunque existen sistemas funcionalmente completos de *IF* (interactive fiction), como puede ser el caso de *Inform*, no se utilizan por lo general tecnicas de lenguajes naturales apropiadas.

El resultado es una falta de flexibilidad a la entrada; el usuario debe conocer cómo se comporta el parser (que tipo de entrada se espera) para poder interactuar correctamente con el mundo virtual. Este es el gran problema de la ficcion interactiva; impide que sea alcanzable por el publico general y por tanto comercialmente viable.

En la presente practica se pretende estudiar como las tecnicas vistas en la asignatura de Lenguajes Naturales se pueden aplicar para mejorar el comportamiento de los sistemas actuales de ficción interactiva.

Para esto realizaremos primero una introduccion a la ficcion interactiva (Seccion 2) y al sistema Inform (Seccion 3), que tomaremos como referencia. En las secciones 4 y 5 discutiremos la implementacion en lenguajes Java e Inform de algunas de las tecnicas vistas en clase, y en la seccion 7 realizaremos una critica al sistema *Inform* y hablaremos de como aplicar estas y otras tecnicas de procesamiento de lenguajes naturales no implementadas al problema de la ficción interactiva.

Aunque inicialmente uno de los objetivos de la practica era tambien el crear un reemplazo del parser del sistema Inform, durante la realización de la misma se ha puesto de relevancia que esta tarea no es en absoluto trivial y desborda las intenciones iniciales. Esto se detalla en la sección 6.

Los detalles concretos de las implementaciones discutidas en las secciones 4 y 5, asi como la manera de compilar el codigo y utilizar el sistema Inform, se encuentran en la seccion 8.

2. Introducción a la ficcion interactiva

La historia de la ficción interactiva hasta el momento actual puede, segun [2], dividirse en cuatro etapas: el desarrollo de los primeros trabajos de ficcion interactiva (1972-81), un *boom* comercial de aproximadamente media decada (1982-86), una etapa de relativo abandono (1987-91) y finalmente una gradual recuperación (1992-...), coincidente con la aparicion de la red Internet.

Los primeros trabajos de ficcion interactiva estan ligados al ambito universitario; *Advent*, considerado el primer trabajo de IF, fue escrito en Fortran por Willie Crowter y Don Woods en un DEC[1] PDP-10 y difundido a traves de la red ARPANET[2] (precursora de la actual red Internet y dependiente del departamento de Defensa de Estados Unidos). Aunque en el ambito universitario los trabajos de ficcion interactiva eran conocidos, el coste de los *mainframes* los hacia inalcanzables al publico general.

La popularizacion de estos trabajos vino con la llegada de los microprocesadores en los 80 y la creacion de las primeras compañias de ficción interactiva. El ejemplo mas relevante es la compañia *Infocom* (ver [11]), fundada por miembros del departamento de Informatica del MIT[3], que llego a obtener unos beneficios netos de 10 millones de dolares en 1984 por la venta de trabajos de ficcion interactiva para micros. En europa la compañia inglesa *Level 9* o en España *Aventuras AD* tuvieron un exito similar.

Sin embargo el publico dejo de interesarse por la interaccion via texto con los ordenadores con la llegada de los microprocesadores con capacidades gráficas. A partir de mediados de los 80, las compañias de ficcion interactiva se diversificaron o disolvieron; pero a la vez que iban paulatinamente desapareciendo los trabajos comerciales del mercado, la aparicion de los microprocesadores permitia a programadores ajenos al ambito universitario desarrollar nuevos trabajos (y direcciones) de ficcion interactiva.

Esta comunidad de desarrollo, surgida en los 80, solo cobró fuerza con la posterior aparición de Internet (inicialmente a traves de la red de noticias Usenet y posteriormente diversificandose en otros foros). Aunque se han hecho progresos en el campo, y han surgido distintos proyectos universitarios motivados por este resurgir de la ficcion interactiva (ver [3],[5]), las herramientas y tecnicas utilizadas actualmente siguen siendo herederas *de facto* de los paradigmas desarrollados por las grandes compañías de IF en su "*época dorada*, y, en general, distan de aplicar tecnicas adecuadas de ingenieria.

Esta practica pretende precisamente poner de relevancia distintas tecnicas de ingenieria que aplicadas al ambito de la ficcion interactiva podrian mejorar el comportamiento de las herramientas actuales.

Para una visión mas completa de la historia de la ficcion interactiva, recomiendo consultar [6][4].

[1] Digital Equipment Corporation
[2] Advanced Research Projects Agency Network
[3] Massachussets Institute of Technology
[4] Disponible en la biblioteca de la universidad

3. Estudio previo: El sistema Inform

Los sistemas *TADS*[5] e *Inform* son los dos grandes referentes dentro del campo de la ficcion interactiva. Ambos son considerados por la comunidad asociada a la ficcion interactiva como completos y flexibles; aunque existen diferencias considerables entre ellos, la funcionalidad de ambos sistemas es similar.

Para esta práctica hemos tomado el sistema *Inform* como referencia, y no *TADS*, por las siguientes razones:

- *Inform* siempre ha sido un sistema de codigo abierto, frente a *TADS* que originalmente era un producto comercial. La disponibilidad del codigo fuente o manuales es mayor que en *TADS*. El codigo fuente esta extensivamente comentado, tanto el compilador como la libreria.

- La modificacion del parser en el sistema *TADS* es mas compleja que en el sistema *Inform*, ya que este se encuentra en el interprete[6] y no en la libreria. Como ya se ha comentado en la introduccion de la memoria, uno de los objetivos iniciales de esta practica era la modificacion de dicho parser, por lo que la utilizacion de Inform era casi mandatoria.

  Una consecuencia de que el parser del sistema *Inform* sea mas facil de modificar es que, a diferencia de *TADS*, Inform haya sido traducido a diversos idiomas, entre ellos el Español, Francés, o Alemán.

3.1. Componentes del sistema Inform

El objetivo del sistema *Inform* es codificar una descripcion de un mundo virtual (compuesto por objetos y reglas que afectan a dichos objetos) en un programa ejecutable por el usuario.

El principal componente del sistema *Inform* es por tanto su compilador, que genera codigo para dos maquinas virtuales distintas: la maquina virtual Z-Machine[7] (16 bits) y la maquina virtual GlulX (32 bits). Para ejecutar un programa compilado es por tanto necesario un interprete; para el caso de la maquina virtual Z-Machine existen varios, de los que destacaremos *Frotz*[8], mientras que para GlulX debemos forzosamente utilizar el interprete *GlulXe*.

El sistema proporciona tambien una libreria (en codigo *Inform*) que contiene el parser, encargado de procesar la entrada del usuario, y lo que en [2] se denomina *World Model*, esto es, un conjunto de reglas que modelan comportamientos por defecto y clases primitivas de objetos.

[5] Text Adventure Development System

[6] Ambos sistemas generan codigo para maquinas virtuales, con lo que es necesario un interprete, a semejanza del lenguaje Java.

[7] La importacia de la maquina virtual Z-Machine es de tipo historico: se trata de la misma maquina virtual utilizada por Infocom en los años 80.

[8] *Frotz* tambien es parte de una distribucion Debian.

3.2. Parsing en Inform

En esta seccion daremos una visión global sobre cómo se procesa la entrada de usuario en el sistema *Inform*, extraida de [2], con el objeto de poder comparar las soluciones proporcionadas por el sistema con la tecnicas vistas en clase, y proponer mejoras basadas en estas últimas.

Ya que la intención de este apartado no es estudiar en detalle cómo funciona internamente el parser, sino conceptualizar cómo se realiza el parsing, que efectividad tiene, y la similitud con las tecnicas vistas en clase, daremos una descripción simplificada de lo que realmente hace el sistema; no hay una separación real entre análisis léxico, sintáctico y generación de respuesta, sino que se trata de un todo algorítmico. Separaremos los conceptos simplemente para una mejor compresión del funcionamiento del parser.

3.2.1. Analisis lexico

Lo primero que hace el parser *Inform* es leer el texto tecleado por el usuario y partirlo en palabras (tokens), asociando cada token con una entrada de diccionario, que contiene las palabras que el usuario es susceptible de introducir como entrada:

- Verbos, normalmente en infinitivo[9], y preposiciones; se suelen definir en la libreria.
- Nombres de objetos (correspondientes a la propiedad `name` de los objetos)
- En general, cualquier cadena que se introduzca entre comillas simples (en vez de las comillas dobles utilizadas por defecto) en cualquier parte del codigo *Inform*.

Aunque cierta información de etiquetado se puede añadir manualmente (mediante la sintaxis `'palabra/etiqueta'`) no es una practica común; dicha informacion se suele extraer por lo general de las propiedades de los objetos cuyo nombre coincide con dicho token[10], como veremos en la seccion 3.2.3.

3.2.2. Analisis sintáctico

Inform soporta varias construcciones sintacticas basicas, pero por simplicidad analizaremos en un principio solamente la construccion mas representativa (y mas comunmente utilizada), denominada *action phrase*. El resto de construcciones se utilizan para hablar con actores o para responder a respuestas retóricas del parser.

Una *action phrase* se compone de una serie de *verb phrases* separadas por comas o por la palabra `then`. Por su parte, cada *verb phrase*, o *frase verbal* se compone de un verbo en infinitivo seguido de una *linea de gramatica*, que a su vez se compone de una seria de *frases nominales* separadas normalmente por preposiciones[11].

Cada una de estas *frases nominales* es una lista de *frases nominales basicas* separadas por las conjunciones `','`, `'and'`, `'then'` o las disyunciones `'but'` y `'except'`.

[9] Que en inglés coincide con el participio presente.

[10] Por ejemplo ante la entrada '`wizened/man/,/eat/the/tortoise`', una vez realizada la particion en tokens y comprobado que las palabras estan en el diccionario, si queremos saber si el token `'tortoise'` es masculino o femenino debemos buscar el objeto cuyo atributo `name` es `'tortoise'`, y comprobar si los atributos `male` o `female` son verdaderos.

[11] Ver seccion 3.2.3 para mas detalles.

A su vez, cada *frase nominal basica* se compone de una lista de *descriptores* seguida por una lista de nombres[12]

En concreto, se consideran cinco tipos de descriptores:

Articulos	Se contemplan el articulo indefinido `'the'` y los articulos definidos `'a'`, `'an'`, `'some'`.
And-words	`'and'`
Other-words	`'other'`[13], `'another'`
Numerales	`'seven'`, etc.
Adjetivos posesivos	`'my'`, `'this'`, `'these'`, `'that'`, `'those'`.
All-words	`'all'`, `'each'`, `'every'`, `'everything'`, `'both'`.

Y tres tipos de nombres:

Nombres de objetos	Palabras que coinciden con la propiedad `name` de algun objeto.
Me-words	Palabras que se refieren al actor principal[14]. En concreto se consideran `'me'`, `'self'`, `'myself'`.
Pronombres	Como `'it'`, `'him'`, `'her'`, `'them'`.

Debe notarse que *Inform* no diferencia adjetivos de nombres hasta la fase de analisis semántico; los adjetivos se encuentran tambien en el campo `name` de los objetos, por lo que son considerados como tales durante esta etapa.

Asi por ejemplo la *frase nominal basica* `'the balloon'` tiene un descriptor y un nombre, mientras que `'red balloon'` tiene dos nombres. Otras *frases nominales* validas son: `'it'`, `'rucksack'`, `'brown bag, pepper'`, `'a box and the other compass'`, `'nine silver coins'`, `'smooth stones'` o `'everything except the rucksack'`.

Algunas construcciones gramaticales que no soporta el parser *Inform* básico (aunque hay maneras de emularlas)[15]:

- *Adverbios*: Por ejemplo `'quickly'` en `'run quickly east'`.

[12] Articulando un poco todo esto:

$$
\begin{aligned}
\langle \text{action phrase} \rangle &= \langle \text{frase verbal} \rangle\ \langle \text{','/'then} \rangle\ \langle \text{frase verbal} \rangle\ \langle \text{','/'then} \rangle \ldots \\
\langle \text{frase verbal} \rangle &= \langle \text{verbo} \rangle\ \langle \text{linea de gramatica} \rangle \\
\langle \text{linea de gramatica} \rangle &= \langle \text{frase nominal} \rangle\ \langle \text{preposicion} \rangle\ \langle \text{frase nominal} \rangle\ \langle \text{preposicion} \rangle \ldots \\
\langle \text{frase nominal} \rangle &= \langle \text{frase nominal basica} \rangle\ \langle \text{','/'and'/'then'/'but'/'except'} \rangle \ldots \\
\langle \text{frase nominal basica} \rangle &= \langle \text{descriptor} \rangle \langle \text{descriptor} \rangle \ldots \langle \text{nombre} \rangle \langle \text{nombre} \rangle \ldots
\end{aligned}
$$

[13] Por ejemplo: `'other than the one i am holding'`

[14] Realmente se tratan de pronombres relativos, pero Inform los inserta en una categoria propia

[15] Habria que preguntarse si estas construcciones gramaticales no se contemplan porque son dificiles de detectar o porque la comprension del enunciado y/o la generacion de una respuesta ante tales frases no es tan trivial como en los casos sí contemplados.

- *Genitivos*: Los objetos no son llamados normalmente por sus circunstancias. `'the box on the floor'` o `'the priest's hat'` por lo general devolveran un error de parsing.
- *Pronombres no comtemplados inicialmente*: Como `'I'` en `'I am happy'`, `'what'` en `'what are you doing?'`, `'that'` en `'eat that'`, pronombres posesivos que aparecen en solitario como `'mine'`[16], etc.
- *Pronominal adverbs* Como `'under it'`, ya que no son comunes en ingles[17]

3.2.3. Analisis semantico

Como se menciona en la sección anterior, en *Inform* cada verbo se asocia con una o mas *lineas de gramatica*[18] que deben coincidir con el texto que sigue al verbo e indican qué rutina del parser ejecutar si se encuentra dicha coincidencia.

La *comprension* de un enunciado se corresponde, precisamente, con la ejecución de dicha rutina.

Por ejemplo los verbos `'take'`, `'get'`, `'carry'` o `'hold'` se asocian con las siguientes *lineas de gramatica*:

```
Verb 'take' 'get' 'carry' 'hold'
        * 'out'                              -> Exit
        * multi                              -> Take
        * multiinside 'from' noun            -> Remove
        * 'in' noun                          -> Enter
        * multiinside 'off' noun             -> Remove
        * 'off' held                         -> Disrobe
        * 'inventory'                        -> Inv;
```

Cada linea se compone de varios tokens (nomalmente preposiciones) y/o directivas, que indican las propiedades que debe cumplir la entrada correspondiente a dicha posición; la 'entrada' puede ser una palabra o una *frase nominal*, dependiendo de la directiva.

Por ejemplo:

[16]El parser reconoceria `'take the man's pencil, give him my pencil'`, pero no `'give him mine'`

[17]La libreria inform para otros idiomas distintos del ingles si soporta este tipo de adverbios.

[18]La libreria base define *lineas de gramática* para aproximadamente 100 verbos de habla inglesa de uso común en la ficcion interactiva, ampliables por el usuario

Directiva	Entrada	Propiedades que debe cumplir la entrada
`noun`	*Frase nominal*	Un unico objeto (ej: `'globo'`), alcanzable por el actor principal.
`held`	*Frase nominal*	Un unico objeto que es propiedad del actor principal y que éste puede utilizar para realizar acciones.
`multi`	*Frase nominal*	Uno o varios objetos (ej: `'globo'`, `'todos los globos'`, `'todos los globos excepto el verde'`, etc) alcanzables por el actor principal.
`multiheld`	*Frase nominal*	Uno o varios objetos, que son propiedad del actor principal y que éste puede utilizar para realizar acciones.
`multiinside`	*Frase nominal*	Uno o varios objetos contenidos en el siguiente objeto presente en la linea de gramatica.
`<rutina>`	*Palabra*	Cualquier texto aceptado por una rutina proporcionada por el usuario
`number`	*Palabra*	Un numero

Una vez reconocida la estructura sintactica de la frase, se intenta, de manera secuencial, emparejar la *frase verbal* con cada una de las lineas de gramatica disponible, identificando los objetos representados por las *frases nominales basicas*, resolviendo ambigüedades si es necesario[19], y comprobando que se cumplen los requisitos especificados por la directiva. En caso afirmativo, se ejecuta la rutina asociada.

Adicionalmente, y antes de llamar a la rutina correspondiente, a cada *frase nominal basica* se le asignan distintas propiedades que pueden ser de utilidad tanto para la comprensión como para la generación de respuesta[20]:

- *Definicion*: Una *frase nominal basica* es *definida* si no lleva articulo (o un articulo definido no asociado a un *all-word* o a una cantidad numeral).
- *Cantidad*: Esta propiedad se refiere al numero de objetos referenciados, que por lo general es la unidad, pero podria tomar otros numerales como 7 en `'seven stones'` o infinito[21] en `'as many as possible'` o `'all the stones'`.
- *Genero*: Puede ser masculino, femenino, o neutro, dependiendo de los atributos `male`, `female`, `neuter` de los objetos involucrados en la frase.
- *Animacion*: Sirve para distinguir entre actores (atributo `'animate'`) y objetos inanimados.

[19] Aunque no se contempla la posibilidad de ambigüedad durante el análisis sintáctico, si se contempla que una misma frase nominal pueda referirse a mas de un objeto presente en la escena (por ejemplo `'take ball'` cuando hay dos bolas de colores distintos presentes en la escena, o ante constucciones como `'take it'`). En caso de ambigüedad el parser puede responder con una pregunta retórica o utilizar un sistema de puntos para resolver cual es el objeto mas susceptible de ser referenciado.

[20] Estas propiedades tambien se utilizan para la resolución de ambigüedad semántica.

[21] Representado internamente por el numero 100.

3.2.4. Generacion de respuesta

Una vez reconocida semánticamente la frase y los objetos referenciados en la misma, se ejecuta la rutina concreta del parser determinada en la *linea de gramatica*, que actua en consecuencia. Para la generacion de respuesta se utilizarán tanto los atributos de los objetos referenciados como los presentes en la escena. Asi por ejemplo los objetos cuyo nombre es plural deben tener el atributo `pluralname`, de manera que se hagan los cambios adecuados en la salida (en este caso concreto el parser cambiaría (ad-hoc) `''You can't move that''` por `''You can't move those''`).

Inform tambien proporciona una serie de funciones que permiten imprimir un objeto acompañado del articulo adecuado en cada caso:

`print (a) objeto`	Imprime el objeto con el articulo indefinido[22]
`print (the) objeto`	Imprime el objeto con el articulo definido[23]
`print (name) objeto`	Imprime solo el nombre del objeto (sin adjetivos)
`print (The) objeto`	Variante con letras mayusculas
`print (A) objeto`	Variante con letras mayusculas

Asi, por ejemplo, `print (a) brasslamp` imprimiria por pantalla `''an old brass lamp''`.

Finalmente, comentar que si el objeto tiene el atributo `proper` esto significa que se trata de un nombre propio y por tanto no lleva articulo.

3.2.5. Un ejemplo de funcionamiento

Otra construcción básica (no vista hasta el momento por simplicidad) es la utilizada para conversar con otros actores presentes en el mundo virtual. Se compone de una *frase nominal*, seguida por una *frase verbal*, donde ambas estan separadas por una coma; es decir:

{`noun phrase`}/{`,`}/{`verb phrase`}

El resultado de analizar la frase `'Conan, put every sword into the box'`, asumiendo que el verbo `'put'` contiene la *linea de gramatica* {`* multiexcept 'in' / 'inside' / 'into' noun ->Insert`} se puede ver en la figura 1.

[23]En este caso, el articulo se determina automaticamente (segun el nombre del objeto empiece por vocal o consonante); en el caso de que el objeto tenga el atributo `pluralname` se cambia `'a'` por `'some'`. Adicionalmente se permite establecer (con fines estéticos) un articulo especifico para un objeto concreto utilizando el atributo `article` del objeto (por ejemplo `''some bundles of''`, `''far too many''` y similares.

[23]El articulo definido siempre es `'the'`, excepto para nombres propios.

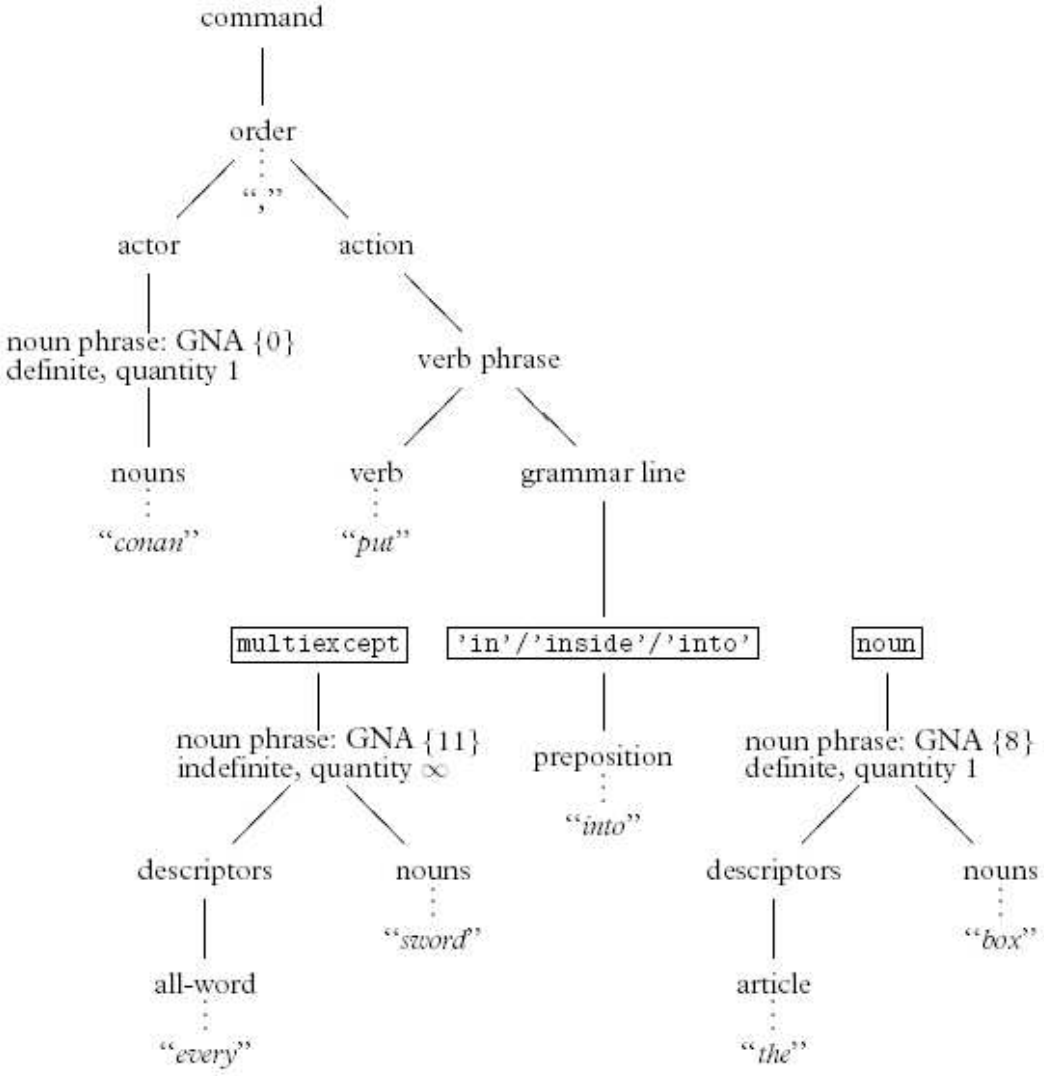

Figura 1: Resultado de analizar la frase `'Conan, put every sword into the box'`

4. Obtencion de una gramática probabilistica

4.1. Eleccion de un corpus

Como ya se ha comentado, uno de los objetivos iniciales de esta practica era crear un parser funcional de reemplazo para el sistema *Inform*[24]. Por tanto era mandatorio el tener acceso a un corpus de calidad, interesandonos en gran medida la información de lema para mejorar el comportamiento en la respuesta[25].

Aunque se dispone de versiones depuradas del corpus SUSANNE preparadas por los profesores de la asignatura, estas no disponen de información de lema, por lo que se ha optado por la utilización directa del corpus SUSANNE[26], que sí proporciona esta información). De todas maneras la implementación se ha hecho lo menos dependiente posible del corpus de entrada, para posteriormente poder utilizar otro corpus o incluso utilizar la version depurada del corpus SUSANNE tomando la información de lema del corpus sin depurar.

4.2. Obtención de la gramática

Para la obtención de la gramática se ha realizado un programa en lenguaje Java que lee una o varias entradas con la sintaxis del corpus Susanne y genera las siguientes salidas para un fichero de entrada `'corpus'`:

Salidas textuales

`corpus.tags.txt`	Relación textual del conjunto de etiquetas (terminales y no terminales)
`corpus.lexicon.txt`	Descripción textual del lexicon (nombre, lema, etiqueta, numero de ocurrencias y probabilidad de cada palabra)
`corpus.lexicon.automata.txt`	Descripción textual del AFND[27]utilizado para representar eficientemente el lexicon (estado, altura, indice, transiciones).
`corpus.gramatica.txt`	Descripción textual de la gramática probabilistica (pruducciones, numero de ocurrencias, probabilidades).

[24]Ver seccion 6.
[25]Ver sección 7
[26]En su version 5 (Agosto 2000).
[27]Automata finito determinista numerado

`corpus.tags.offsets`	Array con el offset de cada etiqueta (representada por un entero) dentro del `corpus.tag.data`.
`corpus.tags.data`	Informacion de etiquetas.
`corpus.gramatica.offsets`	Array con el el offset de comienzo –dentro de `corpus.gramatica.data`– de las reglas que tienen una etiqueta concreta como raiz.
`corpus.gramatica.data`	Informacion de reglas.
`corpus.lexicon.automata.offsets`	Array con el offset de cada estado dentro de `corpus.lexicon.automata.data`
`corpus.lexicon.automata.indices`	Array con la numeración de cada estado. Es utilizado por las funciones `indice_a_cadena` y `cadena_a_indice` para obtener un índice unívoco para cada palabra contenida en el autómata.
`corpus.lexicon.automata.data`	Información de transiciones entre estados
`corpus.lexicon.tabla.offsets`	Array con el offset de comienzo de las disintas etiquetas para un indice (que representa una palabra del lexicon).
`corpus.lexicon.tabla.tags`	Informacion de etiquetas (representadas por enteros) para cada palabra.
`corpus.lexicon.tabla.probabilidades`	Informacion de probabilidades para las distintas etiquetas de cada palabra.

Como ya se observa en las tablas, se ha utilizado un AFND mínimo para obtener una representación óptima de la información contenida en el lexicon. La estrategia de minimización empleada ha sido la incremental propuesta por [8] (tambien descrita en [1])[28].

La información probabilística, en el caso de la salida para las maquinas virtuales *Z* y *GlulX*, no ha sido posible realizarla en coma flotante ya que estas maquinas no soportan punto flotante y tampoco existen librerias para emularlas[29], por lo que ha sido necesario emular decimales por coma fija: 0.16 en el caso de *Z* o 0.32 en el caso de *GlulX*, con lo que podemos representar numeros entre 1 y $2^{-16} \approx 1{,}52e^{-5}$ o entre 1 y $2^{-32} \approx 5e^{-20}$ respectivamente.

4.3. Resultados obtenidos para un treebank ejemplo

Utilizando el treebank `pruebaDiccionario` (ver seccion 8.2 (extraido del corpus SUSANNE):

[28] Aunque los algoritmos subyacentes son eficientes, la profusa utilización del sistema Java (basado en un paradigma de ejecución virtual) y de sus funciones de libreria estandar provoca que los resultados, en tiempo de ejecución, sean bastante menos eficientes que los comentados en [1].

[29] Esto es una verdad a medias, ya que sí existen para la maquina *Z*, pero no son extrapolables a *GlulX* y deberiamos generar entonces distinto código para cada máquina, algo que queremos evitar en la medida de lo posible.

```
A01:0010.06  - AT      The             the             [O[S[Nns:s.
A01:0010.09  - NP1s    Fulton          Fulton          [Nns.
A01:0010.12  - NNL1cb  County          county          .Nns]
A01:0010.15  - JJ      Grand           grand           .
A01:0010.18  - NN1c    Jury            jury            .Nns:s]
A01:0010.21  - VVDv    said            say             [Vd.Vd]
A01:0010.24  - NPD1    Friday          Friday          [Nns:t.Nns:t]
A01:0010.27  - AT1     an              an              [Fn:o[Ns:s.
A01:0010.30  - NN1n    investigation   investigation   .
A01:0020.03  - IO      of              of              [Po.
A01:0020.06  - NP1t    Atlanta         Atlanta         [Ns[G[Nns.Nns]
A01:0020.09  - GG      +<apos>s        -               .G]
A01:0020.12  - JJ      recent          recent          .
A01:0020.15  - JJ      primary         primary         .
A01:0020.18  - NN1n    election        election        .Ns]Po]Ns:s]
A01:0020.21  - VVDv    produced        produce         [Vd.Vd]
A01:0020.24  - YIL     <ldquo> -       .
A01:0020.27  - ATn     +no             no              [Ns:o.
A01:0020.30  - NN1u    evidence        evidence        .
A01:0020.33  - YIR     +<rdquo>        -               .
A01:0020.39  - CST     that            that            [Fn.
A01:0030.03  - DDy     any             any             [Np:s.
A01:0030.06  - NN2     irregularities  irregularity    .Np:s]
A01:0030.09  - VVDv    took            take            [Vd.Vd]
A01:0030.12  - NNL1c   place           place           [Ns:o.Ns:o]Fn]Ns:o]Fn:o]S]
A01:0030.15  - YF      +.              -               .O]
```

Obtenemos el siguiente diccionario:

```
1       "               null            YIL     1       0.03846154
1       "               null            YIR     1       0.03846154
2       's              null            GG      1       0.03846154
3       .               null            YF      1       0.03846154
4       Atlanta         Atlanta         NP1t    1       0.03846154
5       County          county          NNL1cb  1       0.03846154
6       Friday          Friday          NPD1    1       0.03846154
7       Fulton          Fulton          NP1s    1       0.03846154
8       Grand           grand           JJ      1       0.03846154
9       Jury            jury            NN1c    1       0.03846154
10      The             the             AT      1       0.03846154
11      an              an              AT1     1       0.03846154
12      any             any             DDy     1       0.03846154
13      election        election        NN1n    1       0.03846154
14      evidence        evidence        NN1u    1       0.03846154
15      investigation   investigation   NN1n    1       0.03846154
16      irregularities  irregularity    NN2     1       0.03846154
17      no              no              ATn     1       0.03846154
18      of              of              IO      1       0.03846154
19      place           place           NNL1c   1       0.03846154
20      primary         primary         JJ      1       0.03846154
21      produced        produce         VVDv    1       0.03846154
22      recent          recent          JJ      1       0.03846154
23      said            say             VVDv    1       0.03846154
24      that            that            CST     1       0.03846154
25      took            take            VVDv    1       0.03846154
```

Y la siguiente gramática probabilística:

```
0  1  1.0  (#) -> (S) <eof>
1  1  1.0  (Fn) -> CST (Np:s) (Vd) (Ns:o)
2  1  1.0  (Fn:o) -> (Ns:s) (Vd) YIL (Ns:o)
3  1  1.0  (G) -> (Nns) GG
4  1  0.5  (Nns) -> NP1s NNL1cb
5  1  0.5  (Nns) -> NP1t
6  1  1.0  (Nns:s) -> AT (Nns) JJ NN1c
7  1  1.0  (Nns:t) -> NPD1
8  1  1.0  (Np:s) -> DDy NN2
9  1  1.0  (Ns) -> (G) JJ JJ NN1n
10  1  0.5  (Ns:o) -> ATn NN1u YIR (Fn)
11  1  0.5  (Ns:o) -> NNL1c
12  1  1.0  (Ns:s) -> AT1 NN1n (Po)
13  1  1.0  (O) -> (S) YF
14  1  1.0  (Po) -> IO (Ns)
15  1  1.0  (S) -> (Nns:s) (Vd) (Nns:t) (Fn:o)
16  3  1.0  (Vd) -> VVDv
```

Notese que se ha añadido una produccion artificial `(#) ->(S) <eof>` para utilizar como test de parada duante el análisis sintáctico.

4.4. Resultados obtenidos en la minimizacion de un automata ejemplo

Daremos otro ejemplo adicional, ya visto en [1], para comprobar el buen funcionamiento del algorimo de minimización. Nuestra entrada será el treebank `pruebaAutomata`, que tiene una sola produccion y las palabras *discount, dismount, recount, remount, discounted, dismounted, recounted, remounted , discounting, dismounting, recounting, remounting, discounts, dismounts, recounts* y *remounts*.

El autómata resultante es el siguiente:

```
Estado 0  (h=12,i=16) d->1 r->14
Estado 1  (h=11,i=8) i->2
Estado 2  (h=10,i=8) s->3
Estado 3  (h=9,i=8) c->4 m->4
Estado 4  (h=8,i=4) o->5
Estado 5  (h=7,i=4) u->6
Estado 6  (h=6,i=4) n->7
Estado 7  (h=5,i=4) t->8
Estado 8  (h=4,i=4) #->9 e->10 i->12 s->11
Estado 9  (h=0,i=1) (vacio)
Estado 10 (h=2,i=1) d->11
Estado 11 (h=1,i=1) #->9
Estado 12 (h=3,i=1) n->13
Estado 13 (h=2,i=1) g->11
Estado 14 (h=10,i=8) e->3
```

Donde el estado 0 es el estado inicial y el 9 es evidentemente el estado final –ya que no presenta transiciones–.

5. Analisis sintactico

5.1. Elección de un algoritmo de análisis sintáctico

Para el análisis sintáctico hemos utilizado el algoritmo de Earley descrito en [12], ya que es un algoritmo relativamente simple, produce resultados adecuados, y lo que es mas importante, se puede utilizar directamente con el corpus SUSANNE.

En un principio el algoritmo está diseñado para que almacene en la tabla todos los posibles árboles sintácticos resultantes de analizar la frase, pero es posible cambiar este comportamiento para que sólamente almacene el árbol sintactico mas probable[30].

5.2. Obtención del arbol de análisis sintáctico

La implementación del algoritmo de Earley se ha realizado en lenguaje Inform, y es posible compilar el programa tanto para la maquina virtual *Z* como para *GlulX*. Para comprobar los resultados del análisis de una cadena de entrada para un corpus concreto, la información extraida del corpus por el lenguaje Java del programa anterior debe ser compilada en un ejecutable Inform junto con las rutinas de análisis sintáctico, y ejecutada[31].

El programa resultante pide una entrada de usuario y proporciona[32]:

- La separación de la cadena de entrada en tokens y la obtención de las distintas etiquetas posibles para cada token.
- La tabla de análisis generada por el algoritmo de Earley, indicando en que fase del algoritmo (`init`, `predictor`, `scanner`, `completer`) se ha generado cada entrada de la tabla, la probabilidad actual para cada entrada, y la información de referencia necesario para reconstruir posteriormente el arbol sintáctico.
- Los análisis sintácticos que se han completado con éxito presentes en la tabla, o un mensaje en caso de no existir ninguno.

5.3. Resultados obtenidos para una sentencia ejemplo

Para comprobar los resultados obtenidos por el programa, utilizaremos el siguiente treebank (`pruebaParsing`):

[30]Simplemente editar el archivo `analisisSintactico` y descomentar la linea –al principio del archivo– que define la constante `SoloMasProbable`. Esta es una solucion poco elegante, pero mandatoria, ya que no existe información acerca de la linea de comandos en Inform.

[31]Ha sido la unica solucion posible ya que inform no tiene funciones para trabajar con archivos de datos binarios. Ver seccion 6 para mas detalles

[32]Para el caso de la maquina virtual *GlulX* esto no ha sido posible y es necesario introducir tambien la cadena a mano en el codigo. Ver seccion 8.3 para mas detalles

```
- - Np Juan            - [S[S[NP.]
- - V  vio             - [VP.
- - D  un              - [NP.
- - N  hombre          - .]]]
- - P  con             - [PP.
- - D  un              - [NP.
- - N  telescopio      - .]]]
- - Np Juan            - [S[NP.]
- - V  vio             - [VP.
- - D  un              - [NP[NP.
- - N  hombre          - .]
- - P  con             - [PP.
- - D  un              - [NP.
- - N  telescopio      - .]]]]]
```

Que genera el lexicon:

```
1       con             null    P       2       0.14285715
2       hombre          null    N       2       0.14285715
3       juan            null    Np      2       0.14285715
4       telescopio      null    N       2       0.14285715
5       un              null    D       4       0.2857143
6       vio             null    V       2       0.14285715
```

Y la siguiente gramática probabilística:

```
0       1       1.0                     (#) -> (S) <eof>
1       1       0.1428571428571428      (NP) -> (NP) (PP)
2       4       0.5714285714285714      (NP) -> D N
3       2       0.2857142857142857      (NP) -> Np
4       2       1.0                     (PP) -> P (NP)
5       2       0.6666666666666666      (S) -> (NP) (VP)
6       1       0.3333333333333333      (S) -> (S) (PP)
7       2       1.0                     (VP) -> V (NP)
```

Evidentemente, este treebank esta generado de forma artificial para utilizar el ejemplo ambiguo:

```
> juan vio un hombre con un telescopio
```

Visto en clase con anterioridad. El resultado de analizar léxicamente esta frase con nuestro programa es:

```
juan [Np] vio [V] un [D] hombre [N] con [P] un [D] telescopio [N]
```

Y la tabla de análisis obtenida para el algoritmo de Earley:

```
<0   > (Init     ) [0,0] (#) ->  . (S) <eof>      1.0000000000
<25  > (Predictor) [0,0] (S) ->  . (NP) (VP)      0.0666666686
<50  > (Predictor) [0,0] (S) ->  . (S) (PP)       0.0333333343
<75  > (Predictor) [0,0] (NP) ->  . (NP) (PP)     0.0142857149
<100 > (Predictor) [0,0] (NP) ->  . D N           0.0571428596
<125 > (Predictor) [0,0] (NP) ->  . Np            0.0285714298
<146 > (Scanner  ) [0,1] (NP) -> Np .             0.0040816330    <Np='juan'>
<167 > (Completer) [0,1] (S) -> (NP)  . (VP)      0.0027210887    <(NP)=146>
<192 > (Completer) [0,1] (NP) -> (NP)  . (PP)     0.0005830904    <(NP)=146>
<217 > (Predictor) [1,1] (VP) ->  . V (NP)        1.0000000000
<242 > (Predictor) [1,1] (PP) ->  . P (NP)        1.0000000000
<267 > (Scanner  ) [1,2] (VP) -> V  . (NP)        0.0142857149    <V='vio'>
<292 > (Predictor) [2,2] (NP) ->  . (NP) (PP)     0.0142857149
<317 > (Predictor) [2,2] (NP) ->  . D N           0.0571428596
<342 > (Predictor) [2,2] (NP) ->  . Np            0.0285714298
<363 > (Scanner  ) [2,3] (NP) -> D  . N           0.0163265320    <D='un'>
<388 > (Scanner  ) [2,4] (NP) -> D N .            0.0023323618    <D='un', N='hombre'>
<413 > (Completer) [1,4] (VP) -> V (NP) .         0.0003331945    <V='vio', (NP)=388>
<438 > (Completer) [2,4] (NP) -> (NP)  . (PP)     0.0003331945    <(NP)=388>
<463 > (Completer) [0,4] (S) -> (NP) (VP) .       0.0000090665    <(NP)=146, (VP)=413>
<488 > (Completer) [0,4] (#) -> (S)  . <eof>      0.0000090665    <(S)=463>
<513 > (Completer) [0,4] (S) -> (S)  . (PP)       0.0000030221    <(S)=463>
<538 > (Predictor) [4,4] (PP) ->  . P (NP)        1.0000000000
<563 > (Scanner  ) [4,5] (PP) -> P  . (NP)        0.0142857149    <P='con'>
<588 > (Predictor) [5,5] (NP) ->  . (NP) (PP)     0.0142857149
<613 > (Predictor) [5,5] (NP) ->  . D N           0.0571428596
<638 > (Predictor) [5,5] (NP) ->  . Np            0.0285714298
<659 > (Scanner  ) [5,6] (NP) -> D  . N           0.0163265320    <D='un'>
<684 > (Scanner  ) [5,7] (NP) -> D N .            0.0023323618    <D='un', N='telescopio'>
<709 > (Completer) [4,7] (PP) -> P (NP) .         0.0003331945    <P='con', (NP)=684>
<734 > (Completer) [5,7] (NP) -> (NP)  . (PP)     0.0003331945    <(NP)=684>
<759 > (Completer) [2,7] (NP) -> (NP) (PP) .      0.0000011101    <(NP)=388, (PP)=709>
<784 > (Completer) [0,7] (S) -> (S) (PP) .        0.0000000100    <(S)=463, (PP)=709>
<809 > (Completer) [1,7] (VP) -> V (NP) .         0.0000001585    <V='vio', (NP)=759>
<834 > (Completer) [2,7] (NP) -> (NP)  . (PP)     0.0000001585    <(NP)=759>
<859 > (Completer) [0,7] (#) -> (S)  . <eof>      0.0000000100    <(S)=784>
<884 > (Completer) [0,7] (S) -> (S)  . (PP)       0.0000000033    <(S)=784>
<909 > (Completer) [0,7] (S) -> (NP) (VP) .       0.0000000043    <(NP)=146, (VP)=809>
<934 > (Completer) [0,7] (#) -> (S)  . <eof>      0.0000000043    <(S)=909>
<959 > (Completer) [0,7] (S) -> (S)  . (PP)       0.0000000014    <(S)=909>
```

La anterior tabla contiene dos análisis sintácticos completos:

```
<0.0000000100>[(S)[(S)[(NP) juan] [(VP) vio [(NP) un hombre]]] [(PP) con [(NP) un telescopio]]]
<0.0000000043>[(S)[(NP) juan] [(VP) vio [(NP)[(NP) un hombre] [(PP) con [(NP) un telescopio]]]]]
```

Hay que comentar que para compilar el ejmplo hemos utilizado la maquina virtual *GlulX*, y que por tanto estamos utilizando una precision de coma fija de 0.32 bits. En el caso de haber compilado el mismo programa para la maquina virtual *Z* el resultado obtenido habria sido:

```
<0.0000> [(S) [(S) [(NP) juan] [(VP) vio [(NP) un hombre]]] [(PP) con [(NP) un telescopio]]]
<0.0000> [(S) [(NP) juan] [(VP) vio [(NP) [(NP) un hombre] [(PP) con [(NP) un telescopio]]]]]
```

Lo que nos impide distinguir cual de los dos análisis sintacticos es mas probable. Lamentablemente, como se comenta en la seccion 6, *Inform no proporciona soporte de*

punto flotante, por lo que la degradacion hacia cero de las probabilidades durante el análisis sintáctico, ya un problema de por si, se agrava todavia mas. De hecho, la coma fija utilizada en la máquina *GlulX*, aunque superior en precision a la utilizada en la máquina *Z* se vuelve inadecuada en cuanto tratamos con ejemplos un poco mas complejos, y ya no digamos con el corpus SUSANNE.

5.4. Una variante del ejemplo anterior

Para finalizar, mostraremos una variante del ejemplo anterior, donde solo mantenemos los análisis sintácticos mas probables en cada momento, tal como se comentaba en 5.1:

```
<0   > (Init     ) [0,0] (#) ->  . (S) <eof>      1.0000000000
<25  > (Predictor) [0,0] (S) ->  . (NP) (VP)      0.0666666686
<50  > (Predictor) [0,0] (S) ->  . (S) (PP)       0.0333333343
<75  > (Predictor) [0,0] (NP) ->  . (NP) (PP)     0.0142857149
<100 > (Predictor) [0,0] (NP) ->  . D N           0.0571428596
<125 > (Predictor) [0,0] (NP) ->  . Np            0.0285714298
<146 > (Scanner  ) [0,1] (NP) -> Np .             0.0040816330    <Np='juan'>
<167 > (Completer) [0,1] (S) -> (NP)  . (VP)      0.0027210887    <(NP)=146>
<192 > (Completer) [0,1] (NP) -> (NP)  . (PP)     0.0005830904    <(NP)=146>
<217 > (Predictor) [1,1] (VP) ->  . V (NP)        1.0000000000
<242 > (Predictor) [1,1] (PP) ->  . P (NP)        1.0000000000
<267 > (Scanner  ) [1,2] (VP) -> V  . (NP)        0.0142857149    <V='vio'>
<292 > (Predictor) [2,2] (NP) ->  . (NP) (PP)     0.0142857149
<317 > (Predictor) [2,2] (NP) ->  . D N           0.0571428596
<342 > (Predictor) [2,2] (NP) ->  . Np            0.0285714298
<363 > (Scanner  ) [2,3] (NP) -> D  . N           0.0163265320    <D='un'>
<388 > (Scanner  ) [2,4] (NP) -> D N .            0.0023323618    <D='un', N='hombre'>
<413 > (Completer) [1,4] (VP) -> V (NP) .         0.0003331945    <V='vio', (NP)=388>
<438 > (Completer) [2,4] (NP) -> (NP)  . (PP)     0.0003331945    <(NP)=388>
<463 > (Completer) [0,4] (S) -> (NP) (VP) .       0.0000090665    <(NP)=146, (VP)=413>
<488 > (Completer) [0,4] (#) -> (S)  . <eof>      0.0000090665    <(S)=463>
<513 > (Completer) [0,4] (S) -> (S)  . (PP)       0.0000030221    <(S)=463>
<538 > (Predictor) [4,4] (PP) ->  . P (NP)        1.0000000000
<563 > (Scanner  ) [4,5] (PP) -> P  . (NP)        0.0142857149    <P='con'>
<588 > (Predictor) [5,5] (NP) ->  . (NP) (PP)     0.0142857149
<613 > (Predictor) [5,5] (NP) ->  . D N           0.0571428596
<638 > (Predictor) [5,5] (NP) ->  . Np            0.0285714298
<659 > (Scanner  ) [5,6] (NP) -> D  . N           0.0163265320    <D='un'>
<684 > (Scanner  ) [5,7] (NP) -> D N .            0.0023323618    <D='un', N='telescopio'>
<709 > (Completer) [4,7] (PP) -> P (NP) .         0.0003331945    <P='con', (NP)=684>
<734 > (Completer) [5,7] (NP) -> (NP)  . (PP)     0.0003331945    <(NP)=684>
<759 > (Completer) [2,7] (NP) -> (NP) (PP) .      0.0000011101    <(NP)=388, (PP)=709>
<784 > (Completer) [0,7] (S) -> (S) (PP) .        0.0000000100    <(S)=463, (PP)=709>
<809 > (Completer) [1,7] (VP) -> V (NP) .         0.0000001585    <V='vio', (NP)=759>
<834 > (Completer) [2,7] (NP) -> (NP)  . (PP)     0.0000001585    <(NP)=759>
<859 > (Completer) [0,7] (#) -> (S)  . <eof>      0.0000000100    <(S)=784>
<884 > (Completer) [0,7] (S) -> (S)  . (PP)       0.0000000033    <(S)=784>
<909 > (Completer) [0,7] (S) -> (NP) (VP) .       0.0000000043    <(NP)=146, (VP)=809>
```

El único arbol sintactico obtenido en este caso es, evidentemente:

```
<0.0000000100>[(S)[(S)[(NP) juan] [(VP) vio [(NP) un hombre]]] [(PP) con [(NP) un telescopio]]]
```

6. Creación de un reemplazo para el parser del sistema Inform

Uno de los objetivos iniciales de esta practica era, como ya se ha comentado en la introducción, crear un reemplazo para el parser del sistema *Inform*, de manera que se pudiesen comprobar *ad hoc* las diferencias entre la utilizacion de las tecnicas *tradicionales* en los sistemas de ficcion interactiva y las soluciones de ingenieria vistas en clase.

Sin embargo se ha puesto de manifiesto durante la realizacion de la practica que el esfuerzo necesario para realizar esta tarea desbordaba con creces las intenciones iniciales, por las siguientes razones:

- El sistema *Inform*, aunque completo para el caso de la ficcion interactiva, es muy inadecuado (en comparacion con otros lenguajes de programacion actuales) para tareas genéricas de programación.

  Asi, este lenguaje no da soporte a temas tan básicos como la asignación dinámica de memoria, tratamiento de estructuras de datos[33], E/S de archivos binarios[34], punto flotante... Aproximadamente la mitad del tiempo empleado en la implementacion del analizador sintáctico fue para suplir estas deficiencias del lenguaje.

  Evidentemente, aunque el alumno conocia superficialmente el sistema *Inform*, nunca lo habia utilizado para ninguna aplicación (en otro caso las estimaciones iniciales en la propuesta de práctica no habrían sido tan optimistas).

- Aunque el lenguaje *Inform* ha estado por debajo de las expectativas iniciales, la libreria ha estado por encima, es decir, es mas compleja de lo que se esperaba.

- Además existe un problema sobre qué corpus utilizar para el análisis, ya que la comunicación en un mundo de ficción interactiva se realiza por lo general en forma de órdenes (normalmente imperativas), y la aparición de este tipo de formas gramaticales en los textos escritos (y por tanto en los corpus como SUSANNE) es escasa; por tanto necesitaríamos utilizar un corpus de gran tamaño[35], depurar uno existente, o incluso crearlo, para poder obtener una respuesta de calidad contra la que comparar la salida *tradicional*, y ninguna de estas tres tareas es trivial.

[33] Inform es un lenguaje orientado a objetos, pero con matices; se pueden definir clases e instancias de dichas clases, pero de manera estática (con la intencion de crear un mundo virtual). La creación dinámica no está permitida.

[34] Como se trata de un lenguaje orientado a texto, sólamente permite tratar con archivos en formato texto, por lo que para introducir los datos extraidos del corpus ha sido inevitable el incluirlos en el propio ejecutable

[35] Esta seria la solucion mas facil: utilizar directamente el corpus SUSANNE; pero para poder utilizar un corpus de gran tamaño son necesarias soluciones de estructuras de datos muy eficientes, algo que habria que programar desde cero en lenguaje *Inform*

7. Critica al sistema Inform. Soluciones propuestas.

Aunque mejorado en muchos aspectos, el sistema *Inform* no deja de ser una continuación de la metodología y tecnicas utilizadas por las grandes compañias de ficción interactiva en los años 80[36].

- *Utilización de un lexicón externo* En el enfoque tradicional, las palabras reconocidas por un sistema de ficcion interactiva son aquellas susceptibles de encontrar a la entrada (nombres de objetos, y un numero reducido de verbos y construcciones gramaticales). La idea subyacente es que si algo no existe, o no se puede hacer en el mundo virtual, no va a ser necesario referirse a ello en ningun momento.

  Esto es un claro sofisma, ya que el usuario no sabe exactamente qué se puede o no se puede hacer en el mundo virtual, y sólo tiene sentido si es mandatorio el ahorro de recursos de almacenamiento (como era el caso en los micros de los años 80). Estas restricciones históricas han desaparecido, y no tiene ningún sentido el seguir aplicando el concepto tradicional de diccionario, normalmente reducido y siempre dependiente del tamaño y características del mundo virtual.

  La utilizacion de diccionarios externos permitirá, en primer lugar, que se reduczcan a mínimos las ocasiones en las que haya que utilizar respuestas genéricas como `'I can't see such thing'` o similares, debidas al desconocimiento de cómo articular una frase con el nombre proporcionado a la entrada.

  Además la información de etiquetado del diccionario facilita en gran medida la programación de un mundo virtual, ya que evita la necesidad de tener que definir manualmente atributos como el genero[37], numero... de cada objeto, una tecnica altamente propensa a errores. Por ejemplo si en el objeto `vaca` nos olvidamos de definir el atributo `female`, el resultado de `print (a)vaca` sería `'el vaca'`. Un diccionario externo con información de etiquetado evitaria esta situación (ambigüedades puntuales aprte).

  La información de lema tambien sería tremendamente útil en la generación de respuestas, ya que (por ejemplo) teniendo un nombre en plural a la entrada podemos crear una respuesta con dicho nombre en singular (el rango de posibilidades es muy amplio, incluyendo cambios en tiempo verbal, genero...). En el enfoque tradicional una entrada como `'pick up the stones'` tendria forzosamente que responderse con frases com `'I can't see stones here'` o `'there are no stones'` ya que desconocemos el singular de la palabra, mientras que con un lexicon externo tendriamos un rango mucho mas amplio de respuestas posibles (`'Which stone?'`...).

- *Adivinacion (*guessing*) y matching con errores* Ante una entrada en la que existe una palabra que no pertenece al diccionario, la respuesta tradicional es mostrar un mensaje de error (evidentemente genérico ya que no conocemos como articular la palabra).

  Una aproximación mucho mas elegante seria utilizar matching con errores para ver si la palabra (con errores de insercion, borrado o sustitución) se refiere a alguno de

[36] Idem. para *TADS* y otros sistemas de ficcion interactiva

[37] Ver seccion 3.2.3.

los objetos al alcance del usuario en el mundo virtual[38]. Si es asi, se formularía una pregunta retórica al usuario para confirmar la suposición (`'Te refieres a...?'`) y en caso de respuesta afirmativa, se procedería a realizar la acción. El usuario no recibiría un mensaje de error (que revela falta de entendimiento por parte del parser) sino de corrección (que revela justo lo contrario).

En caso de respuesta negativa, o no haber un matching satisfactorio, seria interesante utilizar adivinación para evitar forzar una respuesta genérica.

- *Utilización de una gramática probabilística externa* En la seccion 3 comentabamos que una de las razones para escoger *Inform* como sistema de referencia era que, a diferencia de otros sistemas, permitia una facil modificación de sus librerías.

  Esta facilidad de manipular el comportamiento del parser, si lo analizamos un poco, es bastante relativa, ya que para poder modificar las librerías del sistema, es necesario primero comprender a fondo su funcionamiento (o arriesgarse a efectos colaterales y comportamientos imprevistos). Esto para un usuario sin conocimientos de programación, es casi imposible.

  La utilización de una gramática probabilistica externa, que utilizariamos como base para alguno de los procedimientos de análisis sintáctico vistos en clase (Earley, LR generalizado...), permitiría que este análisis se realizara de una manera mas transparente, comprensible por los usuarios sin conocimientos de programación, y eliminaría la necesidad de generar una gran cantidad de documentación relativa al funcionamiento del parser. Bastaria con remitirse a la bibliografía para el funcionamiento interno del analizador sintáctico y a la gramática externa para las reglas de análisis utilizadas.

  Además tendríamos ahora la posibilidad de, sin variar el resto del sistema, utilizar otras gramáticas que estén a nuestro alcance como referencia, para comparar resultados o probar nuevas direcciones. Por ejemplo si nuestro sistema esta funcionando con el corpus SUSANNE, de manera casi inmediato podríamos probar los corpus LUCY o CHRISTINE, que con una sintaxis casi idéntica modelan la forma de hablar del inglés coloquial o del empleado por los niños pequeños[39]

  Notese que además en el caso de análisis sintácticos ambiguos, podemos determinar cual es el esperado utilizando, además de la información probabilística, la información derivada del mundo virtual; es decir, si el análisis más probable no es correcto porque por ejemplo referencia objetos que no estan al alcance del usuario, podemos probar[40] –antes de proporcionar una respuesta de error– otras hipótesis menos probables y comprobar si se trata de acciones válidas, y en este caso, ejecutarlas.

- *Analisis semántico utilizando estructuras de caracteristicas (*feature structures*) y formalismos basados en unificación* Aunque la politica utilizada en *Inform* es la de no ser demasiado estrictos con la entrada, y aceptarla tal como viene mientras

[38] Es decir, realizamos el matching sobre un conjunto reducido de palabras.

[39] La comprensión de más de un registro lingüstico podria ser fundamental para el exito futuro de los trabajos de ficción interactiva (destinados a un público general); la utilización de varias gramáticas probabilísticas compitiendo entre ellas podría ser un factor a considerar.

[40] Por orden y posiblemente mientras la probabilidad se mantenga por encima de unos umbrales

seamos capaces de comprenderla, seria mucho mas elegante el detectar fallos en la entrada y plantear preguntas retóricas al usuario acerca de los mismos.

- *Redes semánticas (*WordNet*)*: La introduccion de redes semánticas en la ficción interactiva seria un gran avance en el campo. Actualmente uno de los mayores problemas que se encuentra un usuario inexperto que quiere realizar una determinada acción es que si no encuentra las palabras adecuadas puede no conseguir completar la acción con éxito, aún en el caso de que sea perfectamente posible (e incluso necesario) realizarla.

  La introducción de recursos de relación semántica dotaría de flexibilidad a la entrada de usuario. Simplificaría además en gran medida la creación de mundos virtuales, eliminando la necesidad de especificar listas de sinónimos y adjetivos para cada objeto del mundo virtual, una practica muy común en *Inform*.

8. Detalles de implementacion

El archivo `practica.tgz` que se entrega con esta memoria contiene la siguiente estructura de directorios:

`/pdf`	Texto de esta memoria y de la propuesta de practica presentada
`/inform`	Codigo fuente del compilador *Inform*, asi como los interpretes *Frotz* y *GlulXe*[41]
`/parte1`	Implementacion Java del programa descrito en la seccion 4
`/parte2`	Implementacion *Inform* del programa descrito en la seccion 5

El contenido de cada uno de estos directorios y la forma de compilar las implementaciones se detallan en las siguientes secciones.

8.1. Compilador Inform e interpretes

Para obtener el compilador y los interpretes, es necesario entrar en el directorio `inform` y ejecutar el script `compilar`, que se encarga de realizar las acciones necesarias. Se obtendran tres ejecutables `inform`, `glulxe`, `frotz` ubicados en el mismo directorio.

Es improbable que se produzcan errores de compilacion ya que ninguno de los tres programas tiene dependencias externas de otras librerias. Hemos utilizado las versiones más básicas de los interpretes para evitar posibles problemas de portabilidad[42].

No es necesario mover estos ejecutables a otros directorios; serán llamados por los makefiles de las implementaciones correspondientes.

En caso de problemas, o si se quiere obtener otra version de los interpretes (o del compilador), estan disponibles a traves de [10].

8.2. Implementacion Java

La implementacion Java consta de las siguientes clases (ubicadas en el directorio `clases`, excepto el programa principal):

[41] Mencionados en la seccion 3.1

[42] Aunque existen versiones de los interpretes para las librerias *curses* o *X*, hemos optado por la version para terminales tontos (*dumb terminals*).

`Main`	Programa principal; contiene todo el codigo dependiente del corpus de entrada elegido. Lee secuencialmente la entrada y pasa la informacion a los *singletons* Diccionario, Gramatica y Etiquetador.
`Diccionario`	Contiene el comportamiento necesario para mantener un diccionario y obtener una representación eficiente del mismo.
`Palabra`	Representa una entrada de un diccionario.
`Estado`	Representa un estado de un automata determinista finito numerado.
`Gramatica`	Mantiene una gramatica probabilistica y permite obtener una representacion eficiente de la misma (de manera análoga a `Diccionario`)
`Regla`	Representa una entrada de la gramática.
`Etiquetador`	Mantiene una tabla que relaciona etiquetas con indices de manera unívoca.
`GenericWriter`	Interfaz de escritura binaria.
`InformCodeZMachineWriter`	Clase para la escritura a disco con salida en formato .h de *Inform*, utilizando los tipos de dato de la maquina virtual *Z*.
`InformCodeGlulxWriter`	Idem, pero para la maquina virtual *GlulX*.

El programa se compila con `make`, y supone que el compilador `javac` de Sun esta presente en el sistema. Tambien se puede compilar utilizando el compilador `gcc`[43] mediante `make gcc`, pero en este caso es necesario eliminar la referencia a la funcion `replaceAll` presente en el programa principal[44].

Otras opciones que soporta el mismo *makefile* son las siguientes:

`make z`	Procesar el archivo de entrada `corpus/corpus` con salida para la maquina virtual *Z* y generar un unico archivo `corpus/corpus.h` con los distintos archivos de salida obtenidos.
`make u`	Idem. pero para la maquina virtual *GlulX*
`make clean`	Borrar el codigo objeto generado en el directorio actual y sucesivos.

Para facilitar el uso, se recomienda por tanto copiar el corpus de entrada al archivo `corpus/corpus` y utilizar alguna de las opciones del *makefile*. Los corpus de prueba utilizados en secciones previos de esta memoria se encuentran en el directorio `corpus/prueba`, y los ficheros correspondientes a textos de ficcion del corpus SUSANNE se encuentran en `corpus/susanne`. En caso de querer introducir mas de un fichero al programa, se recomienda utilizar `cat` y redirigir la salida `corpus/corpus`.

En caso otro caso, la sintaxis del programa es la siguiente:

[43] *GNU compiler collection*

[44] El soporte Java en gcc es reciente y las librerias todavia no son completamente operativas.

```
Sintaxis:

   main [comandos] [archivo1] [archivo2] [...]

Comandos:

   l: convertir a minusculas las palabras del diccionario
   s: leer archivo de treebank Susanne
   z: salida como fichero .h de Inform (maquina virtual Z)
   u: salida como fichero .h de Inform (maquina virtual GlulX)

Ejemplo:

   "main lssz n01 n02 salida" lee los archivos n01 y n02 del
   treebank Susanne y escribe la salida en formato texto
   para el sistema Inform, en los archivos salida.*; las palabras
   del diccionario seran convertidas a minusculas
```

Notese que si queremos obtener salida para la maquina virtual *Z*, debemos tener en cuenta que se trata de una maquina virtual de 16 bits y que por tanto un corpus ya de tamaño mediano puede desbordar su capacidad. Adicionalmente es necesario convertir la entrada a minusculas ya que dicha maquina virtual no distingue inicialmente mayusculas de minusculas al leer de la entrada estándar.

8.3. Implementacion Inform

La implementacion *Inform* consta de los siguientes archivos:

`main.inf`	Programa principal
`include/etiquetas.h`	Contiene varias funciones basicas para manejar etiquetas
`include/puntoFlotante.h`	Contiene las funciones para emular punto flotante utilizando coma fija (0.16 o 0.32 dependiendo de la maquina virtual destino)
`include/tokens.h`	Contiene las funciones para leer la entrada de teclado, dividirla en tokens, y manejar dichos tokens.
`include/analisisLexico.h`	Contiene las funciones `indice_a_cadena`, `cadena_a_indice` y funciones para obtener e imprimir las etiquetas, probabilidades... de las palabras del lexicon
`include/analisisSintactico.h`	Contiene el algoritmo (`Earley`) que analiza sintacticamente la entrada, y las funciones necesarias para obtener y manejar la tabla de análisis resultante

Adicionalmente, es necesario copiar al directorio `include` un fichero `corpus.h` con toda la informacion obtenida como salida de la implementacion Java del apartado anterior[45]. Si se quiere se puede especificar otro nombre para este fichero simplemente cambiando la referencia en `main.inf`.

El programa se compila con `make`, aceptando los siguientes parametros:

`make c`	Copiar el archivo `corpus.h` del directorio `../parte1/corpus` al directorio `include`
`make z`	Compilar para la maquina virtual *Z* y ejecutar
`make u`	Compilar para la maquina virtual *GlulX* y ejecutar
`make clean`	Borrar el codigo objeto generado en el directorio actual y sucesivos.

En el caso de la maquina virtual `Z` al ejecutar el programa se pide una cadena de entrada y se muestran los resultados del análisis sintáctico para dicha cadena.

Para la maquina virtual `GlulX`, ademas de todos los problemas mencionados en la seccion 6, el alumno ha sido ¡incapaz de encontrar la forma de leer una cadena de entrada![46] por lo que la unica manera de introducir una entrada para analizar es modificando el propio codigo del programa principal, introduciendo a mano la cadena de entrada.

Por lo tanto, para ejecutar pruebas se recomienda utilizar la maquina *Z*[47] y sólo compilar para *GlulX* si queremos probar resultados con corpus de tamaño medio o grande.

[45] Debemos tener en cuenta que el corpus debe haber sido preparado para la misma maquina virtual para la que vayamos a compilar

[46] *Inform* es un compilador muy orientado a utilizarse con una libreria concreta; la introduccion de la maquina virtual GlulX es reciente y no esta bien documentada. Es facil utilizarla utilizando el interfaz de la libreria estandar, pero en otro caso para realizar cualquier 'tonteria' (como leer la entrada estandar) es necesario conocer el ensamblador de la maquina.

[47] Cuyo interprete es ademas bastante mas eficiente que *GlulXe*

Referencias

[1] Miguel A. Alonso Pardo, Jorge Graña Gil: *Material preparado para la asignatura de Lenguajes Naturales, curso 2003/04*
http://www.dc.fi.udc.es/ alonso

[2] Graham Nelson: *Imform Designer's Manual, 4th Edition*
Interactive Fiction Library (Agosto 2001). 576 paginas.

[3] Dennis G. Jerz: *Interactive Fiction annotated bibliography*
http://jerz.setonhill.edu/if/bibliography/index.html

[4] Geoffrey Sampson: *The SUSANNE Analytic Scheme*
http://www.grsampson.net/RSUE.html

[5] Carnegie Mellon University of Computer Science: *OZ project Homepage*
http://www.cs.cmu.edu/afs/cs.cmu.edu/project/oz/web/index.html

[6] Nick Montfort: *Twisty Little Passages: An approach to interactive fiction*
MIT Press (Diciembre 2003). 328 paginas.

[7] Jean Mark Gawron: *Statistical methods in Computational Linguistics*
http://www-rohan.sdsu.edu/ gawron/stat

[8] Jan Daciuk, Stoyan Mihov, Bruce Watson, Richard Watson: *Incremental construction of minimal Acyclic Finite State Automata*
Journal of Computational Linguistics (April 2000)

[9] Doug Arnold: *Computational Linguistics I*
http://courses.essex.ac.uk/LG511

[10] *The Interactive Fiction Archive*
ftp://ftp.gmd.de/if-archive

[11] Hector Briceno, Wesley Chao, Andrew Glenn, Stanley Hu, Ashwin Krishnamurthy, Bruce Tsuchida: *Down from the top of its game: The story of Infocom Inc.*
http://web.mit.edu/6.933/www/Fall2000/infocom

[12] Jan Earley: *An efficient context-free parsing algorithm*
Communications of the Association for Computing Machinery (February 1970)

9. Licencia CreativeCommons

Reconocimiento-NoComercial-SinObraDerivada 2.5 España

CREATIVE COMMONS CORPORATION NO ES UN DESPACHO DE ABOGADOS Y NO PROPORCIONA SERVICIOS JURÍDICOS. LA DISTRIBUCIÓN DE ESTA LICENCIA NO CREA UNA RELACIÓN ABOGADO-CLIENTE. CREATIVE COMMONS PROPORCIONA ESTA INFORMACIÓN TAL CUAL (ON AN "AS-IS" BASIS). CREATIVE COMMONS NO OFRECE GARANTÍA ALGUNA RESPECTO DE LA INFORMACIÓN PROPORCIONADA, NI ASUME RESPONSABILIDAD ALGUNA POR DAÑOS PRODUCIDOS A CONSECUENCIA DE SU USO.

Licencia

LA OBRA (SEGÚN SE DEFINE MÁS ADELANTE) SE PROPORCIONA BAJO LOS TÉRMINOS DE ESTA LICENCIA PÚBLICA DE CREATIVE COMMONS ("CCPL" O "LICENCIA"). LA OBRA SE ENCUENTRA PROTEGIDA POR LA LEY ESPAÑOLA DE PROPIEDAD INTELECTUAL Y/O CUALESQUIERA OTRAS NORMAS RESULTEN DE APLICACIÓN. QUEDA PROHIBIDO CUALQUIER USO DE LA OBRA DIFERENTE A LO AUTORIZADO BAJO ESTA LICENCIA O LO DISPUESTO EN LAS LEYES DE PROPIEDAD INTELECTUAL.

MEDIANTE EL EJERCICIO DE CUALQUIER DERECHO SOBRE LA OBRA, USTED ACEPTA Y CONSIENTE LAS LIMITACIONES Y OBLIGACIONES DE ESTA LICENCIA. EL LICENCIADOR LE CEDE LOS DERECHOS CONTENIDOS EN ESTA LICENCIA, SIEMPRE QUE USTED ACEPTE LOS PRESENTES TÉRMINOS Y CONDICIONES.

1. **Definiciones**

   a) La "obra" es la creación literaria, artística o científica ofrecida bajo los términos de esta licencia.

   b) El "autor" es la persona o la entidad que creó la obra.

   c) Se considerará "obra conjunta" aquella susceptible de ser incluida en alguna de las siguientes categorías:

      1) "Obra en colaboración", entendiendo por tal aquella que sea resultado unitario de la colaboración de varios autores.

2) "Obra colectiva", entendiendo por tal la creada por la iniciativa y bajo la coordinación de una persona natural o jurídica que la edite y divulgue bajo su nombre y que esté constituida por la reunión de aportaciones de diferentes autores cuya contribución personal se funde en una creación única y autónoma, para la cual haya sido concebida sin que sea posible atribuir separadamente a cualquiera de ellos un derecho sobre el conjunto de la obra realizada.

3) "Obra compuesta e independiente", entendiendo por tal la obra nueva que incorpore una obra preexistente sin la colaboración del autor de esta última.

d) Se considerarán "obras derivadas" aquellas que se encuentren basadas en una obra o en una obra y otras preexistentes, tales como: las traducciones y adaptaciones; las revisiones, actualizaciones y anotaciones; los compendios, resúmenes y extractos; los arreglos musicales y, en general, cualesquiera transformaciones de una obra literaria, artística o científica, salvo que la obra resultante tenga el carácter de obra conjunta en cuyo caso no será considerada como una obra derivada a los efectos de esta licencia. Para evitar la duda, si la obra consiste en una composición musical o grabación de sonidos, la sincronización temporal de la obra con una imagen en movimiento ("synching") será considerada como una obra derivada a los efectos de esta licencia.

e) Tendrán la consideración de "obras audiovisuales" las creaciones expresadas mediante una serie de imágenes asociadas, con o sin sonorización incorporada, así como las composiciones musicales, que estén destinadas esencialmente a ser mostradas a través de aparatos de proyección o por cualquier otro medio de comunicación pública de la imagen y del sonido, con independencia de la naturaleza de los soportes materiales de dichas obras.

f) El "licenciador" es la persona o la entidad que ofrece la obra bajo los términos de esta licencia y le cede los derechos de explotación de la misma conforme a lo dispuesto en ella.

g) "Usted" es la persona o la entidad que ejercita los derechos cedidos mediante esta licencia y que no ha violado previamente los términos de la misma con respecto a la obra, o que ha recibido el permiso expreso del licenciador de ejercitar los derechos cedidos mediante esta licencia a pesar de una violación anterior.

h) La "transformación" de una obra comprende su traducción, adaptación y cualquier otra modificación en su forma de la que se derive una obra diferente. Cuando se trate de una base de datos según se define más adelante, se considerará también transformación la reordenación de la misma. La creación resultante de la transformación de una obra tendrá la consideración de obra derivada.

i) Se entiende por "reproducción" la fijación de la obra en un medio que permita su comunicación y la obtención de copias de toda o parte de ella.

j) Se entiende por "distribución" la puesta a disposición del público del original o copias de la obra mediante su venta, alquiler, préstamo o de cualquier otra

forma.

k) Se entenderá por "comunicación pública" todo acto por el cual una pluralidad de personas pueda tener acceso a la obra sin previa distribución de ejemplares a cada una de ellas. No se considerará pública la comunicación cuando se celebre dentro de un ámbito estrictamente doméstico que no esté integrado o conectado a una red de difusión de cualquier tipo. A efectos de esta licencia se considerará comunicación pública la puesta a disposición del público de la obra por procedimientos alámbricos o inalámbricos, incluida la puesta a disposición del público de la obra de tal forma que cualquier persona pueda acceder a ella desde el lugar y en el momento que elija.

l) La "explotación" de la obra comprende su reproducción, distribución, comunicación pública y transformación.

m) Tendrán la consideración de "bases de datos" las colecciones de obras ajenas, de datos o de otros elementos independientes como las antologías y las bases de datos propiamente dichas que por la selección o disposición de sus contenidos constituyan creaciones intelectuales, sin perjuicio, en su caso, de los derechos que pudieran subsistir sobre dichos contenidos.

n) Los "elementos de la licencia" son las características principales de la licencia según la selección efectuada por el licenciador e indicadas en el título de esta licencia: Reconocimiento de autoría (Reconocimiento), Sin uso comercial (NoComercial), Sin obras derivadas (SinObraDerivada).

2. **Límites y uso legítimo de los derechos**. Nada en esta licencia pretende reducir o restringir cualesquiera límites legales de los derechos exclusivos del titular de los derechos de propiedad intelectual de acuerdo con la Ley de Propiedad Intelectual o cualesquiera otras leyes aplicables, ya sean derivados de usos legítimos, tales como el derecho de copia privada o el derecho a cita, u otras limitaciones como la derivada de la primera venta de ejemplares.

3. **Concesión de licencia**. Conforme a los términos y a las condiciones de esta licencia, el licenciador concede (durante toda la vigencia de los derechos de propiedad intelectual) una licencia de ámbito mundial, sin derecho de remuneración, no exclusiva e indefinida que incluye la cesión de los siguientes derechos:

   a) Derecho de reproducción, distribución y comunicación pública sobre la obra.

   b) Derecho a incorporarla en una o más obras conjuntas o bases de datos y para su reproducción en tanto que incorporada a dichas obras conjuntas o bases de datos.

   c) Derecho de distribución y comunicación pública de copias o grabaciones de la obra, como incorporada a obras conjuntas o bases de datos.

   Los anteriores derechos se pueden ejercitar en todos los medios y formatos, tangibles o intangibles, conocidos o por conocer. Los derechos mencionados incluyen el derecho a efectuar las modificaciones que sean precisas técnicamente para el ejercicio de los derechos en otros medios y formatos. Todos los derechos no cedidos

expresamente por el licenciador quedan reservados, incluyendo, a título enunciativo pero no limitativo, los derechos establecidos en la sección 4d.

4. **Restricciones**. La cesión de derechos que supone esta licencia se encuentra sujeta y limitada a las restricciones siguientes:

   a) Usted puede reproducir, distribuir o comunicar públicamente la obra solamente bajo los términos de esta licencia y debe incluir una copia de la misma, o su Identificador Uniforme de Recurso (URI), con cada copia o grabación de la obra que usted reproduzca, distribuya o comunique públicamente. Usted no puede ofrecer o imponer ningún término sobre la obra que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los cesionarios de la misma. Usted no puede sublicenciar la obra. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra con medidas tecnológicas que controlen el acceso o uso de la obra de una manera contraria a los términos de esta licencia. Lo anterior se aplica a una obra en tanto que incorporada a una obra conjunta o base de datos, pero no implica que éstas, al margen de la obra objeto de esta licencia, tengan que estar sujetas a los términos de la misma. Si usted crea una obra conjunta o base de datos, previa comunicación del licenciador, usted deberá quitar de la obra conjunta o base de datos cualquier crédito requerido en el apartado 4c, según lo que se le requiera y en la medida de lo posible.

   b) Usted no puede ejercitar ninguno de los derechos cedidos en la sección 3 anterior de manera que pretenda principalmente o se encuentre dirigida hacia la obtención de un beneficio mercantil o la remuneración monetaria privada. El intercambio de la obra por otras obras protegidas por la propiedad intelectual mediante sistemas de compartir archivos no se considerará como una manera que pretenda principalmente o se encuentre dirigida hacia la obtención de un beneficio mercantil o la remuneración monetaria privada, siempre que no haya ningún pago de cualquier remuneración monetaria en relación con el intercambio de las obras protegidas.

   c) Si usted reproduce, distribuye o comunica públicamente la obra o cualquier obra conjunta o base datos que la incorpore, usted debe mantener intactos todos los avisos sobre la propiedad intelectual de la obra y reconocer al autor original, de manera razonable conforme al medio o a los medios que usted esté utilizando, indicando el nombre (o el seudónimo, en su caso) del autor original si es facilitado, y/o reconocer a aquellas partes (por ejemplo: institución, publicación, revista) que el autor original y/o el licenciador designen para ser reconocidos en el aviso legal, las condiciones de uso, o de cualquier otra manera razonable; el título de la obra si es facilitado; de manera razonable, el Identificador Uniforme de Recurso (URI), si existe, que el licenciador especifica para ser vinculado a la obra, a menos que tal URI no se refiera al aviso sobre propiedad intelectual o a la información sobre la licencia de la obra. Tal aviso se puede desarrollar de cualquier manera razonable; con tal de que, sin embargo, en el caso de una obra conjunta o base datos, aparezca como mínimo

este aviso allá donde aparezcan los avisos correspondientes a otros autores y de forma comparable a los mismos.

d) Para evitar la duda, sin perjuicio de la preceptiva autorización del licenciador, y especialmente cuando la obra se trate de una obra audiovisual, el licenciador se reserva el derecho exclusivo a percibir, tanto individualmente como mediante una entidad de gestión de derechos, o varias, (por ejemplo: SGAE, Dama, VEGAP), los derechos de explotación de la obra, así como los derivados de obras conjuntas o bases de datos, si dicha explotación pretende principalmente o se encuentra dirigida hacia la obtención de un beneficio mercantil o la remuneración monetaria privada.

e) En el caso de la inclusión de la obra en alguna base de datos o recopilación, el propietario o el gestor de la base de datos deberá renunciar a cualquier derecho relacionado con esta inclusión y concerniente a los usos de la obra una vez extraída de las bases de datos, ya sea de manera individual o conjuntamente con otros materiales.

5. **Exoneración de responsabilidad**

A MENOS QUE SE ACUERDE MUTUAMENTE ENTRE LAS PARTES, EL LICENCIADOR OFRECE LA OBRA TAL CUAL (ON AN "AS-IS" BASIS) Y NO CONFIERE NINGUNA GARANTÍA DE CUALQUIER TIPO RESPECTO DE LA OBRA O DE LA PRESENCIA O AUSENCIA DE ERRORES QUE PUEDAN O NO SER DESCUBIERTOS. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE TALES GARANTÍAS, POR LO QUE TAL EXCLUSIÓN PUEDE NO SER DE APLICACIÓN A USTED.

6. **Limitación de responsabilidad**

SALVO QUE LO DISPONGA EXPRESA E IMPERATIVAMENTE LA LEY APLICABLE, EN NINGÚN CASO EL LICENCIADOR SERÁ RESPONSABLE ANTE USTED POR CUALQUIER TEORÍA LEGAL DE CUALESQUIERA DAÑOS RESULTANTES, GENERALES O ESPECIALES (INCLUIDO EL DAÑO EMERGENTE Y EL LUCRO CESANTE), FORTUITOS O CAUSALES, DIRECTOS O INDIRECTOS, PRODUCIDOS EN CONEXIÓN CON ESTA LICENCIA O EL USO DE LA OBRA, INCLUSO SI EL LICENCIADOR HUBIERA SIDO INFORMADO DE LA POSIBILIDAD DE TALES DAÑOS.

7. **Finalización de la licencia**

a) Esta licencia y la cesión de los derechos que contiene terminarán automáticamente en caso de cualquier incumplimiento de los términos de la misma. Las personas o entidades que hayan recibido obras conjuntas o bases de datos de usted bajo esta licencia, sin embargo, no verán sus licencias finalizadas, siempre que tales personas o entidades se mantengan en el cumplimiento íntegro de esta licencia. Las secciones 1, 2, 5, 6, 7 y 8 permanecerán vigentes pese a cualquier finalización de esta licencia.

b) Conforme a las condiciones y términos anteriores, la cesión de derechos de esta licencia es perpetua (durante toda la vigencia de los derechos de propiedad intelectual aplicables a la obra). A pesar de lo anterior, el licenciador se reserva el derecho a divulgar o publicar la obra en condiciones distintas a las presentes, o de retirar la obra en cualquier momento. No obstante, ello no supondrá dar por concluida esta licencia (o cualquier otra licencia que haya sido concedida, o sea necesario ser concedida, bajo los términos de esta licencia), que continuará vigente y con efectos completos a no ser que haya finalizado conforme a lo establecido anteriormente.

8. **Miscelánea**

a) Cada vez que usted explote de alguna forma la obra, o una obra conjunta o una base datos que la incorpore, el licenciador original ofrece a los terceros y sucesivos licenciatarios la cesión de derechos sobre la obra en las mismas condiciones y términos que la licencia concedida a usted.

b) Si alguna disposición de esta licencia resulta inválida o inaplicable según la Ley vigente, ello no afectará la validez o aplicabilidad del resto de los términos de esta licencia y, sin ninguna acción adicional por cualquiera las partes de este acuerdo, tal disposición se entenderá reformada en lo estrictamente necesario para hacer que tal disposición sea válida y ejecutiva.

c) No se entenderá que existe renuncia respecto de algún término o disposición de esta licencia, ni que se consiente violación alguna de la misma, a menos que tal renuncia o consentimiento figure por escrito y lleve la firma de la parte que renuncie o consienta.

d) Esta licencia constituye el acuerdo pleno entre las partes con respecto a la obra objeto de la licencia. No caben interpretaciones, acuerdos o términos con respecto a la obra que no se encuentren expresamente especificados en la presente licencia. El licenciador no estará obligado por ninguna disposición complementaria que pueda aparecer en cualquier comunicación de usted. Esta licencia no se puede modificar sin el mutuo acuerdo por escrito entre el licenciador y usted.

Creative Commons no es parte de esta licencia, y no ofrece ninguna garantía en relación con la obra. Creative Commons no será responsable frente a usted o a cualquier parte, por cualquier teoría legal de cualesquiera daños resultantes, incluyendo, pero no limitado, daños generales o especiales (incluido el daño emergente y el lucro cesante), fortuitos o causales, en conexión con esta licencia. A pesar de las dos (2) oraciones anteriores, si Creative Commons se ha identificado expresamente como el licenciador, tendrá todos los derechos y obligaciones del licenciador.