Conocimiento y Razonamiento Automatizado

Analizador sintáctico de oraciones

Grado en Ingeniería Informática
Universidad de Alcalá

Pablo García García
Álvaro Jesús Martínez Parra
Alejandro Raboso Vindel

27 de abril de 2023

Índice general

Capítulo 1

Introducción y objetivos

1.1. Análisis de oraciones

1.2. Simplificación de oraciones

Capítulo 2

Análisis sintáctico de oraciones

2.1. Diccionario

2.2. Gramática

Capítulo 3

Mejoras

3.1. Traducción a inglés

3.2. Conexión entre Prolog y Python

3.3. Árbol sintáctico en LaTeX

Como mejora, se ha optado por generar una mejor visualización del árbol de sintaxis que la ofrecida por el archivo `draw.pl`. Para ello, hemos elegido LaTeX como lenguaje que nos permitiera realizar el dibujo del árbol en tiempo de ejecución dada la sintaxis de Prolog. Las salidas de esta mejora son los diferentes árboles que se han ido mostrando a lo largo de este documento.

Para llevar esto a cabo, nos hemos aprovechado de lo explicado previamente en la Sección 3.2, añadiendo un archivo llamado `Prolog2LaTeX.py` que contiene diversas funciones, siendo `main` la única que necesitamos llamar para generar nuestro dibujo. Como en el programa `analizador.py` podíamos recoger los resultados de nuestras consultas en Prolog en forma de diccionario, pasaremos esta respuesta a la función `main` previamente nombrada y esta hará todo el trabajo.

Expliquemos ahora el funcionamiento general de `Prolog2LaTeX.py`. En este fichero tenemos tres funciones importantes, que se ejecutan en el siguiente orden en la función `main`: `parser`, `compile`, y `show`.

- `parser`: esta función recibe la respuesta de Prolog en el siguiente formato **PONER EJEMPLO**, y la convierte a la sintaxis que utiliza un objeto `\Tree` del paquete `qtree` de LaTeX. Como la sintaxis es la siguiente `[.P1 [.HIJO1_P1 [.HIJO1_HIJO1_P1] [.HIJO2_HIJO1_P1] ] [.HIJO2_P1] [.HIJO3_P1] ]`, no tenemos más que sustituir con cuidado paréntesis y comas, por corchetes y espacios. Pondremos este objeto dentro de un documento de clase `standalone` para que aparezca sólo el árbol. Además, para poder ver de manera resaltada las palabras clave como `gn` o `gadj` entre otras, se añade un diccionario de forma que se pueda personalizar el contenido que aparecerá

en su lugar en el árbol. Una vez tenemos el archivo de LaTeX listo, lo escribimos en un fichero y pasamos a la siguiente función.

- `compile`: como ya tenemos nuestro documento listo, debemos compilarlo para poder visualizarlo. Para ello, invocamos al compilador pdfLaTeX con el archivo que `parser` ha escrito (podría haberse optado por cualquier otro compilador), y este nos devolverá un archivo `.pdf` con el árbol listo para visualizar. Borra también archivos temporales.

- `show`: al usuario se le ofrece la opción de recibir el árbol como documento `pdf` o como imagen. Para ello, previamente, ha introducido como argumento `-pdf` o `-img`. Esta función se encarga de, en caso de que se quiera imagen llamar a la función `toJPG` que convierte el `pdf` a imagen `jpg`, y muestra el archivo final mediante el visor de `pdf` o imágenes que el usuario haya definido en su sistema operativo como predeterminado.

Apéndice A

Información relevante

A.1. Requisitos y dependencias

Para la realización de la práctica se han usado una serie de librerías y software externo, por lo que detallamos aquí qué es necesario para poder ejecutarla de manera correcta, y qué versiones se recomiendan.

- **Python**: se ha realizado con la versión `3.8.6`, sin embargo no se descarta que no funcione en versiones superiores. Previamente se deberán instalar una serie de dependencias mediante `pip`:
  - `pip install pyswipl`
  - `pip install unidecode`
  - `pip install shutil`
  - `pip install datetime`
  - `pip install pdf2image`
- **Prolog**: deberá utilizarse **obligatoriamente** la versión `8.4.2` de SWI-Prolog, pues con versiones superiores hemos encontrado incompatibilidades con `pyswipl`.
- **LaTeX**: deberemos tener instalada una distribución de LaTeX en nuestro equipo, en nuestro caso al estar trabajando en Windows hemos optado por MiKTeX (en nuestro caso versión `21.2`), pero en general nos puede servir cualquiera que contenga el compilador pdfLaTeX y que lo tenga añadido como variable de entorno. En caso de no ser esto posible o querer usar otro compilador, no habría ningún problema, pero debería modificarse la línea donde se invoca a este en `compile`. Además, durante la primera ejecución deberemos permitir la instalación de los paquetes necesarios como `qtree`.

A.2. Detalles

Se han cubierto todos los objetivos solicitados, funcionando correctamente las oraciones, sin ningún error de implementación.

Respecto al reparto de tareas, la práctica se ha realizado por todos los integrantes del grupo a la vez en una llamada de Discord, aportando cada uno soluciones y alternativas a los problemas que se iban encontrando.

Para finalizar, las referencias consultadas han sido, los apuntes de la asignatura, el manual de SWI Prolog, y StackOverFlow.