Proyecto #2 - Relaciones de Etimología

Inteligencia Artificial

Grupo 1, I Semestre 2018

El objetivo primario del proyecto es exponer a los estudiantes al procesamiento de datos mediante un motor de derivación lógico, en el espíritu de la última área conceptual abordada en el curso.

Para el desarrollo se utilizarán dos elementos primarios: un lenguaje de lógica sobre Python y una base de datos de relaciones etimológicas. Con ellos, los estudiantes desarrollaran una aplicación básica para responder preguntas sobre relaciones entre diferentes palabras contenidas en la base de datos.

Fuente de datos etimológicos

La etimología estudia el origen de las palabras. Debido a que muchos idiomas actuales tienen interrelaciones basadas en raíces comunes, resulta interesante analizarlas de una manera estructurada.

La "Etymological Wordnet" es una base de datos basada en en.wiktionary.org que recopila relaciones entre palabras en múltiples idiomas (aunque iniciando desde inglés). Tomando estos datos, se puede construir un sistema que, basado en proposiciones lógicas, nos permita responder preguntas relacionadas con orígen común de palabras, similitud entre lenguajes, etc.

En http://www1.icsi.berkeley.edu/~demelo/etymwn es posible descargar las base de datos y su descripción. Como se puede observar, existen diferentes tipos de relaciones que serán relevantes más adelante en la descripción del proyecto.

Biblioteca de procesamiento basado en lógica

Las relaciones de ancestralidad son un problema que es muy natural de abordar a través de proposiciones lógicas. Para evitar implementar un motor de lógica completo y permitir a los estudiantes enfocarse en el entendimiento y ejecución de un proyecto de este tipo se utilizará pyDatalog (https://sites.google.com/site/pydatalog).

Ésta es una biblioteca que define un lenguaje de definición de términos y operaciones lógicas sobre Python que permitirá a los estudiantes plantear la solución utilizando proposiciones. Está disponible en el Python Package Index (i.e. instalable con pip) y posee documentación a través

de tutoriales que relacionan las construcciones lógicas con implementaciones estándar en Python.

Se espera que los estudiantes resuelvan TODOS los problemas algorítmicos planteados con lógica, a través de la biblioteca. En caso contrario el profesor se reserva el derecho a no asignar puntos a las diferentes funcionalidades no implementadas de esta forma (ya que el objetivo del proyecto es éste).

Requerimientos detallados

Deberá existir un programa principal que desplegará una interfaz gráfica que le permitirá al usuario especificar datos de entrada (e.g. dos palabras) y ejecutar consultas asociadas a diferentes botones de la aplicación. Se pide a los estudiantes utilizar Tkinter, que es una biblioteca de Python para creación de ventanas y componentes gráficos independiente de plataforma.

La aplicación deberá permitir realizar las siguientes operaciones entre dos palabras:

- Determinar si dos palabras son heman@s
- Determinar si dos palabras son prim@s
- Determinar si una palabra es hij@ de otra
- Determinar si una palabra es ti@
- Determinar si son prim@s y en qué grado. A diferencia de las otras, conceptualmente esta no es sólo una salida de sí o no.

Además deberán poder realizarse las siguientes operaciones entre una palabra e idioma(s):

- Determinar si una palabra está relacionada con un idioma (Si / No)
- Obtener el conjunto de todas las palabras en un idioma originadas por una palabra específica (e.g. una palabra específica en latín puede originar múltiples palabras en español)
- Listar los idiomas relacionados con una palabra

Por último, los estudiantes deberán poder comparar idiomas a través de las siguientes operaciones:

- Contar todas las palabras comunes entre dos idiomas
- Listar todas las palabras comunes entre dos idiomas
- Idioma que más aportó a otro (e.g. latín a español). Medir basado en porcentaje
- LIstar todos los idiomas que aportaron a otro. Similar al anterior pero debe incluir porcentaje para todos los idiomas

Para cada operación realizada, se deberá mostrar las relaciones detalladas en la base de datos que se utilizaron para realizar la inferencia. Esto es clave para revisar la correctitud de las mismas y, por ende, la asignación del puntaje a cada rubro.

En la interfaz deberá ser posible seleccionar cuáles relaciones se considerarán al ejecutar cada operación (sin necesidad de reiniciarla). Por ejemplo, deberá ser posible seleccionar sólo relaciones de etimología directa ("rel:etymology") por un lado, de forma derivada ("rel:has_derived_for") por otro, o bien ambas en conjunto. Es natural esperar que habrá más relaciones entre palabras al permitir más tipos en las operaciones.

Por último, como se discutió en clase, el resolver problemas con lógica puede implicar retos de rendimiento. Dado que los estudiantes están al final de sus estudios y deben poder resolver problemas creativamente se espera que, de ser necesario, los estudiantes puedan aplicar técnicas de diseño y resolución de problemas para resolver los problemas planteados. Todas estas decisiones tomadas deberán documentarse en el reporte final.

Pruebas unitarias

Se espera que los estudiantes agreguen pruebas unitarias para cada uno de los requerimientos esbozados previamente. La expectativa es que cualquier combinación que se pueda ejecutar como prueba en la revisión tenga una correspondencia en pruebas.

Nótese que esto no quiere decir que haya que cargar la base de datos completa para realizar pruebas unitarias. Un buen diseño de pruebas permite crear una representación en memoria de datos de prueba que después permitan ejecutar las pruebas.

Se requiere que se pueda ejecutar "pytest" desde la raíz del código y ejecute todas las pruebas sin configuración adicional.

Informe

Los estudiantes deberán realizar un informe similar a los proyectos anteriores que deberá contener:

- Descripción de instalación. Por ejemplo, se pide que no envíen el archivo con la base de datos, pero se debe describir dónde se debe colocar a la hora de la revisión.
- Un manual de usuario, con capturas de pantalla, para cada operación
- Al menos un resultado interesante, descrito en prosa, para cada una de las operaciones planteadas.
- Detalles de implementación y diseño necesarios para hacer la ejecución de las operaciones eficiente
- Descripción de distribución de trabajo entre miembros del grupo y distribución de nota (a discutir en clase)

El informe deberá realizarse utilizando markdown en un repositorio público de github.com, perteneciente a los estudiantes. Deberá enviarse una copia en PDF al profesor a través de TEC Digital.

Detalles de Entrega y Revisión

El proyecto deberá realizarse en los mismos grupos creados para el Proyecto #1, o bajo autorización previa del profesor. Se debe enviar todos los entregables a través de TEC Digital a más tardar el 14 de junio del año en curso, antes de las 11:59PM.

El profesor se reserva el derecho a asignar una nota de cero si los requerimientos no son cumplidos de forma tal que impida ejecutar las funciones principales.

A manera de resumen los entregables serán:

- Código fuente con prefijo modular de Python tec.ic.ia.p2.g01. Este deberá estar almacenado en un repositorio privado de github. Se deberá enviar también un archivo comprimido a través de TEC digital.
- Pruebas unitarias
- Informe, en formato PDF (a TEC Digital) y Markdown en github.com
- Nótese que NO debe enviarse el archivo con la base de datos.

Consideraciones Python

- Se utilizará Python 3 (no 2.7)
- Se utilizará pip para instalar el módulo enviado. Se debe respetar la estructura de directorios apropiada para que la instalación sea exitosa.
- Se debe utilizar pytest para pruebas unitarias
- Seguir recomendaciones en http://docs.python-guide.org/en/latest/
- Seguir PEP 20. Una guía rápida con ejemplos está disponible en: http://artifex.org/~hblanks/talks/2011/pep20_by_example.pdf
- Utilizar PEP 8 como guía de estilo: http://pep8.org/