Text Minning

GENERACIÓN EXTRACTIVA DE TEXTOS

Ivan Martin Sanz
URJC | C/TULIPAN S/N

Contenido

Introducción

Objetivo

El objetivo es poner en práctica diferentes conceptos aprendidos durante el curso relacionados con la Minería de Texto, utilizando los conocimientos aprendidos en redes semánticas y resúmenes de texto automáticos.

Para ello se propone realizar un resumen por extracción del libro "*La comunidad del anillo*" de la trilogía "*El señor de los anillos*" de J.R.R. Tolkien.

Para la realización de este objetivo, son necesarios los siguientes ítems:

1. Se debe generar un código completamente funcional capaz de realizar resúmenes de texto dado cualquier contenido.
2. Debe de haber un método que acepte por parámetro el contenido y el factor de compresión deseado para el resumen.
3. Se debe mostrar el resumen al menos por pantalla, aunque si se desea se puede generar un PDF o cualquier otro tipo de documento que lo pueda contener.

Recorremos los anteriores hitos en orden de aparación, exponiendo las soluciones planteadas a las diferentes problemáticas encontradas en el desorrollo de esta prática.

Librerias

Para la realización de la práctica se ha hecho uso del lenguaje de programación ***Python*** en su versión 3.7, y de diversas librerías habituales en el desarrollo de proyectos NLP y de análisis de datos en general.

Estas librerías son:

- ***Pandas*** es una librería de Python especializada en el manejo y análisis de estructuras de datos. Nos permitirá manejar los datos a alto nivel y con una interfaz fácil e intuitiva.
- ***NLTK*** es una plataforma usada para construir programas para análisis de texto.
- ***spaCy*** es una biblioteca de procesamiento de lenguaje natural Python diseñada específicamente con el objetivo de ser una biblioteca útil para implementar sistemas listos para producción. Es particularmente rápido e intuitivo, por lo que es un competidor superior para las tareas de procesamiento de lenguaje natural (PLN).
- ***Tika***: Las herramientas Apache Tika detectan y extraen metadatos y texto de más de mil tipos de archivos diferentes.
- ***Gensim*** es una biblioteca de procesamiento de lenguaje natural de Python, que puede convertir documentos en modo vectorial de acuerdo con TF-IDF, LDA, LSI y otros modelos para su posterior procesamiento.
- ***Networkx*** es una biblioteca de Python para el estudio de grafos y análisis de redes.

Todas las versiones de dichas librerías se encuentras especificadas en el archivo *requirements.txt*, para la correcta ejecución del código.

Implementación

Para el desarrollo de los diferentes ítems de la práctica, se ha utilizado el paradigma de programación orientada a objetos ya que aporta claridad y sencillez al código.

Además, se decidió la creación de el resumen por capítulos para una mayor precisión y coherencia de los resúmenes resultado.

A continuación se exponen los diferentes pasos y elecciones tomadas a lo largo de la codificación.

Selección de red semántica

Para la creación de la red semántica, se ha seleccionado el modelo propuesto por ***Carlos García Hernán*** , siguiendo la recomendación realizada en la corrección de las redes semánticas.

Esta red semántica detecta de manera precisa los sujetos y los objetos así como las relaciones, de modo que aunque es básica y quizá los resultados podrían ser mejores con redes de mayor complejidad, para este caso de uso es perfectamente funcional.

Factor de compresión

Para la selección del factor de compresión se barajaron diversas opciones, y debido a que el resultado que se busca obtener se trata de un resumen de un contenido literario, se seleccionó el siguiente método:

Al seleccionar el factor de compresión afectamos al resumen a dos niveles:

1. A nivel de centroides: El factor de compresión determina el número de centroides seleccionados del total. Es decir, a mayor valor de compresión mas cercano estaremos a la selección de la totalidad de los centroides del grafo de la red semántica, mientras que a menor factor, se seleccionan los centroides con mayor grado.
2. A nivel de frase: El factor de compresión determina el numero de frases en total que se han de seleccionar, para, posteriormente, seleccionar de manera cronológica aquellas frases con mayor numero de apariciones de los centroides calculados anteriormente.

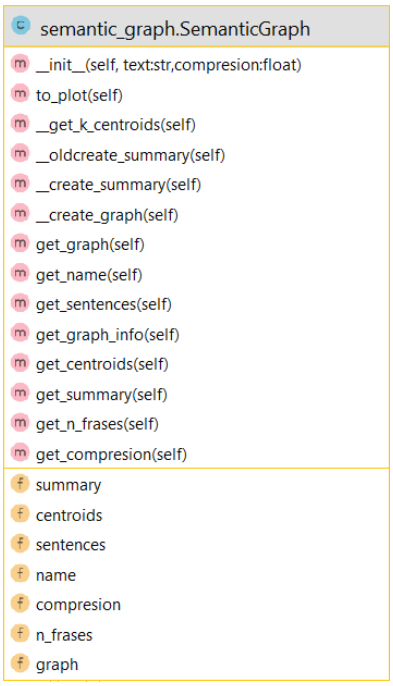
De este modo conseguimos que la longitud del resumen sea modulable y a la vez no perder el sentido del propio resumen.

Funciones auxiliares

En el código se hace uso de las siguientes funciones, contenidas en el fichero *tools.py*:

- **read_txt**: Lector del pdf haciendo uso de ***Tika.***
- **sent_to_words** y **remove_stopwords**: Preprocesamiento del texto para el correcto funcionamiento del modelo LDA.
- **LDA_analysis**: Haciendo uso de la librería ***Gensim*** realizamos un análisis LDA de cada capítulo.
- **output_summary**: Creación del texto output, con los centroides asociados al factor de compresión, las frases con mayor número de estos centroides, y los resultados de LDA.

Estructura del código



La clase *SemanticGraph* posee todos los métodos relacionados con el resumen del texto, y tiene los siguientes métodos privados para la generación del resumen y los centroides:

- **to_plot**: Dibujar el plot del grafo.
- **get_k_centroids**: Calcula los centroides correspondientes al factor de compresion seleccionado.
- **create_summary**: Creación del resumen en base a los centroides y al número de frases a seleccionar asociado el factor de compresión.
- **create_graph**: Creación del grafo semántico en base al Código de Carlos.
- Diversos **getter** y **setters** de utilidad para el código.

Con esta combinación de métodos públicos y privados conseguimos la generación de un resumen y la selección de los centroides más representativos a nivel de capitulo, de modo que en el **main** solo se repite la inicialización de la clase para cada capítulo.

Resultado

A continuación se adjunta un ejemplo de salida por capitulo, haciendo uso *MarkDow*n:

```
## Capitulo 0
### Centroides asociados al factor de compresión  0.01 :
[('Bilbo', 291), ('Frodo', 132), ('hobbit', 118), ('Bag', 101),
('End', 101), ('party', 99), ('Mr.', 91)]
### Resumen:
When **Mr.** **Bilbo** Baggins of **Bag** **End** announced that he
would shortly be celebrating his eleventy-first birthday with a
**party** of special magnificence, there was much talk and
excitement in Hobbiton.

When **Bilbo** was ninety-nine he adopted **Frodo** as his heir, and
brought him to live at **Bag** End; and the hopes of the Sackville-
Bagginses were finally dashed.

the fellowship of the ring  The eldest of these, and Bilbo’s
favourite, was young **Frodo** Baggins.

The history and character of **Mr.** **Bilbo** Baggins became once
again the chief topic of conversation; and the older folk suddenly
found their reminiscences in welcome demand.

‘A very nice well-spoken gentlehobbit is **Mr.** Bilbo, as I’ve
always said,’ the Gaffer declared.

But be that as it may, **Mr.** **Frodo** is as nice a young
**hobbit** as you could wish to meet.
```

```
A decent respectable **hobbit** was **Mr.** Drogo Baggins; there was
never much to tell of him, till he was drownded.'

I'd not long come prentice to old Holman (him being my dad's
cousin), but he    a long-expected **party** 31  had me up at
**Bag** **End** helping him to keep folks from tram- pling and
trapessing all over the garden while the sale was on.

### LDA:

**Topic**: 0
**Words**: 0.015*"said" + 0.010*"frodo" + 0.006*"ring" +
0.005*"long" + 0.005*"gandalf" + 0.004*"would" + 0.004*"one" +
0.004*"many" + 0.004*"go" + 0.004*"come"


**Topic**: 1
**Words**: 0.010*"frodo" + 0.008*"said" + 0.005*"could" +
0.005*"one" + 0.005*"long" + 0.005*"ring" + 0.004*"bilbo" +
0.004*"go" + 0.004*"gandalf" + 0.004*"like"


**Topic**: 2
**Words**: 0.019*"said" + 0.013*"frodo" + 0.007*"ring" +
0.005*"great" + 0.005*"came" + 0.005*"long" + 0.005*"gandalf" +
0.005*"come" + 0.005*"far" + 0.004*"dark"


**Topic**: 3
**Words**: 0.014*"said" + 0.009*"frodo" + 0.006*"came" +
0.005*"gandalf" + 0.005*"ring" + 0.004*"like" + 0.004*"away" +
0.004*"would" + 0.004*"back" + 0.004*"one"


**Topic**: 4
**Words**: 0.017*"said" + 0.011*"frodo" + 0.006*"ring" +
0.006*"could" + 0.005*"gandalf" + 0.005*"would" + 0.004*"sam" +
0.004*"one" + 0.004*"came" + 0.004*"like"
```

Conclusiones

Tras todo el proceso descrito anteriormente se obtienen resultados que son bastante acordes a la intuición, es decir, aquellos centroides mas representativos de los textos se suelen corresponder con los personajes principales, y dado que los resúmenes se generan en base al numero de estos centroides presentes en cada frase además de cronológicamente, los resultados suelen eliminar los fragmentos que no involucran a personajes principales en la acción.

Todas las etapas son mejorables a nivel de funcionamiento, pero el resultado es muy bueno. Debido a las limitaciones que nos produce la red semántica en sí misma, cupiera esperar resultados menos representativos, pero responde de manera muy correcta al texto analizado.