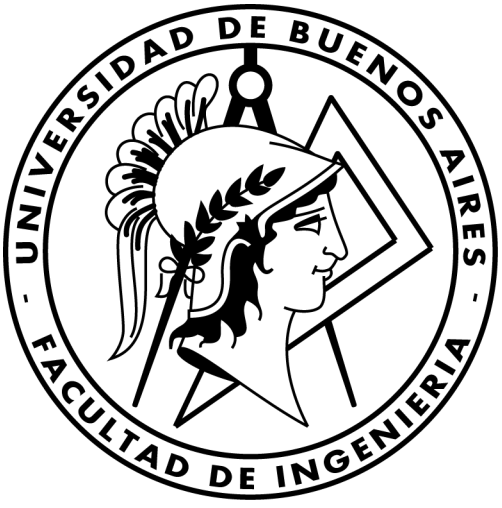

Universidad de Buenos Aires
Facultad de Ingeniería

Módulo de resúmenes automáticos basado en TextRank con integración a Gensim

Barrios, Federico – 92954
López, Federico – 92278

Mayo, 2015

Índice

1. Motivación

Los resúmenes automatizados son muy utilizados en tareas relacionadas con el procesamiento de lenguaje natural y de aprendizaje automático. Su uso en motores de búsqueda, por ejemplo, mejora la eficiencia de indexación de textos y a su vez asiste en la presentación de resultados de manera efectiva. El incremento en la cantidad de información disponible en Internet ha intensificado su utilización en los últimos años y, en consecuencia, se ha dedicado enorme esfuerzo para mejorar los algoritmos existentes. El desarrollo de este Trabajo Profesional está motivado por el número de aplicaciones del tema en cuestión en tareas de actualidad.

2. Objetivos

El presente trabajo consta de tres objetivos principales:

- Desarrollar el módulo para generar resúmenes automáticos usando un algoritmo conocido.
- Analizar, diseñar e implementar modificaciones para intentar mejorar el rendimiento del algoritmo seleccionado.
- Integrar las implementaciones a una herramienta de código abierto de procesamiento del lenguaje natural.

3. Introducción teórica

3.1. Resúmenes: generación y clasificación

Un resumen es una reducción a términos breves y precisos de lo esencial de una fuente de información. Su objetivo es el de extraer contenido intentando sintetizar sus conceptos más importantes, y su uso es altamente benéfico en tareas de aprendizaje debido a que:

- Facilitan la selección de información,
- Acortan tiempos de lectura,
- Simplifican búsquedas en textos,
- Optimizan la creación de índices.

La investigación indica, además, que contribuyen en tareas automatizadas: su utilización para presentar resultados de motores de búsquedas atrajo el interés de académicos desde principios de la década del 2000, convirtiéndose hoy en día en una funcionalidad básica de los principales buscadores de Internet.

Sin embargo, si bien esta tarea puede no resultar costosa para un ser humano, durante muchos años fue considerada como difícil de automatizar. Este tema es uno de los que investiga el campo de procesamiento de lenguaje natural, dedicado a facilitar la interacción entre las computadoras y los seres humanos.

3.1.1. Enfoque abstractivo y extractivo

Existen dos métodos para generar resúmenes automatizados: abstractivos y extractivos.

Los primeros se construyen regenerando el contenido extraído del texto original; esto es, se reformulan las frases por medio de técnicas de generación de lenguaje natural: fusión, combinación o supresión de términos. De esta manera, se obtienen pasajes que en principio no pertenecían al texto de origen, similar al modo en que lo harían las personas.

Por el contrario, los resúmenes extractivos se crean a partir de la selección de un conjunto de palabras u oraciones consideradas sobresalientes en el texto original. Las oraciones se obtienen literalmente, se unen libremente, y se presentan con el objetivo de crear un resumen del texto.

Las técnicas abstractivas son un área de creciente interés. Sin embargo, debido a la complejidad y las restricciones técnicas aparejadas, la investigación se ha volcado hacia el enfoque extractivo.

3.1.2. Aprendizaje supervisado y no supervisado

En el campo del aprendizaje automático, a los algoritmos que requieren de un conjunto de entrenamiento con datos etiquetados se denominan supervisados. Analizando estos ejemplos de entrenamiento, el sistema infiere funciones que pueden ser usadas para aplicarlas a nueva información.

Por el contrario, a los algoritmos que no son entrenados se los denomina no supervisados. Este grupo utiliza técnicas para encontrar patrones o grupos ocultos en los datos a analizar.

3.1.3. Fuente de los datos

Los algoritmos de generación de resúmenes también se categorizan de acuerdo a la cantidad de documentos que se toman como fuente, determinando sumarización de un documento y multi-documento.

Para este último caso se proporcionan textos con una temática afín. El resumen resultante permite a los lectores familiarizarse rápidamente con los tópicos principales de una colección de documentos.

3.1.4. Propósito del resumen

Los resúmenes de documentos pueden crearse de manera genérica o bien basados en consultas.

Aquéllos resúmenes basados en consultas favorecen temas o aspectos específicos de un texto, de acuerdo a dicha consulta; teniendo en cuenta sus palabras clave o los tópicos a los que hace referencia. Los resúmenes genéricos, por otro lado, proporcionan los temas más relevantes tratados en el texto por sí solo.

3.2. Extracción de palabras claves

La extracción de palabras claves consiste en identificar el conjunto de términos que mejor describen a un documento de manera automatizada. Esto sirve como introducción a sus conceptos más relevantes, pues uno de los primeros pasos para modelar el conocimiento de una comunidad es tomar vocabulario del dominio. Es por ello que se han estudiado métodos para extraer este léxico técnico en base a documentos del área o comunidad en estudio.

Dichas palabras se pueden utilizar para construir índices de colecciones, clasificar textos, extraer terminología de dominio específica, o incluso pueden servir como un breve resumen.

El enfoque más sencillo para esta tarea ha empleado un criterio basado en la frecuencia de las palabras. Sin embargo, se han investigado otras debido a los malos resultados de esta técnica. Las principales se basan en métodos supervisados, donde el sistema es entrenado para reconocer palabras claves tomando en cuenta cuestiones sintácticas y/o léxicas.

En la sección 'TextRank para palabras claves' se analiza y desarrolla la aplicación de TextRank como método no supervisado para esta tarea. Los resultados obtenidos se equiparan los de las técnicas principales en el área.

4. Trabajo previo

El campo de la generación automática de resúmenes ha atraido interés desde finales de la década de 1950 hasta la actualidad [1]. Los métodos tradicionales tienen en cuenta la frecuencia de palabras o frases introductorias para identificar las oraciones más sobresalientes del texto. También se han desarrollado modelos estadísticos basados en corpus de entrenamiento para combinar varias heurísticas: palabras clave, posición de las oraciones, longitud de las oraciones, frecuencia de palabras y palabras contenidas en los títulos [2]. Otros enfoques más modernos se basan en la representación del texto en forma de grafo: las oraciones importantes y los conceptos son las entidades altamente conectadas y, por esto, forman parte del resumen [3].

Con esta idea se usan métodos del campo de Recuperación de Información para identificar oraciones similares y determinar las más importantes, que formarán al resumen final [4]. El enfoque propuesto, tanto por Mihalcea y Tarau en TextRank [5] como por Erkan y Radev en LexRank [6], consiste en utilizar el prestigio de las unidades léxicas (oraciones o palabras) dentro del grafo.

5. Herramientas y fundamentos

5.1. PageRank

PageRank es un algoritmo creado en 1996 por Larry Page y Sergey Brin, fundadores de Google, como parte de una investigación sobre motores de búsqueda. Su objetivo es asignar un valor numérico a cada nodo en una red, con el propósito de medir la importancia relativa de cada elemento del conjunto. Para esto analiza los vínculos que componen la red. La idea subyacente es que los nodos más relevantes de la red serán los que tengan la mayor cantidad y calidad de conexiones con otros nodos.

En sus orígenes, PageRank fue pensado para aplicarse sobre la red de Internet, pero puede aplicarse a cualquier otra red para estimar la importancia de los nodos. En términos matemáticos, PageRank brinda una manera iterativa de calcular el autovector principal de la matriz de adyacencia del grafo. Esto da como resultado la distribución de probabilidad de acceder a los nodos de la red, navegando de manera aleatoria a través de los vínculos o aristas.

5.2. TextRank

TextRank es un algoritmo no supervisado basado en grafos para realizar resúmenes automáticos extractivos u obtener palabras claves de un texto. Fue presentado en 2004 por Rada Mihalcea y Paul Tarau en [5].

El algoritmo aplica una variación de PageRank [7] sobre un grafo especialmente diseñado para la tarea. De esta manera permite explotar la estructura del texto, identificando los conceptos principales, sin necesidad de datos previos de entrenamiento. Debido a que se basa en PageRank, se sirve de la noción del "prestigio" o "recomendación" entre los elementos del grafo. Por este motivo, TextRank puede ser aplicado a cualquier texto, incluso en distintos idiomas, generando un resumen basado sólo en las propiedades intrínsecas del texto.

El algoritmo modela el texto en base a un grafo, y luego, busca crear relaciones significativas (aristas) entre las entidades léxicas (vértices). Dependiendo de la aplicación que se desee dar al algoritmo, las entidades pueden ser palabras, frases, oraciones, párrafos, entre otros. De manera similar, también debe definirse el tipo de relación que se usa para unir los vértices: semántica, contextual, de superposición, y demás.

Los pasos principales que se llevan a cabo son los siguientes:

1. Identificar las unidades del texto y agregarlas al grafo como vértices.
2. Identificar relaciones que conectan a estas unidades, y agregarlas al grafo como aristas entre los vértices. Las aristas pueden ser dirigidas o no, y ponderadas o no.
3. Aplicar PageRank para asignarle un puntaje a cada vértice.
4. Ordenar los vértices de acuerdo al puntaje y utilizarlo para armar el resumen de acuerdo a algún criterio.

5.3. Gensim

Gensim es una biblioteca de código libre, escrita en Python, para el modelaje de tópicos y la indexación de documentos que está diseñada para trabajar con conjuntos de textos de gran tamaño. Su uso se ha expandido tanto en lo comercial como lo académico, apuntando especialmente al área de procesamiento del lenguaje natural y a la búsqueda y recuperación de la información.

Gensim utiliza un paradigma muy poderoso en el área de procesamiento de lenguaje natural, denominado VSM (modelo de espacio de vectores), en el cual se representan los documentos de un corpus como vectores de un espacio multidimensional. Esta representación explota la idea de que los textos en el área del modelado de tópicos se pueden expresar según un número de conceptos, lo que aumenta la eficiencia y también ayuda a eliminar ruido.

La biblioteca está diseñada para proveer independencia del tamaño del corpus, es decir, para poder analizar textos que sean más grandes que la memoria RAM del sistema. El proyecto, además apunta a exponer una interfaz mínima e intuitiva utilizando terminología de procesamiento de lenguaje.

Gensim proporciona implementaciones de algoritmos como TF-IDF, análisis semántico latente, proyecciones aleatorias y alocación de Dirichlet latente.

5.4. Evaluación de resúmenes

La herramienta por defecto para evaluar los sistemas generadores de resúmenes es Rouge (Recall-Oriented Understudy for Gisting Evaluation). Este método fue desarrollado en la universidad de Southern California para automatizar la comparación entre resúmenes generados por el sistema bajo evaluación contra otros (ideales) escritos por seres humanos, tomados como referencia.

Históricamente, los puntajes asignados en tareas de competencias como DUC fueron manuales; juzgando coherencia, capitalización, cohesión, consistencia y contenido [8]. Todo este trabajo es caro y difícil de coordinar, demandando de al menos 3.000 horas hombre. El surgimiento de Rouge atiende la necesidad de automatizar ese proceso, reemplazando un método llamado BLEU, cuyos resultados no siempre se correlacionaban con los asignados por los jurados.

El mecanismo de funcionamiento de los métodos del paquete Rouge consiste en calcular la sensibilidad (recall) de unidades léxicas entre los resúmenes de sistema y los de referencia. Se generan así diferentes métricas:

- Rouge-N: cuando se tienen en cuenta n-gramas (n palabras). Este método favorece a documentos con palabras compartidas por más candidatos.
- Rouge-L: cuando se tiene en cuenta la subsecuencia máxima común (LCS, longest common subsequence). Este método propone que un conjunto de documentos va a ser más parecido mientras más larga sea la subsecuencia común máxima.
- Rouge-W: le asigna pesos a las subsecuencias según su largo.
- Rouge-S: es similar a Rouge-2, teniendo en cuenta bigramas pero permitiendo una cantidad arbitraria de espacios entre palabras.
- Rouge-SU: extiende la idea de Rouge-S, pero además le otorga valor a las palabras que haya en común, sin importar que no verifiquen un orden.

Rouge asigna un puntaje entre cero y uno a un esquema de evaluación conformado por una serie de documentos generados y sus referencias. Por diseño, se espera que buenos resúmenes tengan puntajes más altos; pero inevitablemente se requiere de trabajo humano para proveer las referencias.

Debido a que los resultados arrojados por las métricas se correlacionan con los asignados manualmente, las evaluaciones en las competencias de generación de resúmenes automáticos se llevan a cabo usando exclusivamente Rouge.

6. Implementación

6.1. Preprocesamiento

El preprocesamiento del texto es una parte esencial en cualquier sistema que trate con datos del lenguaje natural. En esta fase se identifican las unidades léxicas que serán analizadas por la aplicación en las etapas subsiguientes.

La calidad de este proceso tendrá un gran impacto en los resultados obtenidos. Es por ello que se deben seleccionar unidades léxicas significativas. Es decir, aquellas que contengan las características lingüísticas relevantes para el caso en estudio. A su vez, se deben filtrar las que no aportan información útil. En este proceso también, se unifica la codificación que pudiera existir en los distintos textos.

Aquí se detallan las etapas principales.

6.1.1. Separación del texto

Esta etapa, también conocida como *tokenización* consiste en separar el texto en oraciones o palabras. Separar palabras suele resultar más sencillo, dado que los espacios en blanco son una buena aproximación como límite de separación.

Separar por oraciones suele ser caso más complejo. Definir los límites es una cuestión realmente problemática en el área del lenguaje natural. Generalmente se utilizan los símbolos .!? como separadores, pero se deben tratar con cuidado excepciones como *Ph.D.*, *Dr.*, *Lic.*, *$19.99*, *p.m.* o *RR.HH.* El enfoque más utilizado para resolver esta cuestión se basa en la aplicación de expresiones regulares, aunque existen otras alternativas que implementan algoritmos entrenados con datos específicos del área sobre la cual se esté trabajando.

6.1.2. Filtrado

Las palabras más frecuentes muchas veces no tienen tanto significado por sí solas. En muchas tareas de procesamiento de lenguaje natural estas palabras aportan "ruido", y por lo tanto se las debe quitar.

La forma de filtrar palabras consiste en crear una lista de "palabras prohibidas" (*stop-words*) a ser eliminadas del texto.

6.1.3. Stemming

El proceso de *stemming* o *lemmatización* consiste en quitar la desinencia de las diferentes términos, para llevarlos a su raíz. Esta etapa se lleva a cabo con el objetivo de que el algoritmo detecte cuando se están tratando los mismos conceptos, dado que se utilizan las mismas palabras. Por ejemplo, las palabras "canta", "cantaba", "cantando", pueden ser asociadas al verbo "cantar".

El stemming es un proceso complejo dado que puede haber muchos casos excepcionales, y se necesitan reglas muy diferentes para cada idioma. El stemmer más usado en inglés es el Porter Stemmer. Para cada idioma existe una adaptación. Sin embargo, todos estos algoritmos tienen cierta tasa de error.

6.2. TextRank para extracción de oraciones

El problema de la extracción de oraciones apunta a identificar las secuencias más representativas del texto. Para este caso, las unidades léxicas tenidas en cuenta para aplicar el algoritmo serán oraciones completas [9].

Inicialmente, se construye un grafo en base al texto, agregando a cada oración como vértice. Para crear las aristas con su peso correspondiente, se debe definir una función de similitud entre dos oraciones dadas. Esta función será la que dicte cuánto una oración "recomienda" a otra por poseer contenidos similares y abordar los mismos conceptos.

La función utilizada formalmente por el algoritmo original se define de la siguiente manera:

Definición 1 *Sean S_i, S_j dos oraciones representadas por un conjunto de n palabras que en S_i aparecen como $S_i = w_1^i, w_2^i, ..., w_n^i$. La función de similitud para S_i, S_j se define como:*

$$Similitud(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)} \tag{1}$$

El resultado de este proceso es el texto representado como un grafo altamente denso, al cuál se le aplicará PageRank para seleccionar los vértices más relevantes. Finalmente, a las oraciones más importantes se las presenta de acuerdo al orden de aparición en el texto original.

6.3. TextRank para palabras claves

La otra aplicación que se le puede dar a TextRank es la extracción de palabras clave (*keywords*) de un texto. Es un problema similar a la extracción de oraciones, dado que el objetivo es identificar un conjunto de unidades que sean representativas de dicho texto. Aquí, las unidades serán palabras, sueltas o combinadas.

Al igual que en el caso de la extracción de oraciones, luego de la etapa de preprocesamiento, las palabras no filtradas se agregan al grafo. Se pueden agregar filtros sintácticos adicionales para limitar el análisis sólo a, por ejemplo, sustantivos y verbos, o cualquier otra combinación.

En lugar de una función de similitud, aquí se define una función de coocurrencia: dos vértices estarán conectados si las palabras que representan ocurren en el texto dentro de una ventana de n palabras, que suele tomar un valor entre dos y diez. Los vínculos de coocurrencia demuestran una relación entre los elementos sintácticos que sirve como indicador de la cohesión del texto. Es por ello que son útiles para indicar la relevancia de las palabras principales.

Con el grafo no ponderado y no dirigido construido, el valor de cada vértice se establece en uno, y se aplica el PageRank. Una vez asignados los puntajes, se toman los k vértices más relevantes, donde k suele ser un tercio de la cantidad de vértices totales.

Finalmente, las palabras preseleccionadas se marcan en el texto. En caso que estas palabras formen secuencias adyacentes, se las une en un "concepto clave". Por ejemplo, si se

preseleccionan las palabras “Puerto” y “Rico”, y en el texto figuran juntas, se las combina en “Puerto Rico”.

6.4. Métodos de evaluación

Considerando que parte del alcance planteado incluye la propuesta de mejoras al método, se trabajó con el paquete ROUGE, desarrollado en lenguaje Perl. Se usó un proyecto de Python que funciona como wrapper para adaptar los textos de entrada (pyrouge) y ejecutar automáticamente la evaluación.

Se implementó, además, una capa de abstracción adicional para acotar la cantidad de opciones que se manejan, y para automatizar la corrida de múltiples textos de una base de datos.

Al ser un paquete de evaluación muy completo y versátil, Rouge provee una gran cantidad de parámetros. Se configuraron sus opciones como se usan en las conferencias de DUC, calculando ROUGE-1, ROUGE-2 y ROUGE-SU4 en un intervalo de confianza del 95 % y aplicando un método de stemming [10].

Una de las restricciones que posee este método de evaluación es que requiere de resúmenes hechos por seres humanos para tomar como referencia, lo que implica un esfuerzo humano considerable si se quiere probar el rendimiento del método con varios textos. Se utilizó el corpus de DUC edición 2002, por ser en el que se desarrollaron las pruebas del paper original de TextRank.

Todos los resultados obtenidos se compararon contra un sistema de referencia (*baseline*) que genera resúmenes que consisten de las primeras 100 palabras de cada texto [11]. Se configuró un ambiente de integración continua para permitir la ejecución de evaluaciones automatizadas y que además verifica la calidad del código programado.

7. Mejoras

En esta sección se analizan las propuestas de mejora que arrojaron mejores resultados. Los métodos listados a continuación consisten en redefinir la similaridad entre las oraciones. El listado total de ensayos realizados y sus puntajes se puede consultar en el Anexo I.

7.1. Propuestas

7.1.1. Subcadena común

Dadas dos frases, el problema de la subcadena común más larga consiste en identificar la secuencia de caracteres de mayor extensión presente en ambas. Por ejemplo, entre “la cocina verde” y “una cocina vale más si es verde”, la secuencia más larga es “a cocina v” [12].

La propuesta consiste en modificar la función de distancia de la implementación original y reemplazarla por el largo de la subcadena común más larga. En el ejemplo anterior, la similitud sería de 10, dado que es el largo de la secuencia “a cocina v”.

7.1.2. Similitud coseno

La similitud coseno es una medida del parecido entre dos vectores en un espacio que posee un producto interior, y resulta de evaluar el valor del coseno del ángulo comprendido entre ellos. Para poder hacer uso de esta propiedad se utiliza el modelo vectorial, modelando a los documentos como vectores.

Este modelo algebraico es utilizado para representar documentos en lenguaje natural de una manera formal, y tiene la característica de que favorece la dirección a la cuál apuntan los documentos independientemente de su longitud. Consecuentemente, textos que hablan de los mismos temas en distinta cantidad de palabras pueden tener una gran similitud con esta métrica [13].

La propuesta se basa en aplicar este modelo para tratar a cada oración del texto como un vector n-dimensional (siendo n la cantidad de palabras distintas presentes en el documento), y luego compararlas utilizando la similitud coseno. Las componentes de cada vector estarán compuestas por el resultado de aplicar la función “frecuencia de término – frecuencia inversa de documento” (TF–IDF de sus siglas en inglés) a cada palabra de la oración representada.

Tomando como ejemplo el documento “Esta bien. Todo bien.”, las oraciones quedarían modeladas como vectores de la siguiente forma:

```
v1=[ TFIDF("Esta") , TFIDF("bien") ,       0       ]
v2=[       0       , TFIDF("bien") , TFIDF("Todo") ]
```

Dado que la imagen de la función TF–IDF está contenida en el intervalo [0,1], todos los vectores quedan conformados por entradas no negativas, haciendo que ninguna similitud sea menor a cero.

7.1.3. BM25

BM25 / Okapi BM25 es una función de ranking utilizada como estado de arte para tareas de Recuperación de Información. Esta función es una variación del modelo TF–IDF usando un modelo probabilístico [14].

Definición 2 *Dadas dos oraciones R, S, BM25 se define como:*

$$BM25(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot(1-b+b\cdot\frac{|D|}{avgDL})} \tag{2}$$

donde k y b son parámetros establecidos a $1,2$ y $0,75$, respectivamente. $avgDL$ es el largo promedio de las oraciones en el texto.

Esta función define que el valor de aquellas palabras que aparecen en más de la mitad de los textos resulte negativo. Dado que este comportamiento puede traer problemas en las fases posteriores del algoritmo, se aplica la siguiente fórmula correctiva:

$$IDF(q_i) = \begin{cases} log(N - n(q_i) + 0{,}5) - log(n(q_i) + 0{,}5) & \text{si } n(q_i) > N/2 \\ \varepsilon \cdot avgIDF & \text{si } n(q_i) \leq N/2 \end{cases} \tag{3}$$

donde ε ronda entre 0,5 y 0,30 y avgIDF es el idf promedio para todos los términos. También se ensayaron otras alternativas como establecer $\varepsilon = 0$ y usar modificaciones más simples a la fórmula clásica de IDF.

También se experimentó con una variante de este método, conocida como BM25+, que repara deficiencias de BM25 relacionadas a la frecuencia de término y a la penalización a documentos largos frente a documentos cortos irrelevantes [15].

7.2. Resultados

Los mejores resultados se obtuvieron usando BM25 y BM25+. El incremento más alto se logró al reemplazar los valores negativos por la constante $\varepsilon = 0{,}25$ en la ecuación 3, dando una mejora total de 2,92 % para BM25 y 2,60 % para BM25+. En el Cuadro 1 se detallan las alternativas ensayadas.

Los tiempos de ejecución también se superaron. Se pudieron procesar los 567 documentos de la base de datos de DUC2002 utilizando 84 % del tiempo requerido por la versión original.

El resultado de la métrica de similitud coseno también fue satisfactoria, presentando una mejora de 2,54 % por sobre el método original. A su vez, los métodos de secuencias de máxima longitud también mostraron una mejora considerable: cerca del 1,40 % por sobre TextRank. En la Figura 1 se comparan los distintas técnicas.

8. Integración a Gensim

La integración del módulo a Gensim[1] se desarrolló a partir de su última versión, la 0.11.1-1. Se agregan a la biblioteca ambas funcionalidades descriptas anteriormente: la extracción de palabras claves y la generación automática de resúmenes extractivos.

En la integración se incluyen tres casos de uso:

- Generación de resúmenes automáticos dado un texto. En este caso la función requiere una cadena de texto con el documento a resumir, y devuelve el texto resumido.
- Selección de documentos más importantes dado un corpus. Este caso es similar al anterior, con la diferencia de que se recibe un listado de documentos (que pueden ser oraciones de un texto) y se devuelve un listado de aquéllos más importantes.
- Obtención de palabras claves de un texto. En este caso se devuelve un listado de palabras clave dada una cadena de texto.

[1] Pull request: `https://github.com/piskvorky/gensim/pull/324`

Cuadro 1: Resultados de las distintas propuestas

Método	ROUGE-1	ROUGE-2	ROUGE-SU4	Mejora
BM25 (Neg a epsilon)	0.4042	0.1831	0.2018	2,92 %
BM25+ (Neg a epsilon)	0.404	0.1818	0.2008	2,60 %
Cosine TF-IDF	0.4108	0.177	0.1984	2,54 %
BM25+ (IDF = log(N/NI))	0.4022	0.1805	0.1997	2,05 %
BM25 (IDF = log(N/NI))	0.4012	0.1808	0.1998	1,97 %
Longest Common Substring	0.402	0.1783	0.1971	1,40 %
BM25+ (Neg a cero)	0.3992	0.1803	0.1976	1,36 %
BM25 (Neg a cero)	0.3991	0.1778	0.1966	0,89 %
TextRank	**0.3983**	**0.1762**	**0.1948**	**-**
BM25	0.3916	0.1725	0.1906	-1,57 %
BM25+	0.3903	0.1711	0.1894	-2,07 %
DUC Baseline	0.39	0.1689	0.186	-2,84 %

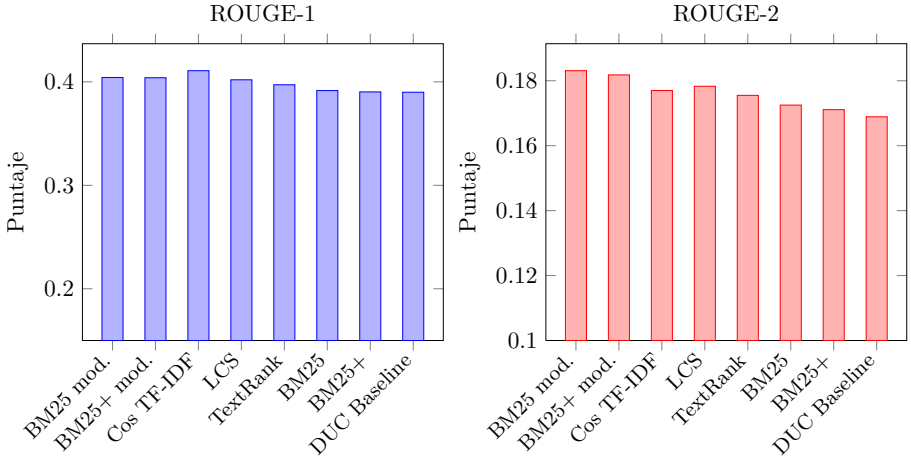

Figura 1: Comparación de métricas.

9. Conclusiones

En este trabajo se analizaron variantes al algoritmo de TextRank. A partir del mismo se propusieron e implementaron optimizaciones cuyos resultados fueron significativos: se obtuvo una mejoría del 2,92 % por sobre el método original. Este número es notable si se tiene en cuenta que TextRank por sí solo performa 2,84 % por sobre el estándar de comparación.

Las evaluaciones fueron hechas con los mismos conjuntos de datos y métricas utilizados en competencias internacionales, validando así las mediciones.

Finalmente, la adaptación a la biblioteca de lenguaje natural Gensim también resultó sencilla gracias a las facilidades que provee su interfaz de programación y su gran cantidad de métodos.

Queda para próximos trabajos explorar alternativas para la extracción de palabras claves, y el ensayo de distintos métodos algebraicos, haciendo uso del modelo del espacio vectorial.

10. Anexo I: modificaciones

Además de las mejoras descriptas en la sección "Mejoras", también se ensayaron otras ideas cuyos resultados no fueron tan destacables como los mencionados.

Como se explicó anteriormente, las modificaciones apuntan a cambiar el criterio de similaridad entre oraciones. Algunas modificaciones que se tuvieron en cuenta son similares a la de subcadena común, que otorga puntajes de acuerdo a la cantidad de unidades léxicas en orden que estén presentes en las dos oraciones. Estas métricas son similares a las usadas por el paquete ROUGE, y los métodos de subsecuencia de palabras común, subsecuencia de palabras común con peso, bigramas comunes y SU4 son idénticos a los usados por los métodos ROUGE-L, ROUGE-W, ROUGE-2 y ROUGE-SU4.

Se implementaron también métricas de distancias de Levenshtein, un método que pondera con un valor más alto a las palabras clave del texto, y una lógica de envejecimiento que asigna puntaje adicional a aquéllas palabras que se mencionaron más recientemente.

Finalmente, se experimentó también reemplazando el algoritmo de PageRank (que utiliza como solución el autovector asociado al valor asociado más grande de la matriz de similaridad) por el de HITS (que utiliza el vector singular asociado al valor singular más grande de la matriz); y también se probó aproximando la solución reduciendo el rango de la matriz.

10.1. Resultados

Método	ROUGE1	ROUGE2	ROUGESU4	Mejora	Tiempo
TextRank	0.3983	0.1762	0.1948	-	0:00:25
BM25 (Neg a epsilon)	0.4042	0.1831	0.2018	2,92 %	0:00:21
BM25+ (Neg a epsilon)	0.404	0.1818	0.2008	2,60 %	0:00:21
Cosine TF-IDF	0.4108	0.177	0.1984	2,54 %	0:03:30
BM25+ (IDF = log(N/NI))	0.4022	0.1805	0.1997	2,05 %	0:00:24
BM25 (IDF = log(N/NI))	0.4012	0.1808	0.1998	1,97 %	0:00:22
Longest Common Substring	0.402	0.1783	0.1971	1,40 %	0:18:32
LCW Subsequence	0.4032	0.1773	0.1968	1,38 %	0:01:02
Weighted LCW Subsequence	0.4032	0.1773	0.1968	1,38 %	0:01:12
BM25+ (Neg a cero)	0.3992	0.1803	0.1976	1,36 %	0:00:17
BM25 (Neg a cero)	0.3991	0.1778	0.1966	0,89 %	0:00:22
Distancia temporal	0.3992	0.1762	0.195	0,48 %	0:03:15
Levenshtein	0.3961	0.1781	0.1955	0,38 %	0:27:49
Longest Common Subsequence	0.396	0.1775	0.1949	0,22 %	0:28:08
SU4	0.3955	0.1732	0.1924	-0,73 %	0:01:02
Prioridad keywords	0.3952	0.1721	0.1915	-1,03 %	0:07:13
Peso de conceptos	0.3953	0.1713	0.1914	-1,14 %	0:04:09
Distancia temporal invertida	0.3913	0.1725	0.1912	-1,53 %	0:03:05
BM25	0.3916	0.1725	0.1906	-1,57 %	0:00:18
HITS	0.3914	0.1699	0.1898	-2,03 %	0:00:46
BM25+	0.3903	0.1711	0.1894	-2,07 %	0:00:17
HITS + DVS	0.3912	0.1686	0.189	-2,33 %	0:00:46
DUC Baseline	0.39	0.1689	0.186	-2,84 %	0:00:01
Bigramas comunes	0.393	0.1633	0.1842	-3,42 %	0:00:50
Weighted LCSubstring	0.3755	0.1509	0.1729	-8,79 %	0:17:23
Levenshtein normalizado	0.348	0.1374	0.1565	-16,28 %	0:28:13

Referencias

[1] Luhn, H. P.: *The Automatic Creation of Literature Abstracts.* IBM J. Res. Dev., 2(2):159–165, Abril 1958, ISSN 0018-8646. `http://dx.doi.org/10.1147/rd.22.0159`.

[2] Lin, Chin Yew y E. H. Hovy: *Identifying Topics by Position.* En *Proceedings of 5th Conference on Applied Natural Language Processing*, Washington D.C., March 1997.

[3] Barzilay, Regina y Kathleen McKeown: *Sentence Fusion for Multidocument News Summarization.* Computational Linguistics, 31(3):297–328, 2005. `http://dblp.uni-trier.de/db/journals/coling/coling31.html#BarzilayM05`.

[4] Salton, G., A. Singhal, M. Mitra y C. Buckley: *Automatic Text Structuring and Summarization.* Information Processing and Management, 33(2):193 – 207, 1997.

[5] Mihalcea, Rada y Paul Tarau: *TextRank: Bringing Order into Texts*. En Lin, Dekang y Dekai Wu (editores): *Proceedings of EMNLP 2004*, páginas 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[6] Erkan, Günes y Dragomir R. Radev: *LexRank: Graph-based Lexical Centrality as Salience in Text Summarization*. J. Artif. Intell. Res. (JAIR), 22:457–479, 2004. `http://dblp.uni-trier.de/db/journals/jair/jair22.html#ErkanR04`.

[7] Page, L., S. Brin, R. Motwani y T. Winograd: *The PageRank citation ranking: Bringing order to the Web*. En *Proceedings of the 7th International World Wide Web Conference*, páginas 161–172, Brisbane, Australia, 1998. `citeseer.nj.nec.com/page98pagerank.html`.

[8] *Procedure for human comparison of model (reference) and peer (system-generated and other) abstracts*. `http://www-nlpir.nist.gov/projects/duc/duc2002/protocol.html`, 2002.

[9] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze: *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[10] *DUC 2007: Task, Documents, and Measures*. `http://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html#main`, 2007.

[11] *DUC 2002: Automatically produced baseline abstracts*. `http://www-nlpir.nist.gov/projects/duc/duc2002/baselines.html`, 2002.

[12] Gusfield, Dan: *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA, 1997, ISBN 0-521-58519-8.

[13] Singhal, Amit: *Modern Information Retrieval: A Brief Overview*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 24(4):35–43, 2001. `http://singhal.info/ieee2001.pdf`.

[14] Robertson, S.E., S. Walker, S. Jones, M.M. Hancock-Beaulieu y M. Gatford: *Okapi at TREC-3*. En *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, páginas 109–126, 1994. `http://trec.nist.gov/pubs/trec3/papers/city.ps.gz`.

[15] Lv, Yuanhua y ChengXiang Zhai: *Lower-bounding term frequency normalization*. En *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, páginas 7–16, 2011.