UNIVERSIDAD
REY JUAN CARLOS

MÁSTER EN INGENIERÍA EN SISTEMAS DE DECISIÓN

Curso Académico 2018/2019

Trabajo Fin de Máster

Marco de trabajo para evaluar la relevancia de los artículos en el dominio científico

Autor : Adrián Alonso Barriuso

Tutor : Dr. Alberto Fernández Isabel

Dedicado a

mi familia, pareja, amigos y a todos los que me aguantan, en el buen sentido.

Agradecimientos

Resumen

Summary

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto

1.1.1. Dominio de aplicación

1.2. Objetivos

1.2.1. Objetivo General

1.2.2. Objetivos específicos

-
-

1.3. Estructura de la memoria

1. Estado del arte:
2. Propuesta:
3. Experimentos y resultados:
4. Conclusiones:

Capítulo 2

Estado del arte

2.1. Algoritmos de reputación

2.2. Obtención de relevancias

Capítulo 3

Propuesta

En este capítulo, se describe la propuesta del sistema completo, definiendo entradas y salidas explicadas a nivel de diseño. Se empieza con una subsección donde se ve la arquitectura y el propósito general y después se entra en detalle para cada uno de los módulos en las subsiguientes secciones.

3.1. Arquitectura general

En la Figura 3.3, se pueden observar el módulo principal: *Relevance estimator module*, que es el encargado de estimar las relevancias de los artículos de entrada, y sus correspondientes submódulos: *Text processor, Reputation calculator y Relevance calculator*. Se cuenta además, con dos fuentes de información precalculadas utilizadas por el submódulo *Relevance calculator*, a saber, *Relevance lexicon* que consiste en cuatro lexicones[1] de relevancias de términos médicos y *Neural network*, que consiste en cuatro modelos entrenados de redes neuronales convolucionales (CNN)[2] por sus siglas en inglés. Por otra parte, se cuenta con un módulo de visualización(*Visualization*), que se utiliza para ver la salida del sistema y para proporcionar la entrada (*Text*). Por último, se utiliza una fuente de información externa en tiempo real (*Web information resources*).

En primer lugar se entra en detalle en cómo se construye Relevance lexicon, después Neural network y, por último, Relevance estimator module, el cuál emplea todo lo anterior para calcular las relevancias de nuevos documentos.

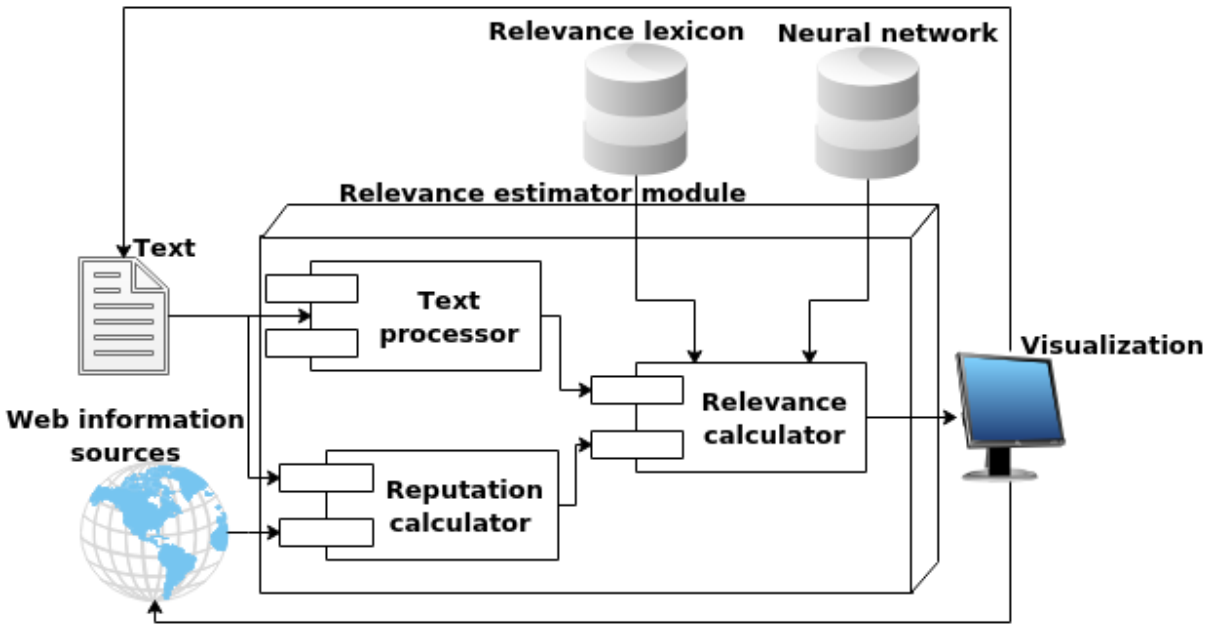

Figura 3.1: Arquitectura general

3.2. Creación del lexicón de relevancias

El lexicón de relevancias tiene un papel fundamental a la hora de estimar la relevancia de los artículos, en concreto, contiene la relevancia de las palabras de el dominio de aplicación utilizado para crear el mismo. Para la creación del lexicón se ha diseñado un marco de trabajo completo, cuya arquitectura puede verse en la Figura **??**. Esta cuenta con un módulo de extracción, transformación y carga (ETL por sus siglas en inglés)[3], un módulo de gestión de conomicimiento (Knowledge managment) y uno de visualización. Se cuenta, además, con dos bases de datos, una orientada a documentos[**?**](Document knowledge consolidation) y una ElasticSearch[**?**] para visualización con Kibana [**?**].

3.2.1. Módulo ETL

El módulo ETL se encarga de la obtención y el preprocesado del corpus de artículos médicos. Se divide a su vez en dos submódulos: Corpus processor y Reputation calculator. Corpus processor obtiene artículos de Pubmed Central[1] utilizando técnicas de web scrapping [4]. Estos artículos en formato XML [5], son parseados y almacenados en formato JSON [6] en Document

[1] `https://www.ncbi.nlm.nih.gov/pmc/`

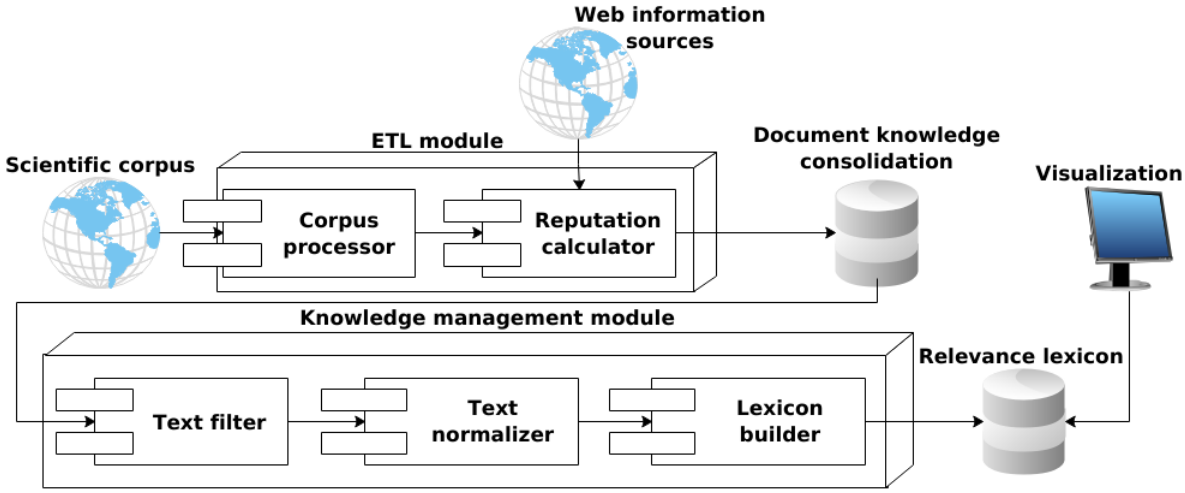

Figura 3.2: Arquitectura de generación de lexicones

knowledge consolidation con una estructura de párrafos y frases, eliminando tablas, gráficas u otros artefactos no aprovechables por el sistema. Una vez un documento se almacena en la base de datos, se procede a calcular su resumen automático aplicando el clásico algoritmo Text Rank [7] utilizando una popular implementación en Python [8]. Este tipo de resúmenes ofrecen una ordenación de las frases de un documento por relevancia, sirviendo como factor de filtrado de información poco relevante o redundante y a la vez como factor de compresión, haciendo la información más manejable en memoria. Se obtiene un 20 porciento del tamaño original de los artículos y se almacena como un nuevo artibuto del documento en la base de datos.

Una vez preprocesados y almacenados los artículos, emphReputation calculator calcula la reputación de los mismos para que esta sea añadida como nuevo atributo y ser utilizada posteriormente por el módulo Knowledge managment. El algoritmo de reputación empleado es una adaptación del introducido en el artículo [9]. La reputación a priori de un artículo viene dada por:

$$rep_p = \alpha \cdot rep\_authors_p + (1 - \alpha) \cdot citations_p \,, \tag{3.1}$$

donde $rep\_authors_p$ es la reputación media del los autores del artículo y $citations_p$ su número de citas total en el momento de la consulta. El parámetro $\alpha \in (0, 1)$ actúa como modulador de importancia relativa entre las citas y los autores. La reputación de cada autor viene dada por:

$$rep_i = \omega_1 \cdot inf\_citation\_count + \omega_2 \cdot citation\_velocity + \omega_3 \cdot seniority + \omega_4 \cdot papers \,, \tag{3.2}$$

donde $\sum_{i=1}^{4} \omega_i = 1$. El parámetro $inf\_citation\_count$ representa el número de citas altamente influyentes del autor. [**?**]. $citation\_velocity$ indica lo popular que es el autor durante los últimos 3 años. $seniority$ es la cantidad de años transcurridos entre la primera y última publicación del autor. Por último, $papers$ es el número total de artículos publicados por el autor. Estos parámetros son extraídos de la API REST de Semantic Scholar[10] a través del Identificador Digital de Objeto (DOI, por sus siglas en inglés)[**?**] del documento. Una vez que el módulo Reputation Calculator ha calculado la reputación del artículo, esta se añade a *Document knowledge consolidation*, como un atributo nuevo.

3.2.2. Módulo de Gestión del conocimiento

Este módulo consulta *Document knowledge consolidation* y sirviéndose de 3 submódulos, construye finalmente el lexicón. Estos 3 submódulos son Text filter, Text normalizer y Lexicon builder.

3.2.3. Text filter

Este módulo analiza los resúmenes extraidos por el módulo ETL por cada documento, aplicando técnicas de procesamiento de lenguaje natural (NLP) [11] extrayendo los sustantivos y eliminando palabras vacías [12], tanto genéricas como de dominio médico[13] y académico[**?**]. Finalmente, se obtienen los lemas de los sustantivos, lo que permite cierta desambiguación, ya que el lema de una palabra depende de la función sintáctica de la misma. Además, se resuelven posibles conflictos entre mayúsculas, minúsculas, singulares y plurales.

3.2.4. Text normalizer

Este submódulo construye una matriz de términos por documentos [14] a partir de la cuál construye la matriz TF-IDF [15]. Los pesos resultantes se combinan con las reputaciones de los artículos dando la medida de relevancia de cada término rel_lex_t. La relevancia de cada término t en los N artículos pertenecientes al corpus C viene dada por:

$$rel\_lex_t = \log\left(\frac{1}{N} \cdot \sum_{p=1}^{N} \beta \cdot tfidf(t)_p + (1-\beta) \cdot rep_p\right), \forall p \in C\,, \tag{3.3}$$

donde rep_p es la reputación del artículo p, $tfidf(t)_p$ es el valor TF-IDF del término t en el artículo p y $\beta \in (0, 1)$ es otro parámetro que modula la importancia relativa del valor TF-IDF sobre la reputación. Cabe destacar que $tfidf(t)_p \in (0, 1)$ ya que han sido normalizados por simplicidad y se aplica el logaritmo para normalizar la distribución.

3.2.5. Lexicon builder

Este componente construye y organiza el lexicón a partir del texto normalizado. Contempla además la posibilidad de ponderar aún más el peso de aquellos términos de dominio específico proporcionados por un diccionario, médico en este caso [16].

Se organizan y construyen lexicones por cada conjunto de artículos correspondientes a cada año disponible, teniendo en cuenta la evolución de la relevancia de cada término a lo largo de los años. De esta manera se puede modular la curva de olvido [17] y tener en consideración las tendencias del dominio de aplicación.

Se construyen diferentes valores de relevancia para cada término específico $rel\_lex_t(y)$ de acuerdo a un año específico y, manteniendo las palabras del año anterior en el nuevo de la forma:

$$rel\_lex_t(y) = \rho \cdot rel\_lex_t + (1 - \rho) \cdot rel\_lex_t(y - 1)\,, \tag{3.4}$$

donde rel_lex_t es la relevancia proporcionada por el marco de trabajo para el término t y $rel\_lex_t(y - 1)$ es la correspondiente al año anteruir $y - 1$. El parámetro ρ controla el peso del año anterior.

3.3. Creación del la red neuronal

Por muy grande que sea el corpus que se utilice para la creación de los lexicones, es virtualmente imposible contar con la relevancia de todos los términos existentes. Por tanto, se ha desarrollado una Red Neuronal Convolucional (CNN, por sus siglas en inglés)[2] para servir de apoyo al sistema. El propósito de la red es predecir la relevancia de aquellas frases que no contengan ninguna palabra presente en el lexicón. En la Tabla 3.1 se puede ver la configuración de la misma, la cuál es una versión optimizada de la aproximación introducida en [18]. Cabe destacar que la capa de *embedding* utiliza un modelo pre-entrenado de Glove [19] con un vocabulario

de $400,000$ palabras de Wikipedia [20]. La capa de salida cuenta con activación *softmax*, la cuál proporciona valores entre 0 y 1. En las capas ocultas se cuenta con convoluciones, funciones de activación *relu* y funciones de *dropout*.

Para construir la red, se propone la siguiente metodología: En primer lugar, el usuario selecciona un conjunto de artículos del corpus que no hayan sido utilizados para construir el lexicón. Después, se procesa el texto separando por frases y términos, eliminando palabras vacías y lematizando de manera análoga a como se procede en el módulo de ETL, de esta manera se aumenta el número potencial de coincidencias entre el texto utilizado para la creación de la red y las palabras del lexicón. Se etiquetarán como relevantes (1) aquellas frases con mayor relevancia de acuerdo al lexicón y como no relevantes (0) las menor relevancia o aquellas que no contengan palabras del lexicón. Se define, por tanto, un umbral de relevancia ϵ para elegir el mínimo necesario de acuerdo a lo estricta que se quiere que sea la red neuronal. Finalmente se debe elegir el número de frases etiquetadas para conformar el conjunto de entrenamiento y test. La red neuronal resultante debería ser capaz de prececir la relevancias de las frases sin palabras del lexicón.

Layers
1. Embedding input_dim 400000 output_dim 50
2. Dropout rate 0.4
3. Conv1D 250 filters of 3 with stride 1
4. Pool1D (max) with stride 1
5. Dense units 250
6. Dropout rate 0.4
7. Relu
8. Dense units 1
9. Softmax

Tabla 3.1: Capas de la red neuronal convolucional.

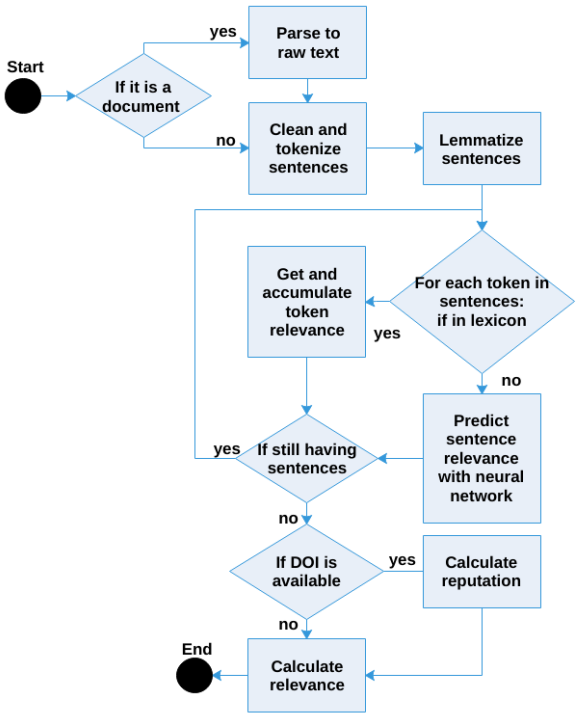

Figura 3.3: Flujo de trabajo de cálculo de relevancia de un artículo

3.4. Estimación de relevancias de artículos

Se ha diseñao un flujo de trabajo para ilustrar como funciona todo el proceso del marco de trabajo. Se comienza eligiendo un texto para evaluar y se concluye devolviendo su relevancia normalizada entre 0 y 1 (ver Figura. **??**).

En primer el sistema detecta si el documento cuya relevancia se desea conocer consiste en texto plano o un fichero (de tipo PDF, html o similar). En caso de ser un fichero, este parseado, limpiado y partido en frases y párrafos. Estas tareas se llevan a cabo en los pasos *Parse to raw text* y *Clean and tokenize sentences*, respectivamente.

Una vez se tienen mapeados los párrafos en listas de lemas de sustantivos, se consulta el lexicón por cada lista, acumulando el valor de relevancia de cada lema. En caso de que una frase no contenga ningún lema presente en el lexicón, la lista de lemas se tranforma en los vectores que entran a la red neuronal, que devuelve la predicción de relevancia. Estas tareas son llevadas a cabo por el módulo *Relevance calculator*. Una vez recorridas todas las listas de lemas, la relevancia viene dada por la Ecuación:

$$combined\_rel_p = \theta \cdot rel\_lex_p + (1-\theta) \cdot \frac{1}{K} \sum_{k=1}^{K} rel\_neural(s_k)\,, \tag{3.5}$$

donde rel_lex_p es la relevancia media del artcículo proporcionada por el lexicón, $\{s_k\}_{k=1}^{K}$ es el conjunto de frases cuya relevancia es desconocida y rel_neural es la relevancia predicha por la red neuronal para cada s_k. El parámetro $\theta \in (0, 1)$ parameter modula la importancia relativa de la red neuronal.

En última instancia, en caso de que la entrada al sistema sea un documento y no texto plano, se comprueban los metadados del mismo para tratar de encontrar el DOI. En caso de estar disponible, el módulo ***Relevance calculator***, calcula su reputaciñón y esta es utilizada para calcular la relevancia final del documento, la cuál viene dada por:

$$combined\_rel\_doi_p = \gamma \cdot combined\_rel_p + (1 - \gamma) \cdot rep_p \,, \tag{3.6}$$

donde $combined\_rel_p$ es la relevancia combinada del lexicón y la red neuronal para el artcículo y rep_p es su reputación. El parámetro $\gamma \in (0, 1)$ modula, su importancia relativa.

En caso de que el DOI no esté disponible, se devuelve el resultado de la Ecuación 3.5 como medida de relevancia final.

Capítulo 4

Experimentos y resultados

4.1. Planificación temporal

En la Figura 4.1 se puede ver un diagrama de Gantt[21] que refleja el tiempo empleado en cada una de las fases del proyecto.

4.2. Experimentos

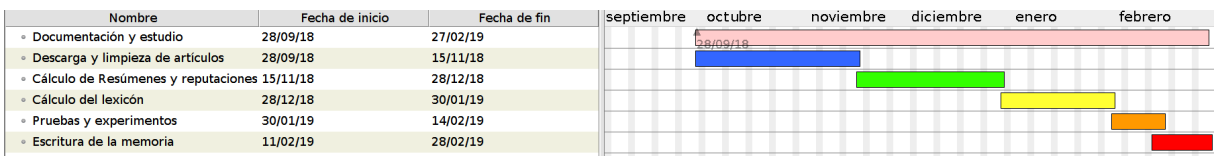

Figura 4.1: Diagrama de Gantt del desarrollo

Capítulo 5

Conclusiones

Bibliografía

[1] James Pustejovsky. The generative lexicon. *Computational linguistics*, 17(4):409–441, 1991.

[2] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2539–2544, 2015.

[3] Panos Vassiliadis. A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining (IJDWM)*, 5(3):1–27, 2009.

[4] Ryan Mitchell. *Web Scraping with Python: Collecting More Data from the Modern Web*. .ºReilly Media, Inc.", 2018.

[5] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and Franois Yergeau. Extensible markup language (xml) 1.0, 2000.

[6] Douglas Crockford. The application/json media type for javascript object notation (json). Technical report, 2006.

[7] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[8] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. Variations of the similarity function of textrank for automated summarization. *CoRR*, abs/1602.03606, 2016.

[9] Alberto Fernández-Isabel, Juan Carlos Prieto, Felipe Ortega, Isaac Martín de Diego, Javier M Moguerza, José Mena, Sara Galindo, and Liana Napalkova. A unified knowledge compiler to provide support the scientific community. *Knowledge-Based Systems*, 161:157–171, 2018.

[10] Allen Institute for Artificial Intelligence and Semantic Scholar. Semantic Scholar API. `https://api.semanticscholar.org/`, 2018. [Online: accedido 20-Dic-2018].

[11] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

[12] Aravind Chandramouli. Domain-specific stopword removal from unstructured computer text using a neural network, August 9 2018. US Patent App. 15/426,958.

[13] Sonal Gupta. *Distantly Supervised Information Extraction Using Bootstrapped Patterns*. PhD thesis, Stanford University, 2015.

[14] Murugan Anandarajan, Chelsey Hill, and Thomas Nolan. Term-document representation. In *Practical Text Analytics*, pages 61–73. Springer, 2019.

[15] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142, 2003.

[16] Merriam Webster. Merriam-webster medical dictionary. *Available from: Merriam-Webster and https://en. wikipedia. org/wiki/Iatrogenesis*, 2017.

[17] Lee Averell and Andrew Heathcote. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology*, 55(1):25–35, 2011.

[18] Krishna Bhavsar, Naresh Kumar, and Pratap Dangeti. *Natural Language Processing with Python Cookbook: Over 60 recipes to implement text analytics solutions using deep learning principles*. Packt Publishing Ltd, 2017.

[19] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[20] Dan O'Sullivan. *Wikipedia: a new community of practice?* Routledge, 2016.

[21] James M Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2):430–437, 2003.