Procesamiento de Lenguaje Natural para la Búsqueda Semántica en Repositorios Académicos

Heider Delgado[1], Franky Uzcátegui[1]
heider2112@gmail.com, fuzcateguip.ucv@gmail.com

[1] Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela

Resumen: La búsqueda semántica es un tipo de búsqueda de nueva generación que utiliza inteligencia artificial (IA) para obtener resultados de mayor relevancia para el usuario. Este artículo muestras las características relevantes del diseño y desarrollo de una solución que utiliza herramientas de lenguaje natural para la búsqueda semántica en repositorios académicos, se utilizó el framework de Haystack para elaborar el buscador. El caso de uso se centra en el desarrollo de un buscador semántico para los Trabajos Especiales de Grado (TEG) del repositorio institucional Saber UCV, se utilizó la metodología de garaje de IBM, más concretamente el ciclo de vida modelOps para el desarrollo, prueba de modelos y aplicación web. Se evaluaron 8211 TEG, clasificándolos por carrera y tutor usando los algoritmos de las librerías de spaCy y Flair, obteniendo precisiones mayores de 90 %, logrando también un buscador semántico con alta exhaustividad.

Palabras Clave: Inteligencia Artificial; Procesamiento de Lenguaje Natural; Análisis Semántico; Motor de Búsqueda Semántico; Modelos de Lenguaje.

1. Introducción

Los motores de búsqueda son definidos como un sistema informático que busca archivos almacenados en servidores web, varios de estos sistemas pueden ser buscadores de páginas web, como es el caso más conocido de Google [1]. Estos sistemas también pueden ser usados para encontrar archivos académicos dado un repositorio que tenga estos datos.

La búsqueda en repositorios académicos no es algo nuevo, pero no es algo trivial tampoco. Los artículos científicos pueden perderse muy fácilmente en motores de búsqueda web y requieren de motores especializados como son el caso de Google Scholar o arxiv.org, también señala la importancia de la web semántica y como los motores de búsqueda de artículos científicos parecieran distanciarse de esta nueva tecnología web [2].

Sin embargo, recientemente con la llegada de modelos de lenguaje como BERT (*Bidirectional Encoder Representations from Transformers*) demostraron que la semántica se pueda utilizar de otra forma. La búsqueda institucional, así como la búsqueda empresarial están bastante relacionadas en el sentido de que es difícil encontrar información relevante en sus repositorios, con BERT se propone que es posible ir más allá que de la búsqueda de solo palabras clave [3].

White, realizó una investigación a 20 empresas y todos los ejecutivos le hablaban de como deseaban tener un sistema mejor de búsqueda empresarial que fuera parecido a Google. El problema fundamental que encontró es que las organizaciones no ven a la información como un activo y no tienen idea de la cantidad de información que tienen [4].

Varias empresas especializadas en búsqueda empresarial han desaparecido o se han fusionado con otras empresas debido al alza de software libre que ha hecho posible que tecnologías que antes no existían, existan ahora [4][5].

Borges et al. [6] propusieron crear un sistema de NLS (*Natural Language Search*) que utilizando lenguaje natural se podían hacer consultas sobre bases de datos de clientes en un sistema de CRM (*Customer Relationship Management*), con este trabajo demostraron que era posible satisfacer mejor al cliente utilizando lenguaje natural . Su trabajo estaba inspirado en la investigación de [7] para realizar consultas SQL utilizando lenguaje natural y de [8] en Lenguajes Naturales Simplificados. El trabajo de Borges es un caso de uso sofisticado de implementación múltiple de lenguaje natural y palabras clave para la búsqueda de documentos.

La **búsqueda semántica** es una búsqueda que utiliza inteligencia artificial que no se basa en coincidencias de cadenas de caracteres exactas o variante de estas, sino en la intención de la consulta y la similaridad que tiene estas con un fragmento de texto. Al hacer esto se logra encontrar información que es más relevante mucho más rápido y con mayor precisión que si se utilizaran métodos tradicionales estadísticos [9].

Los TEG del repositorio académico saber.ucv no se encuentran clasificados por carrera, facultad, ni por tutor, realizar consultas dentro de este repositorio puede ser frustrante. En el desarrollo de esta solución se implementó un sistema que fue capaz de clasificar los TEG por carrera y por tutor. Después de ser clasificados los TEG de pregrado se utilizó el framework de Haystack para desarrollar un buscador semántico y se desarrolló una interfaz web para

acceder a este buscador.

2. Metodología

Para el desarrollo de esta solución se utilizó una adaptación del método de garage de IBM [10], el método de garage de IBM es una metodología única desarrollada por IBM, que incorpora conceptos ágiles como construir un producto mínimo viable. Más concretamente se implementó el ciclo de vida ModelOps un diagrama de este se encuentra Figura 1 [11].

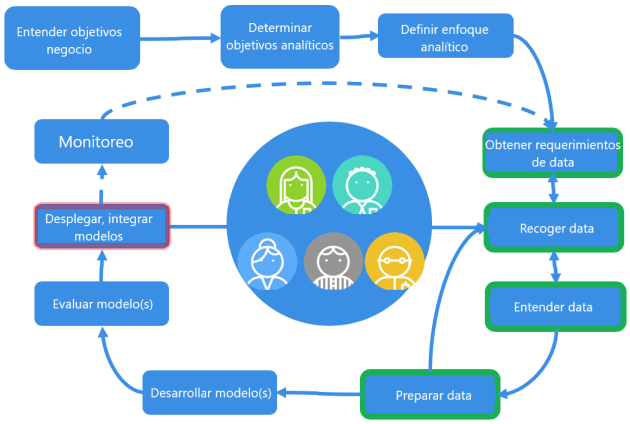

Figura 1: Ciclo de Vida ModelOps

2.1. *Clasificación de los TEG*

En los primeros pasos de este ciclo de vida se establece la creación de una solución que sea capaz de clasificar TEG del repositorio académico saber.ucv por carrera y por tutor. Para saber que también se han clasificado los TEG se utiliza la ecuación (1) para muestras infinitas y ecuación (2) para muestras finitas, donde Z es el Z-score, ϵ es el margen de error, $\hat{p}$ es la proporción de la población y N es la cantidad de muestras [12].

$$n = \frac{Z^2 \times \hat{P}(1-\hat{P})}{\epsilon^2} \tag{1}$$

$$n' = \frac{n}{1 + \frac{Z^2 \times \hat{P}(1-\hat{P})}{\epsilon^2 N}} \tag{2}$$

Los datos de los TEG: año, autor, título, peso, URL y URI, eran suministrados directamente de la página de saber.ucv. Estos fueron obtenidos usando la librería de Python para webscraping BeautifulSoup y de solicitudes HTTP Request. Luego de ser descargados estos datos se guardaron en una tabla usando como total **8211 TEG**. Los TEG fueron descargados usando el URL donde estaba contenido el pdf, el peso total de todos los TEG era de **44.79 GB**, se realizaron procesos de validación para comprobar que los TEG se habían descargado, luego se realizó un proceso en determinar si los TEG eran archivos escaneados o texto, se siguieron los siguientes criterios:

- La primera hoja es imagen o no.
- La segunda hoja es imagen o no.
- La hoja del medio más uno es imagen o no.
- La hoja del final -20 es imagen o no (esto es por la bibliografía).
- Una hoja aleatoria que no sea la primera ni la segunda es imagen o no.

Si el archivo pdf cumple con **4** de las **5** condiciones se considera que es imagen, de lo contrario el archivo es clasificado como texto.

Con los TEG clasificados por texto se procede a clasificarlos por escuela, para esto se tenía una lista de todas las escuelas de la Universidad Central de Venezuela, consistente de 11 facultades y 43 escuelas. Normalmente el nombre de la escuela a la cual pertenece un TEG se encuentra en la portada, utilizando la librería de Python para NLP spaCy, donde si una oración de los primeros o últimos caracteres de un TEG coincidía con alguna de las escuelas en primera instancia, entonces el TEG estaría clasificado por esa escuela y facultad de la lista.

Para la clasificación como tutor se utilizó la librería de Python para NLP Flair que utiliza el modelo de FLERT (*Document-Level Features for Named Entity Recognition*). Este modelo de lenguaje tiene un peso de **2 GB** y una medida F1 de **94,36** (en el set de datos CoNLL-03) [13]. Se utilizó para extraer entidades, más concretamente nombres propios. Para identificar a un tutor en un TEG se siguieron estos pasos:

1. Se extraen los nombres de las primeras y últimas páginas de un TEG, para crear una lista de nombres candidatos ordenados por la forma en que aparecen.
2. Estos nombres candidatos luego son filtrados, si son el nombre del autor entonces no es el tutor.
3. Si es un nombre que está contenido en el título del TEG (ejm: Estudio en el Hospital "Jose Maria Vargas") es descartado este nombre de la lista.
4. Si aparece la palabra tutor en cercanía a el nombre entonces era tomado como relevante para ser el tutor.
5. El nombre que estuviera al tope de esta lista era clasificado como tutor de ese TEG.

2.2. *Búsqueda Semántica*

Para realizar un buscador semántico se hizo uso del framework de Haystack [14], este framework permite la creación de sistemas de búsqueda que implementan lenguaje natural de principio a fin. Este framework se basa en responder preguntas de forma extractiva, donde se tienen 3 partes Figura 2 [14]: El almacén de documentos donde se encuentran los TEG, el recuperador y el lector. En el desarrollo de la solución se omitió el lector ya que para hacer búsqueda semántica sólo se necesita el recuperador y si se quiere responder preguntas se utiliza el recuperador y luego el lector.

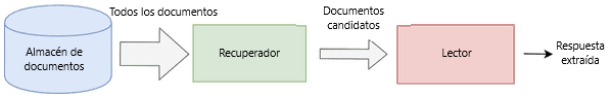

Figura 2: Framework de Haystack

Preparar la data es un paso fundamental para desarrollar una aplicación de inteligencia artificial. En este caso el modelo de lenguaje es BETO (variación en español de BERT) y será llamado IIC (Instituto de Ingeniería del Conocimiento una organización de desarrollo de software de España) [15]. Este modelo fue pre entrenado utilizando la técnica de DPR (Dense Passage Retrieval) que consiste en utilizar dos modelo de lenguaje en el recuperador de pasajes, basado en el codificador de la pregunta y codificador del pasaje que luego utilizan una medida de similitud para dar el pasaje mejor ranqueado que responda a una pregunta en lenguaje natural [16].

Estos modelos de lenguaje necesitan pasajes que consistan de pocas palabras, por lo tanto para entrenar este modelo utilizando aprendizaje por transferencia se etiquetaron 151 pares de pasajes y preguntas de TEG de computación con pasajes consistentes de 200 palabras, Con 101 pares para entrenamiento, 25 pares de evaluación y 25 pares de prueba. Luego se entrenaron 10 modelos con hiperparametros diferentes, la forma de evaluarlos es a través de la **precisión**: en este contexto la precisión se refiere a si el primer documentos es el correcto de los recuperados (el que responde la pregunta), la **exhaustividad**: se refiere a que de un set de elementos recuperados el documento que responde a la pregunta está en ese set, y por último la **medida F1**: una relación armónica entre precisión y exhaustividad.

Se escogió el modelo con las mejores métricas y este luego fue usado en el framework de Haystack para subir los pasajes de los TEG en ElasticSearch. El proceso de subida de pasajes de archivos es intensivo en tiempo para este ejemplo subir 90 TEG tomó aproximadamente 3 horas con GPU 3070ti y CPU i7- 12700K y 48GB de memoria.

Se desarrolló una aplicación web para mostrar los resultados de las consultas, Haystack posee un filtro por metadata y con esto se permite filtrar los TEG en las consultas por carrera, tutor, autor y fecha de publicación.

2.3. Arquitectura

Utilizando la librería Request de Python a través de una petición HTTP se puede obtener el código HTML de saber.ucv, que luego pasa por un proceso de webscrapping donde se obtienen los datos de título, URI de TEG y URL del pdf.

Se realiza el proceso de validación de existencia, por imagen, texto utilizando apache Tika, y luego la clasificación de los TEG utilizando spaCy y Flair. Se obtienen los modelos de lenguaje a través del portal Huggingface, los TEG son

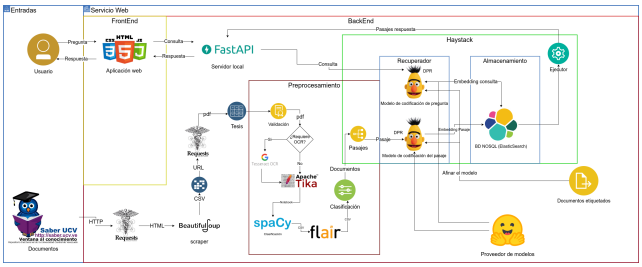

Figura 3: Arquitectura de la Solución

subidos a ElasticSearch utilizando el framework de Haystack. Para acceder a estos TEG se utiliza una interfaz web y como servidor se utiliza la librería de Python FastAPI. La Figura 3 contiene un diagrama de la arquitectura.

3. Resultados

Utilizando las fórmulas de (1) y (2) evaluando manualmente 365 TEG, se obtuvo el resultado de que este algoritmo de clasificación por carrera tiene **100 %** de precisión, con **95 %** de confianza y **5 %** de margen de error dado un muestreo aleatorio. El número N de muestras es 7103 como se muestra en la Figura 5. En la Figura 4 se muestra la cantidad de TEG por escuela de la Facultad de Ciencias, siendo la Escuela de Química la que posee la mayor cantidad de TEG con 628, seguida por la Escuela de Computación con 537.

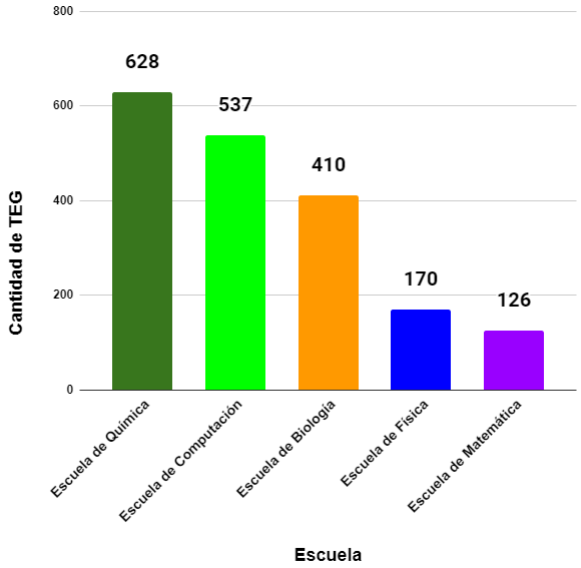

Figura 4: Cantidad de TEG en la Facultad de Ciencias

Utilizando las fórmulas de (1) y (2) y evaluando manualmente 365 TEG, se obtuvo el resultado de que este algoritmo de clasificación por tutor tiene **90,68 %** de precisión, con **95 %** de confianza y **5 %** de margen de error dado un muestreo aleatorio. El número N de muestras es **7078** como se describe en la Figura 5.

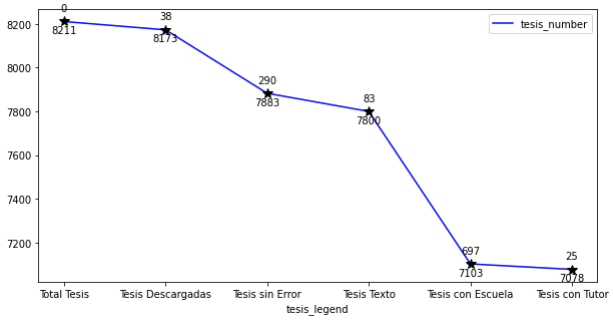

Figura 5: Reducción de TEG

En la Figura 5 se muestra cómo a medida que se realizan los diferentes pasos de preprocesamiento se reduce la cantidad de TEG clasificados. El número arriba de las estrellas es el número de reducción con respecto al paso anterior y el número de abajo representa la cantidad total de TEG en ese paso. El primer paso fueron los TEG totales a los descargados, el segundo paso es de los TEG descargados los TEG que no tienen errores para abrir, el tercer paso los TEG con texto, el cuarto paso los TEG con escuela, y el último paso los TEG con tutor, el total son 6 dataframes diferentes. Esto indica una reducción de 1133 TEG (**13.79 %**) del primero al último paso.

El mejor modelo entrenado con aprendizaje por transferencia arrojó los mismos resultados que el modelo base de IIC, con **0.517** de precisión media promedio en el set de prueba y **0.539** en el set de evaluación; con **0.84** de exhaustividad en el set de prueba y **0.833** en el set de evaluación. Comparando con el modelo estadístico BM25 [17] de ElasticSearch que tuvo los resultados de **0.469**, **0.537**, de precisión media promedio en el set de prueba y evaluación respectivamente; y **0.68**, **0.75** de exhaustividad en el set de prueba y evaluación respectivamente. Esta comparación se puede observar en la Figura 6, estos resultados nos muestran que es mejor utilizar un modelo que utilice DPR que un modelo estadístico.

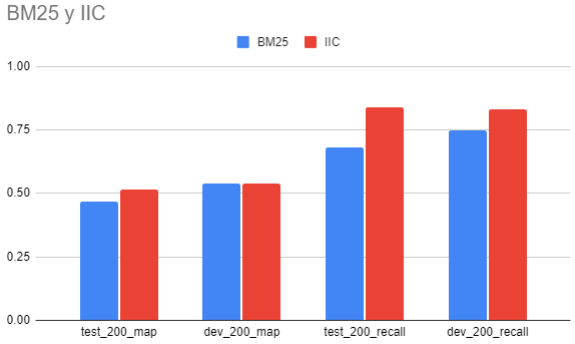

Figura 6: Comparación entre BM25 y IIC

4. Conclusión

El uso de la metodología de garage de IBM y el ciclo de vida de ModelOps fue una innovación única de este trabajo en Venezuela y su uso aplicativo en repositorios académicos. Se comprobó que al utilizar modelos de lenguaje se pueden obtener mejores resultados que utilizando métodos estadísticos, se creó una solución que tiene un filtro por carrera, tutor, autor y fecha que puede ser aplicado al repositorio académico saber.ucv y de esta forma tener un buscador más preciso y exhaustivo. Se debe tener en consideración el tiempo de subida de archivos en un almacén de datos ya que puede tomar mucho tiempo.

Referencias

[1] J. Tramullas Saz y M. D. Olvera Lobo, *Recuperación de la Información en Internet*, Spanish. 1ª edición, Madrid: Ra-Ma, 2001, OCLC: 49319083, ISBN: 978-84-7897-458-0

[2] L. Codina, *Motores de Búsqueda de Información Científica y Académica*, Hipertext.net, no. 5, 2007, Publisher: Sección Científica de Ciencias de la Documentación. https://arxiu-web.upf.edu/hipertextnet/numero-5/motores-busqueda.html

[3] J. Devlin, M. Chang, K. Lee y K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* CoRR, vol. abs/1810.04805, 2018. arXiv: 1810.04805. http://arxiv.org/abs/1810.04805.

[4] S. White, *Enterprise Search: Enhancing Business Performance*, Second edition. Boston: O'Reilly, 2016, OCLC: ocn921989928, ISBN: 978-1-4919-1553-0.

[5] P. H. Cleverley y S. Burnett, *Enterprise Search: A State of the Art*, Business Information Review, vol. 36, no. 2, pp. 60-69, Junio 2019, ISSN: 0266-3821, 1741-6450. DOI: 10.1177/0266382119851880. http://journals.sagepub.com/doi/10.1177/0266382119851880.

[6] F. Borges, G. Balikas, M. Brette et al., *Query Understanding for Natural Language Enterprise Search*, arXiv:2012.06238, 2020. https://arxiv.org/abs/2012.06238

[7] I. Androutsopoulos, G. D. Ritchie y P. Thanisch, *Natural Language Interfaces to Databases an Introduction*, Natural Language Engineering, vol. 1, no. 1, pp. 29-81, 1995, Publisher: Cambridge University Press

[8] R. Schwitter, *Controlled Natural Languages for Knowledge Representation* Coling 2010: Posters, 2010, pp. 1113-1121

[9] H. Bast, B. Buchhold y E. Haussmann, *Semantic Search on Text and Knowledge Bases*, Foundations and Trends® in Information Retrieval, vol. 10, no. 1, pp. 119-271, 2016, ISSN: 1554-0669, 1554-0677. DOI: 10.1561/1500000032. http://www.nowpublishers.com/article/Details/INR-032

[10] C. IBM, *Infuse IBM Artificial Intelligence Field Guide*, 2020. https://www.ibm.com/cloud/architecture/files/ai-analytics-fg.pdf

[11] C. IBM, *Method for Developing Intelligent Applications - IBM Cloud Architecture Center*, 2020. https://www.ibm.com/cloud/architecture/architecture/practices/method-for-developing-intelligent-apps/

[12] L. Naing, T. Winn y B. N. Rusli, *Practical Issues and Calculating the Sample Size for Prevalence Studies*, Archives of Orofacial Sciences, vol. 1, pp. 9-14, 2006, Publisher: School of Dental Sciences, Universiti Sains Malaysia.

[13] S. Schweter y A. Akbik, *FLERT: Document-Level Features for Named Entity Recognition*, 2020. arXiv: 2011.06993 [cs.CL].

[14] Haystack *Docs*, https://web.archive.org/web/20211020171700/https://haystack.deepset.ai/docs/intromd

[15] A. V. Serrano, Alba, D. Betancur y G. G. Subies, *IIC/dpr-spanish-passage_encoder-allqa-base*. Hugging Face, Marzo 2022. https://huggingface.co/IIC/dpr-spanish-passage_encoder-allqa-base

[16] V. Karpukhin, B. Oğuz, S. Min et al., *Dense Passage Retrieval for Open-Domain Question Answering*, 2020. arXiv: 2004.04906 [cs.CL]

[17] Shane Connelly, *Practical BM25 - Part 2: The BM25 Algorithm and its Variables*, Abril 2018. https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables