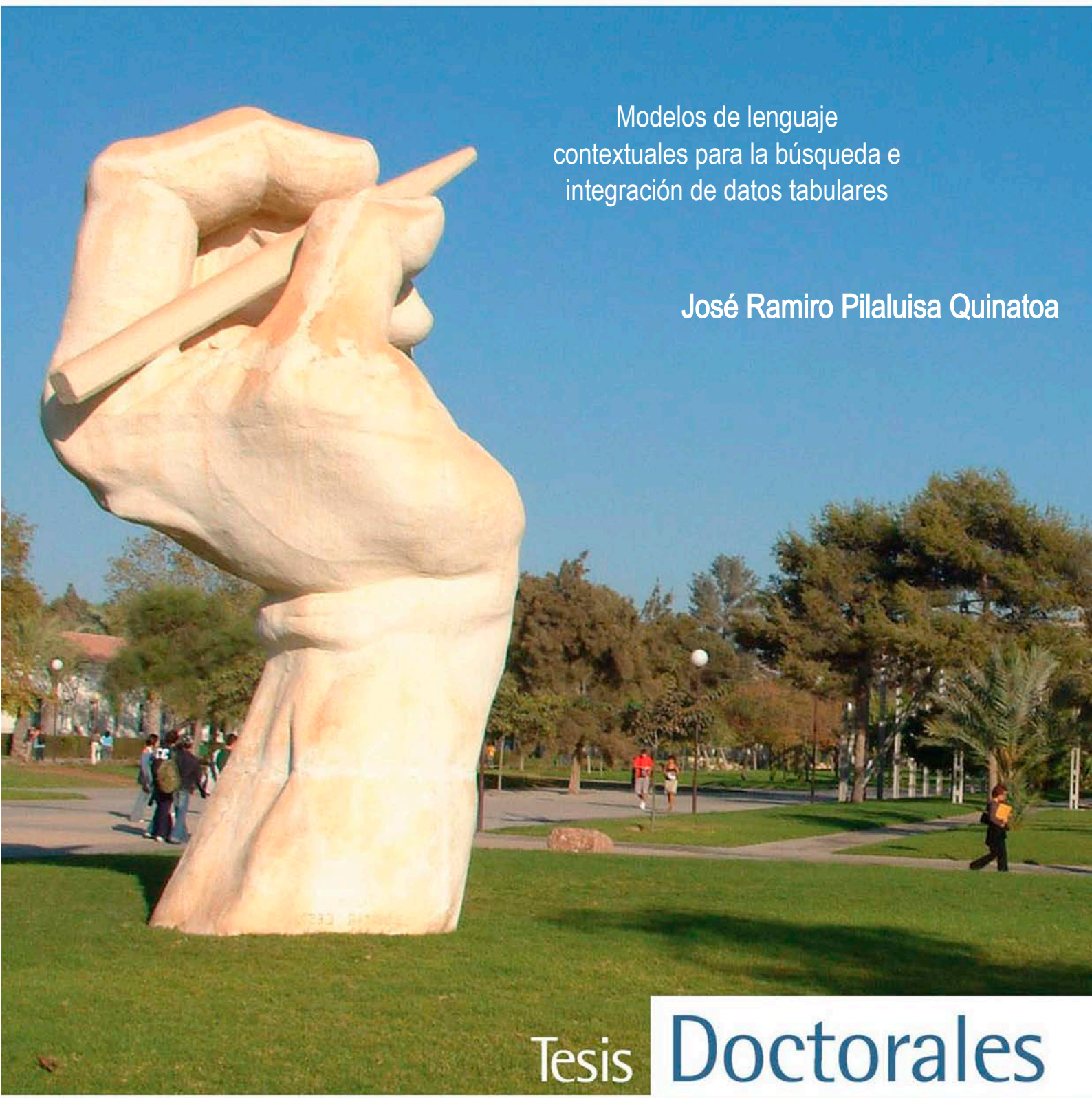
Universitat d'Alacant
Universidad de Alicante



Modelos de lenguaje contextuales para la búsqueda e integración de datos tabulares

José Ramiro Pilaluisa Quinatoa

Tesis Doctorales
UNIVERSIDAD de ALICANTE

Unitat de Digitalització UA
Unidad de Digitalización UA

Universitat d'Alacant
Universidad de Alicante

Departamento de Lenguajes y Sistemas Informáticos
Escuela Politécnica Superior

Modelos de lenguaje contextuales para la búsqueda e integración de datos tabulares

José Ramiro Pilaluisa Quinatoa

Tesis presentada para aspirar al grado de
Doctor por la Universidad de Alicante
Doctorado en Informática

Dirigida por:
Dr. David Tomás Díaz

Alicante, diciembre de 2022

Agradecimientos

En estas líneas me gustaría agradecer a todas aquellas personas que me han apoyado durante este largo camino.

Quiero dedicar este trabajo a Dios, a mi madre que con un gesto, con una palabra, siempre nos ha hecho saber que cuento con ella en todo momento y a mi esposa quien me apoya en cada proyecto que emprendo, por el apoyo que siempre me ha brindado, además de su comprensión y cariño diario.

Un agradecimiento muy especial a mi tutor y director de tesis David, por su apoyo incondicional que desde el principio fueron encaminándome en esta investigación.

Además también, a todas las personas que de una manera u otra, han influido en que continuase en este camino y a los que no puedo agradecerles lo suficiente su apoyo y confianza puesta en mí.

Resumen

Esta tesis propone una aproximación para la búsqueda e integración de datos en formato tabular. La novedad de la propuesta radica en el uso de modelos de lenguaje contextuales. Estos modelos han revolucionado el campo del procesamiento del lenguaje natural (PLN) en los últimos años. Sin embargo, son pocas las aproximaciones que han utilizado estos modelos para trabajar con datos estructurados como son las tablas. Si bien existe alguna aproximación para la tarea de búsqueda de tablas, no existen en la actualidad aproximaciones que usen estos modelos en todo el proceso de búsqueda e integración a nivel de unión y combinación de datos.

En este trabajo se hace una propuesta de adaptación de estos modelos de lenguaje, originalmente usados sobre datos no estructurados, para ser aplicados sobre datos estructurados. Durante el proceso se evaluará la efectividad de diferentes modelos existentes y se ajustarán sus parámetros de entrada para determinar la configuración más efectiva en la tarea. Además, se contrastarán los modelos contextuales con otros no contextuales, analizando el papel que tiene el contexto en el rendimiento del sistema.

El trabajo incluye también un estudio para la mejora del rendimiento de estos sistemas mediante la eliminación de contenido de las tablas. Para ello, se estudia cómo reducir el número de filas de las tablas afecta a la representación vectorial (*word embedding*) generada por el modelo de lenguaje. De esta manera se busca determinar la posibilidad de reducir tablas de gran tamaño sin perder representatividad en el espacio semántico que genera el modelo.

Por último, la tesis concluye haciendo una propuesta de anotación de datos tabulares para conseguir un conjunto de datos que permita entrenar y evaluar mejor este tipo de sistemas basados en técnicas de aprendizaje automático. En la actualidad no existen conjuntos de datos que resulten desafiantes y variados para la tarea de integración, especialmente en el caso de la operación de combinación (*join*). Se incluye un estudio piloto de anotación en el que se desarrolla un corpus inicial de tablas para la tarea de búsqueda e integración de datos tabulares.

Abstract

This thesis proposes an approach for searching and integrating data in tabular format. The novelty of the proposal lies in the use of contextual language models. These models have revolutionised the field of natural language processing (NLP) in recent years. However, few approaches have used these models to work with structured data such as tables. Although some approaches exist for the task of table retrieval, there are currently no approaches that use these models in the whole process of search and integration with union and join operators.

In this paper a proposal is made to adapt these language models, originally used on unstructured data, to be applied on structured data. In the process, the effectiveness of different existing models will be evaluated and their input parameters will be adjusted to determine the most effective configuration for the task. In addition, contextual models will be contrasted with non-contextual models, analysing the role of context in the performance of the system.

The work also includes a study of how to improve the performance of these systems by removing content from the tables. To this end, we study how reducing the number of rows in the tables affects the vector representation (*word embedding*) generated by the language model. In this way, we want to determine the possibility of reducing large tables without losing representativeness in the semantic space generated by the language model.

Finally, the thesis concludes with a proposal for the annotation of tabular data in order to obtain a dataset that allows better training and evaluation of this type of systems based on machine learning techniques. At present, there are no challenging and varied datasets for the integration task, especially in the case of the join operation. A pilot study of annotation is included, in which an initial corpus of tables is developed for the task of searching and integrating tabular data.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

La ciencia de datos tiene como objetivo llegar de los datos sin procesar al conocimiento a través de diferentes pasos, como la limpieza, la integración, la discusión, el análisis y la visualización. Se estima que el proceso de limpieza e integración de datos comprende cerca del 80 % del trabajo de un científico de datos (Abadi et al., 2022) debido a su naturaleza compleja (Halevy et al., 2006). Estas tareas implican varios tipos de transformaciones, como la normalización, que utilizan operaciones como la unión y la combinación para ofrecer una vista unificada de un conjunto de datos heterogéneos de diferentes fuentes. En este escenario es esencial que los científicos de datos tengan herramientas para respaldar el proceso de integración (Konstantinou et al., 2019).

La integración de datos es incluso más compleja dentro de un escenario de datos abiertos, ya que los datos a consumir no se conocen a priori y se requiere más esfuerzo para recuperarlos y comprenderlos (Miller, 2018). La mayoría de datos abiertos disponibles en Internet se publican en portales en forma de datos tabulares, usando formatos como CSV (Neumaier et al., 2016). Los datos abiertos tabulares son una forma común de almacenar, clasificar y difundir información, aunque en ocasiones es difícil extraer y manipular automáticamente su contenido (Rastan et al., 2019). El procesamiento automático de datos tabulares involucra disciplinas como ciencia de la información (Khodabakhsh and Bagheri, 2021), los datos enlazados (*linked data*) (Bizer and Schultz, 2009) y el procesamiento del lenguaje natural (PLN) (Herzig et al., 2020; Yin et al., 2020).

El uso de datos abiertos tabulares desconocidos es una tarea que requiere mucho tiempo para los científicos de datos, que deben comprender adecuadamente los datos (por ejemplo, descubrir sus relaciones) antes de recuperar otros datos que son potencialmente útiles para la integración. Esta tesis presenta una aproximación para ayudar a los científicos de datos a encontrar relaciones entre los datos con el fin de recuperar e integrar datos tabulares relevantes.

El método propuesto consta de dos pasos. En primer lugar se busca, dada una tabla inicial, recuperar conjuntos de datos tabulares[1] que tengan columnas similares y, por lo tanto, tengan más probabilidad de ser susceptibles de integración con la tabla inicial. Una vez que se obtiene este conjunto de tablas similares, el segundo objetivo es determinar qué columnas específicas deben estar involucradas en el proceso de integración final.

El presente estudio se centra en los operadores de unión (*union*) y combinación (*join*), ya que son especialmente relevantes para la integración de datos, facilitando su consumo desde fuentes dispares (Miller, 2018). Es importante destacar que trabajar con estos dos operadores proporciona la base para incluir otros ampliamente utilizados en la integración de datos, como el filtrado y la clasificación.

Para llevar a cabo los procesos de búsqueda e integración se ha definido una medida de similitud semántica entre datos tabulares. Esta medida aprovecha los modelos de lenguaje contextuales para obtener incrustaciones de palabras (*word embeddings*) (Peters et al., 2018b) y calcular la similitud entre tablas, utilizando tanto el nombre de las columnas como el contenido de las celdas.

El uso de *word embeddings* es una técnica que permite transformar palabras o frases en vectores de números reales, capturando regularidades semánticas en el espacio vectorial. De esta manera, al comparar dos términos los *word embeddings* superan los problemas de los enfoques léxicos basados en la similitud de cadenas: palabras como "ciudad" y "ubicación" podrían considerarse muy diferentes si las comparamos a nivel léxico. Sin embargo, en el espacio vectorial generado por los *word embeddings* estos dos términos pueden estar muy próximos entre sí y estrechamente relacionados, considerándose de esta manera como muy similares.

Los *word embeddings* tradicionales, como los generados por el modelo Word2vec (Mikolov et al., 2013a), proporcionan la misma representación vectorial para una palabra independientemente del contexto en el que aparezca. A diferencia de estos *word embeddings* estáticos (no contextuales), los recientes modelos contextuales son capaces de capturar todos los posibles significados de una palabra, proporcionando diferentes representaciones vectoriales según el contexto en el que tienen lugar. El uso de *word embeddings* contextuales es omnipresente hoy en día en múltiples tareas de PLN, como la recuperación de información (Dai and Callan, 2019), la búsqueda de respuestas (Qu et al., 2019), los resúmenes automáticos (Liu and Lapata, 2019) y detección del discurso de odio (Plaza-del Arco et al., 2021). En estas áreas se han conseguido mejoras significativas. Recientemente, estos modelos se han aplicado también a la recuperación de tablas y, en menor medida, a la integración de datos tabulares.

[1] La propuesta puede extenderse a cualquier tipo de formato tabular, aunque en los experimentos realizados se trabajó únicamente con datos en formato CSV.

Esta tesis presenta el primer intento de utilizar modelos de lenguaje contextuales a lo largo de todo el proceso de búsqueda e integración de tablas, incluyendo de manera conjunta las operaciones de unión y combinación. El uso de estos modelos va a permitir superar las limitaciones de los sistemas que se basan únicamente en la similitud de cadenas para recuperar los datos, permitiendo aumentar la cobertura de estos sistemas y evitar dejar fuera datos que puedan ser de interés.

1.1. Motivación

Como se ha mencionado al inicio de esta tesis, en la actualidad la mayoría de datos abiertos existentes en Internet se hayan en forma de datos tabulares. En estas circunstancias, resulta difícil para el usuario de estos datos el conseguir identificar conjuntos que sean de su interés y, más aún, conseguir datos que puedan extender a los que ya poseen mediante operaciones de integración.

Hay varias razones por las que hoy en día es importante contar con un buen sistema de búsqueda e integración de datos tabulares. La cantidad de datos que generan y recogen las organizaciones está aumentando rápidamente, y cada vez es más difícil gestionar y analizar estos datos sin el uso de herramientas y sistemas especializados. Disponer de un buen sistema para recuperar e integrar los datos tabulares permite a las organizaciones analizar de forma eficiente y precisa grandes cantidades de datos, lo que es esencial para tomar decisiones informadas y mejorar las operaciones.

Un buen sistema de búsqueda e integración de datos tabulares puede ayudar a garantizar la exactitud y fiabilidad de la información que se utiliza, ya que permite a los usuarios acceder rápida y fácilmente a los datos que necesitan y combinarlos de forma que proporcionen información útil. Esto es especialmente importante en el mundo actual, en el que la capacidad de tomar decisiones oportunas e informadas es a menudo fundamental para el éxito.

Más aún, disponer de un buen sistema para buscar e integrar datos tabulares permite a las compañías combinar fácilmente datos de múltiples fuentes, lo que es esencial para obtener una visión global de los datos, haciendo predicciones y previsiones precisas. Esto es de gran importancia para las compañías, impulsadas por los datos, ya que estas deben ser capaces de analizar rápidamente y con precisión los datos procedentes de diversas fuentes para tomar decisiones informadas y obtener una ventaja competitiva.

1.2. Objetivos

El principal objetivo de esta tesis es el desarrollo de un sistema para la búsqueda e integración de datos tabulares mediante el uso de

modelos de lenguaje contextuales en todas sus fases. Estos modelos se usan habitualmente en tareas de PLN, pero en la actualidad están demostrando también su potencial en el manejo de datos estructurados como son las tablas.

Para conseguir este objetivo principal se deben alcanzar una serie de objetivos secundarios que se listan a continuación:

- Definir una medida de similitud entre tablas basada en *word embeddings* obtenidos mediante modelos contextuales
- Crear un sistema de búsqueda de información estructurada (tablas) basado en la medida de similitud identificada
- Evaluar diferentes modelos de lenguaje para determinar el mejor para la consecución de nuestros objetivos
- Comparar el rendimiento de modelos contextuales y no contextuales en esta tarea
- Determinar el impacto que tiene el contexto en el desempeño de los modelos contextuales
- Evaluar la aportación que los nombres de las tablas y el contenido de las celdas tienen a la hora de identificar tablas similares
- Establecer un mecanismo basado en la medida de similitud especificada para determinar si dos tablas se pueden unir o combinar
- Estudiar la posibilidad de aumentar el rendimiento de estos sistemas reduciendo la información de la tabla sin alterar el *word embedding* que la representa
- Definir una metodología para la creación de conjuntos de datos tabulares que sirvan para el entrenamiento y evaluación de sistemas basados en aprendizaje automático para la integración de tablas
- Crear un conjunto de datos adecuado para la evaluación de este tipo de sistemas

1.3. Metodología

Lo primero que se planteó en este trabajo fue un análisis del área de la búsqueda e integración de datos tabulares. Mediante este estudio se identificó que, aunque existían algunos trabajos que usaban modelos de lenguaje no contextuales, el uso de modelos contextuales era todavía minoritario y no habían estudios profundos sobre el impacto del contexto en estas tareas.

A continuación se planteó una estrategia para poder obtener una representación en forma de *word embedding* de las tablas usando modelos de lenguaje. Estos modelos están originalmente pensados para su uso con texto libre en tareas de PLN, por lo que su aplicación a datos tabulares no era inmediata y había que plantear una aproximación para poder adaptarlos.

Una vez se pudieron obtener *word embeddings* a nivel de tabla, se planteó una medida de similitud que pudiera explotar esa representación vectorial. Esa medida es la que permite al sistema de búsqueda de tablas determinar, para una tabla inicial, cuáles son las tablas más similares a ésta. La medida de similitud definida se especializó en dos, dependiendo de si el objetivo de integración tras la recuperación de las tablas era unirlas o combinarlas con la tabla de consulta. Esta medida de similitud tiene tanto en cuenta el nombre de las columnas de la tabla como el contenido de sus celdas.

Tras definir estas medidas se evaluaron diferentes modelos de lenguaje, contextuales y no contextuales, con el objetivo de determinar aquellos que obtenían mejor rendimiento tanto en la tarea de recuperación como en la posterior integración. También se estudió el impacto de usar el nombre de las columnas y el contenido de las celdas en el rendimiento del sistema. Así mismo, se analizó el papel que jugaba el contexto en estos resultados.

Una vez configurado el sistema, se estudió la posibilidad de mejorar su rendimiento, ya que obtener la representación vectorial de una tabla en forma de *word embedding* puede ser una tarea computacionalmente costosa si la tabla tiene un número elevado de filas y columnas. Para ello se llevaron a cabo diversos experimentos de eliminación de contenido de las tablas para ver cómo afectaba esa eliminación a la representación vectorial. Es decir, si eliminar parte del contenido de la tabla no cambia sustancialmente el *word embedding* generado para ésta, podemos tener la seguridad de que el rendimiento del sistema no se va a ver alterado y conseguiremos aumentar su velocidad de ejecución al reducir el volumen de los datos.

Por último, durante el proceso de evaluación de los sistemas se vio que los datos tabulares existentes para evaluar este tipo de aproximaciones no eran óptimos. Su obtención se basaba en proyecciones y selecciones sobre unas pocas tablas originales, lo que llevaba a que el conjunto de datos final resultara bastante repetitivo y no todo lo desafiante que se podía esperar para poder ver la aportación de los modelos de lenguaje a la tarea. Por esta razón se planteó una metodología para la creación de un dataset de tablas que pudiera usarse para evaluar la búsqueda e integración de tablas mediante unión y combinación, así como para entrenar sitemas basados en aprendizaje automático en estas tareas. Esta metodología se aplicó en una primera prueba piloto en la que participaron 30 anotadores de manera colaborativa para etiquetar 500 pares de tablas.

1.4. Estructura de la tesis

El resto de la tesis se estructura de la siguiente manera:

- Capítulo 2: presenta una introducción a las tareas de búsqueda e integración de datos tabulares, dando el contexto necesario para ubicar al lector en ambas tareas. Se incluye también una descripción de los trabajos más significativos en el área.
- Capítulo 3: describe todos los conceptos ténicos relaciondos con los modelos de lenguaje y los *word embeddings*, introduciendo para ello las bases de la inteligencia artificial, el aprendizaje automático, las redes neuronales y el PLN.
- Capítulo 4: se muestra en detalle el sistema propuesto para la búsqueda e integración de datos tabulares, incluyendo las medidas de similitud y la evaluación llevada a cabo con los diferentes modelos.
- Capítulo 5: ofrece un estudio del impacto que tiene la eliminación de contenido de una tabla en su representación mediante *word embeddings*. El objetivo es medir hasta qué punto se puede reducir la información de la tabla sin alterar su representación vectorial, reduciendo de esta manera el coste computacional de generar su *word embedding*.
- Capítulo 6: se centra en describir la metodología de desarrollo de un conjunto de tablas para evaluar las tareas de búsqueda e integración, incluyendo la descripción de una experiencia piloto llevada a cabo para obtener una primera anotación de los datos.
- Capítulo 7: se resumen las conclusiones del trabajo, los objetivos alcanzados, las contribuciones realizadas y el trabajo futuro derivado de esta tesis.
- Capítulo 8: finalmente, se muestran las publicaciones realizadas en el ámbito científico fruto del trabajo de investigación desarrollado.

2

Estado de la cuestión

Este capítulo va a ofrecer una revisión del estado actual de las tecnologías de búsqueda e integración de datos tabulares, describiendo las tipologías de búsqueda que se pueden dar, las distintas operaciones para el aumento de tablas y los trabajos relacionados en el área.

Antiguamente la información estaba contenida en libros, enciclopedias y hojas que debíamos leer para extraer los datos que nos interesaban. Con la llegada de Internet este mismo contenido se encuentra en miles de documentos digitales que se crean día a día.

Las tablas son herramientas poderosas y populares para organizar y manipular datos, útiles en muchos escenarios de aplicación. Se pueden utilizar eficazmente para recopilar y organizar información de múltiples fuentes. Con la ayuda de operaciones adicionales, como ordenar, filtrar, combinar y unir, esta información se puede convertir en conocimiento y, en última instancia, se puede utilizar para apoyar la toma de decisiones.

Gracias a su conveniencia y utilidad, existen un gran número de tablas disponibles en Internet, lo cual representa un valioso recurso de conocimiento que ha sido un foco de investigación desde tiempo atrás (Zhang and Balog, 2020). La Web consta de una gran cantidad de documentos no estructurados, pero también contiene datos estructurados en forma de tablas HTML. Estas *tablas web* difieren de las tablas tradicionales (es decir, tablas en bases de datos relacionales y tablas creadas en programas de hojas de cálculo) en varios aspectos. Las tablas web están incrustadas en las páginas web. Hay mucha información contextual que se puede explotar, como el título de la página, la estructura de enlaces y el texto circundante. Las tablas web son muy heterogéneas en cuanto a calidad, organización y contenido.

2.1. Búsqueda de tablas

El objetivo de la recuperación o búsqueda de tablas (*ad hoc table retrieval*) es responder a una consulta de búsqueda con una lista ordenada de tablas que se consideran relevantes para esa consulta (Zhang and Balog, 2018). Es una tarea importante en sí misma, pero también se considera un paso fundamental en la posterior integración de los datos (Zhang and

Balog, 2020). La función de búsqueda de tablas está disponible en productos comerciales como Microsoft Power Query.[1]

Dependiendo de los datos de entrada del sistema de búsqueda, podemos tener dos aproximaciones. En la primera, el usuario utiliza un conjunto de palabras clave para realizar la consulta, a semejanza de las búsquedas que se realizan en los sistema de recuperación de información tradicionales como Google. En la segunda aproximación, la entrada proporcionada por el usuario es una tabla, siendo el objetivo del sistema el recuperar tablas similares a ella. Ambas aproximaciones se explican en los siguientes apartados.

2.1.1. Búsqueda basada en palabras clave

En este tipo de búsqueda, el usuario introduce un conjunto de palabras clave (*keywords*), como se haría en un buscador tradicional en la Web, con el objetivo de recuperar tablas relacionadas con esas palabras. Uno de los primeros trabajos en esta área es el de Cafarella et al. (2008), donde implementan la búsqueda de tablas mediante palabras clave sobre un motor de búsqueda web existente. Específicamente, lo que hacen es extraer las `n` primeras tablas de las páginas web devueltas. Un sistema similar llamado OCTOPUS (Cafarella et al., 2009) amplía el mismo método (denominado *SimpleRank*) con un mecanismo de reordenación (*SCPRank*) que considera las co-ocurrencias de atributos en las tablas.

A la hora de desarrollar aproximaciones para la recuperación de tablas, podemos crear una representación de la tabla y tratarla como si fuera un documento. Esta representación puede contener todo el texto incluido en la tabla o solo ciertos elementos de ella (por ejemplo, etiquetas de título o encabezados). Estos documentos pueden entonces usarse como entrada para algoritmos de recuperación de información tradicionales, usando esquemas de pesado como TF-IDF (Pimplikar and Sarawagi, 2012), que tiene en cuenta tanto la frecuencia de los términos en el documento (TF) como lo infrecuentes que resultan en el resto de documentos (IDF).

Otra forma de afrontar la recuperación de tablas consiste en emplear técnicas de aprendizaje automático supervisado utilizando diferentes características de las tablas. Tenemos tres tipologías principales de características:

- Las *características de consulta* incluyen la longitud de la consulta y las puntuaciones IDF de los términos de la consulta
- Las *características de la tabla* identifican sus dimensiones (número de filas y columnas), coherencia del esquema y, en el caso de tablas web, la conectividad de la página (ej. visualizaciones, enlaces de entrada y de salida) y la importancia de la tabla dentro de ella (Bhagavatula et al., 2013).

[1] https://powerquery.microsoft.com.

- Las *características de la tabla de consulta* capturan el grado de coincidencia entre la necesidad de información del usuario y las tablas, proporcionando puntuaciones de similitud entre la consulta y varios elementos de la tabla.

Los trabajos de Cafarella et al. (2008) y Bhagavatula et al. (2013) son dos ejemplos de este tipo de aproximación, donde se entrenaron diferentes clasificadores basados en regresión lineal sobre características extraídas de las tablas.

En el trabajo de Pimplikar and Sarawagi (2012), en lugar de confiar en una sola consulta de palabras clave como entrada, toman un conjunto q de columnas, cada una descrita por una serie de palabras clave $(Q_1 \cdots Q_q)$. Por ejemplo, Q_1 = "elemento químico", Q_2 = "número atómico" y Q_3 = "peso atómico". Esta consulta múltiple devolvería una tabla con q columnas como respuesta. Para conserguir esto, primero clasifican las tablas usando la unión de palabras clave. Posteriormente, cada columna de la tabla se etiqueta con la columna de consulta a la que se asigna. Finalmente, las columnas y filas relevantes se fusionan en una sola tabla, considerando los puntajes de relevancia a nivel de tabla y los puntajes de confianza a nivel de columna. Para decidir si dos filas son duplicadas entre sí emplean el método definido por Gupta and Sarawagi (2009).

En el trabajo de Zhang and Balog (2018) se busca la concordancia semántica entre las palabras clave de la consulta y las tablas. Primero representan consultas y tablas en múltiples espacios semánticos para luego introducir varias medidas de similitud sobre esas representaciones semánticas. Tanto las consultas como las tablas se representan extrayendo entidades, categorías, *word embeddings* y *graph embeddings*.

En el caso de (Zhang and Balog, 2017a), los autores consideran todas las combinaciones posibles de representación semántica y medidas de similitud, utilizándolas como características de entrada en un modelo de aprendizaje supervisado. Demuestran mejoras significativas y sustanciales sobre un *baseline* basado en funciones de última generación. Más recientemente, Zhang et al. (2019) utilizaron también *word embeddings* sobre un conjunto de tablas de Wikipedia para obtener resultados comparables a los anteriores.

2.1.2. Búsqueda basada en tablas

A la hora de buscar tablas no solo podemos usar un conjunto de *keywords* como entrada, sino que también se puede usar una tabla, en cuyo caso la tarea de devolver tablas relacionadas se denomina *búsqueda basada en tablas*. En esencia, esta tarea se reduce a calcular una puntuación de similitud entre la tabla de entrada y las tablas candidatas del conjunto de datos. La búsqueda por tabla se puede realizar con diferentes finalidades:

- Recuperar tablas similares y satisfacer las necesidades de información del usuario (Sarma et al., 2012; Limaye et al., 2010; Nguyen et al., 2015).

- Servir como paso intermedio para otras tareas, como el aumento de tablas que se describe en la Sección 2.2 (Ahmadov et al., 2015; Lehmberg et al., 2015; Nargesian et al., 2018; Yakout et al., 2012).

Una forma de abordar esta aproximación es utilizar ciertos elementos de la tabla como si fueran las palabras clave de la consulta y utilizar de esta manera los mismos métodos de recuperación que se usaban en el caso anterior. Por ejemplo, Ahmadov et al. (2015) utilizan entidades y encabezados (nombres de las columnas) de la tabla de entrada para construir dos consultas y recuperar una lista ordenada de tablas relevantes. Las dos listas devueltas se cruzan posteriormente para llegar a un conjunto de resultados único.

Otra forma de abordar la búsqueda basada en tablas es dividir las tablas en varios elementos (como el título de tabla, las entidades, nombres de columna y valores de las celdas) para luego calcular similitudes a nivel de cada uno de estos.

Vale la pena señalar que, en la mayoría de los trabajos realizados en este ámbito, la búsqueda de tablas no es el objetivo final sino un componente en una aplicación más grande. Por ejemplo, el motor de búsqueda de Lehmberg et al. (2015) pretende ampliar la tabla de entrada con atributos adicionales. Compara los nombres de columna de la tabla de entrada y las tablas candidatas. Primero filtran tablas que comparten al menos un nombre de columna con la tabla de entrada, utilizando la coincidencia exacta de términos. Luego, la puntuación de coincidencia de la tabla se calcula de dos formas. En la primera se construye una matriz de similitud de distancia de edición entre los nombres de columna de las tablas de entrada y candidatas. En la segunda se calculan la similitud de Jaccard de las dos tablas utilizando la puntuación de coincidencia de dicha la matriz.

En el trabajo de Nargesian et al. (2018) se proponen tres modelos estadísticos para estimar la probabilidad de que dos tablas contengan valores que están en el mismo dominio:

- En el primer caso, llamado *dominios de conjuntos*, se usa el tamaño de la intersección de valores entre dos columnas.

- En el segundo caso, llamado *dominios semánticos*, miden la similitud semántica entre los valores asignando las columnas a clases.

- En el tercer caso, llamado *dominios de lenguaje natural*, mide la semántica de valores expresados mediante lenguaje natural.

En esta aproximación usan *word embeddings* sobre documentos de Wikipedia para definir dominios de lenguaje natural, usando pruebas estadísticas entre los vectores para evaluar la probabilidad de que dos atributos pertenezcan al mismo dominio.

En (Das Sarma et al., 2012) los autores tienen como objetivo encontrar tablas relacionadas para aumentar la tabla de entrada con filas o columnas adicionales. Para ello consideran la relación que se da entre conjuntos de entidades de las tablas de entrada y candidatas. La relación entre dos entidades se estima representando entidades como conjuntos de etiquetas ponderadas (de una base de conocimientos o de un conjunto de tablas) y tomando su producto escalar. El trabajo propone múltiples métodos para agregar puntajes que determinen la relación entre dos conjuntos de entidades.

Yakout et al. (2012) proponen *InfoGather*, un método holístico para aumentar tablas que soporta tres operaciones centrales: aumento por nombre de columna, aumento por ejemplos y descubrimiento de nombres de columna. Consideran las similitudes en cuanto a elementos, incluido el contexto de la tabla, la URL, las tuplas, los nombres de columna, los valores de columna y los datos de la tabla, así como la similitud entre elementos de la tabla y el contexto. Miden la similitud utilizando el producto vectorial de los vectores de términos ponderados mediante TF-IDF. Luego, las puntuaciones de similitud a nivel de elemento se combinan como características en un modelo de aprendizaje automático. Este trabajo se amplió en (Zhang and Chakrabarti, 2013) para incorporar tablas con atributos numéricos y variables en el tiempo.

En (Nguyen et al., 2015) los autores consideran la diversidad de las tablas devueltas. Se centran en dos elementos de la tabla: nombres de columna y datos de tabla. El primero es similar en espíritu al motor de búsqueda y combinación de Lehmberg et al. (2015). Este último funciona midiendo la similitud entre las columnas de la tabla, que se representan como vectores de frecuencia de términos.

A diferencia de los métodos anteriores, que consideran las tablas como la unidad de recuperación, Limaye et al. (2010) devuelven una lista ordenada de celdas como resultado. Entrenan un método de aprendizaje automático para anotar entidades en celdas de tablas, columnas con tipos y relaciones entre columnas. Luego, la búsqueda se realiza emitiendo una consulta estructurada generada de manera automática.

2.2. Aumento de tablas

El aumento de tablas se refiere a la tarea de ampliar una tabla semilla con datos adicionales. Esta área engloba tres tareas: extensión de filas, extensión de columnas y completar datos. La Figura 2.1 muestra una ilustración de

los tres casos. Estas funcionalidades las podría ofrecer un agente inteligente que tenga como objetivo brindar asistencia a las personas que trabajan con tablas (Zhang and Balog, 2017b).

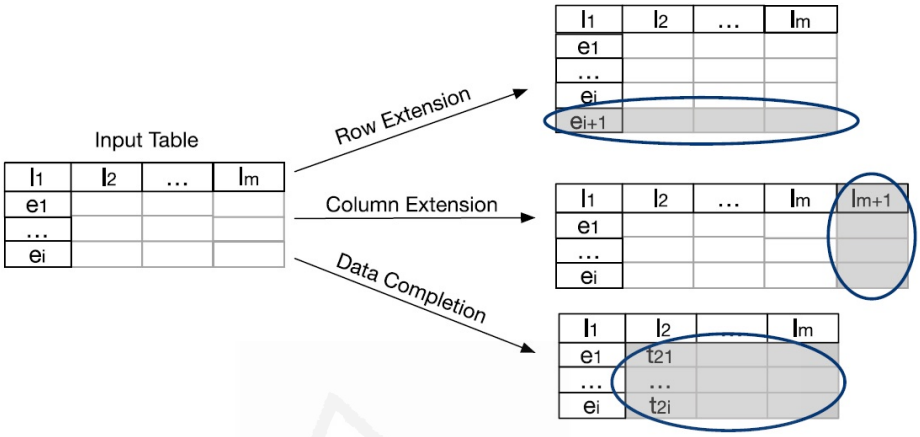


Figura 2.1: Ilustración de tres tareas para el aumento de tablas: extensión de filas (*row extension*), extensión de columnas (*column extension*) y completar datos (*data completion*). Fuente: Zhang et al. (2020).

2.2.1. Extensión de filas

La extensión de filas tiene como objetivo extender una tabla determinada con más filas o elementos de fila (ver Figura 2.2). Se centra principalmente en un tipo particular de tabla, las tablas relacionales. Más específicamente, la extensión de filas se dirige principalmente a tablas relacionales horizontales, donde las filas representan entidades y las columnas describen los atributos de esas entidades. En tales tablas, generalmente existe una columna principal (o *columna clave*) conteniendo principalmente entidades (Bhagavatula et al., 2015a; Venetis et al., 2011). En lugar de proporcionar directamente una tupla completa (fila), el trabajo existente se ha centrado en identificar entidades para poblar dichas columnas centrales, es decir, el escenario de la parte superior de la Figura 2.2.

Rellenar entidades en la columna principal de una tabla es similar al problema de la *expansión del concepto*, también conocido como expansión del conjunto de entidades, donde un conjunto dado de entidades semilla debe completarse con entidades adicionales (Bron et al., 2013; He and Xin, 2011; Metzger et al., 2013, 2014). Los métodos existentes para la expansión de conceptos adolecen de dos problemas principales: la ambigüedad de la entrada y la deriva semántica, es decir, las entidades que pertenecen a diferentes conceptos se mezclan durante la expansión.

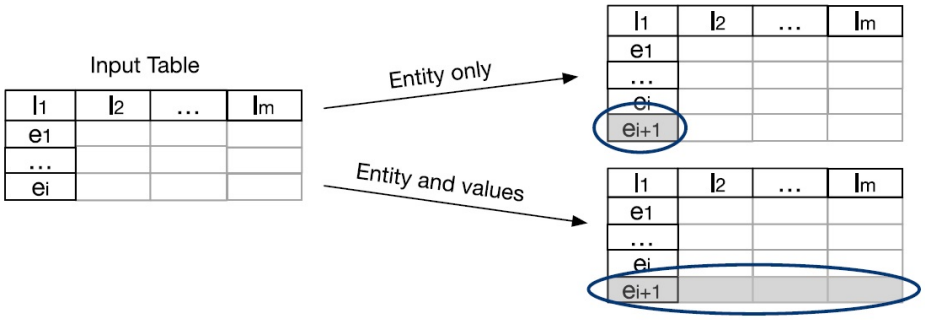


Figura 2.2: Ilustración de la tarea de extensión de filas agregando solo una entidad (*entity only*) o una fila completa, es decir, una entidad y valores de celda (*entity and values*). Fuente: Zhang et al. (2020).

Motivados por la intuición de que las tablas tienden a agrupar entidades que pertenecen a un concepto coherente, Wang et al. (2015) aprovechan las tablas web para la tarea de expansión de conceptos con el objetivo de evitar la deriva semántica. Proporcionan tanto las entidades semilla como un nombre de concepto como aporte. Primero, recuperan tablas relacionadas con las entidades semilla. Luego utilizan un método de clasificación basado en grafos para clasificar las entidades candidatas que coexisten con las entidades semilla en esas tablas. Primero expanden el conjunto agregando iterativamente las tablas más relevantes basadas en la verosimilitud del concepto y recolectando entidades allí. Luego refinan la estimación anterior y eliminan las tablas menos relevantes basadas en información más completa. Si bien este método está desarrollado para la expansión de conceptos, es directamente aplicable al problema de poblar entidades en una columna clave.

En (Sarma et al., 2012), los autores buscan tablas complementarias de entidades que estén relacionadas semánticamente con entidades en la tabla de entrada. Luego, las k tablas más relacionadas se utilizan para completar la tabla de entrada.

Un enfoque similar se toma en *InfoGather* (Yakout et al., 2012), sistema del que ya se habló más arriba. En esta aproximación, primero buscan tablas relacionadas y luego consideran entidades de estas tablas, ponderadas por las puntuaciones de relación de la tabla. Construyen un grafo de coincidencia de esquemas entre tablas web basado en la similitud de tablas por pares. A pesar del uso de técnicas escalables, esto sigue siendo computacionalmente costoso, lo cual es una limitación principal del enfoque.

En lugar de confiar solo en tablas relacionadas de un conjunto de tablas, Zhang and Balog (2017b) también consideran una base de conocimiento, DBpedia,[2] para identificar entidades candidatas. Para ello recopilan entidades que comparten los mismos tipos o categorías con las entidades de entrada de DBpedia y entidades de tablas similares, es decir, tablas que comparten entidades semilla, que tienen títulos similares o que incluyen los mismos nombres de columnas. Encuentran que la información de tipo de entidad en DBpedia es demasiado general para ayudar a identificar candidatos relevantes y terminan usando solo información de categoría cuando extraen candidatos de DBpedia. También se muestra que el uso de tablas relacionadas y el uso de una base de conocimiento son complementarias a la hora de identificar entidades candidatas.

En un trabajo reciente, Zhang et al. (2019) utilizan Word2vec para entrenar *word embeddings* para entidades de columnas celave. La combinación de las puntuaciones de similitud y las puntuaciones basadas en la probabilidad de Zhang and Balog (2017b) da como resultado mejoras de rendimiento adicionales.

2.2.2. Extensión de columnas

La segunda tarea que forma parte del área del aumento de tablas es la extensión de columnas. Esta tarea consiste en extender una tabla añadiendo nuevas columnas, similar a la operación de combinación de tablas (*join*). Se suele afrontar localizando en primer lugar tablas similares para después revisar los nombres de columnas y valores almacenados. Hay dos formas de aproximar esta tarea: añadiendo únicamente el nombre de la nueva columna o añadiendo tanto el nombre de columna como el contenido de las celdas (ver Figura 2.3).

Vamos a ver en primer lugar las aproximaciones realizadas para añadir únicamente el nombre de la columna. Como las columnas de la tabla a menudo corresponden a entidades, esta tarea también se conoce como *descubrimiento de atributos* (Yakout et al., 2012) o *autocompletado de esquemas* (Cafarella et al., 2008).

El sistema WebTables (Cafarella et al., 2008) implementa esta funcionalidad basándose en la base de datos de estadísticas de correlación de atributos *ACSDb*.[3] Esta base de datos contiene estadísticas de frecuencia de atributos y pares de atributos coexistentes en un conjunto de tablas. Comprende 5,4 millones de nombres de atributos únicos y 2,6 millones de esquemas únicos.

El método estadístico propuesto en (Cafarella et al., 2008) fue el primer enfoque para la extensión de columnas y se encontró que podía proporcionar

[2] https://es.dbpedia.org.
[3] https://web.eecs.umich.edu/~michjc/data/acsdb.html.

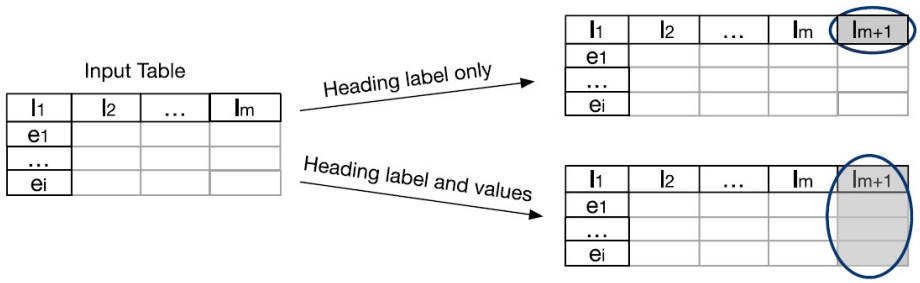


Figura 2.3: Ilustración de la extensión de columnas agregando solo un nombre de columna (*heading label only*) o una columna completa, es decir, el nombre de columna y valores de las celdas (*heading label and values*). Fuente: Zhang et al. (2020).

sugerencias de encabezados coherentes. Sin embargo, investigaciones posteriores han demostrado que considerar funciones adicionales puede mejorar aún más el rendimiento.

Sarma et al. (2012) se centran en encontrar tablas relacionadas, con el objetivo de complementar el esquema. Consideran dos factores: la cobertura de entidades y los beneficios potenciales de agregar atributos adicionales de esas tablas. Una vez más, se apoyan en la tarea de búsqueda de tablas.

En (Zhang and Balog, 2017b) intentan encontrar los encabezados (nombres de columnas) que se pueden colocar como la siguiente columna en una tabla de entrada. Primero encuentran los encabezados candidatos de tablas similares (la misma estrategia que también usan para la población de filas). Observan que las entidades de entrada y el título de la tabla contribuyen de manera comparable a la identificación de candidatos, mientras que los encabezados de las tablas son el componente menos importante. Sin embargo, al igual que la población de filas, todas estas fuentes son complementarias, es decir, cada fuente puede identificar encabezados candidatos que ninguno de los demás pudo. En un paso de ordenación posterior, los candidatos se puntúan según la similitud de la tabla, agregando similitudes por elementos entre la tabla de entrada y las tablas relacionadas.

Zhang et al. (2019) utilizan Word2vec para entrenar *word embeddings* para nombres de columnas. De manera similar a la población de filas, la combinación de las puntuaciones de similitud incorporadas con las probabilidades de Zhang and Balog (2017b) produce mejoras adicionales en el rendimiento.

Los enfoques anteriores difieren en los datos que utilizan como entrada, es decir, si utilizan solo los títulos de la tabla (Cafarella et al., 2008; Lehmberg

et al., 2015) o la tabla completa (Sarma et al., 2012; Zhang and Balog, 2017b).

La segunda variante de extensión de columnas de la que hablamos consistía en añadir tanto el nombre de la columna como el contenido de las celdas. Bhagavatula et al. (2013) presentan la tarea de unión relevante, que devuelve una lista ordenada de tripletes de columnas para una tabla de entrada determinada. Cada triplete consta de columna origen, columna emparejada y columna candidata. Proponen una medida de relación semántica para encontrar tablas candidatas en páginas de Wikipedia relacionadas, donde la relación entre páginas se estima en función de las intersecciones de enlaces. Su idea es calcular la similitud entre columnas de modo que si la columna origen y la columna emparejada comparten valores muy similares, entonces la tabla de entrada se puede ampliar con la columna candidata. Estas columnas candidatas se clasifican como relevantes o no relevantes utilizando un modelo de clasificación lineal antes de realizar la unión real. Para reducir el número de columnas candidatas, algunas se filtran en una etapa de preprocesamiento utilizando heurísticas simples: las columnas que se mantienen deben ser no numéricas, tener más de cuatro filas y una longitud promedio de cadena mayor a cuatro. En sus experimentos encuentran que valores más dispares en la columna de origen y un porcentaje de coincidencia más alto conducen a uniones de mejor calidad.

La operación de extensión de columnas también se aborda en el motor de búsqueda propuesto por Lehmberg et al. (2015). En primer lugar buscan tablas relacionadas basadas en nombres de columnas. Luego aplican una serie de combinaciones (*join*) entre la tabla de consulta y las tablas devueltas. Posteriormente, se realiza una operación de consolidación para combinar columnas. Específicamente, emplean un operador de coincidencia que se basa en datos de bases de conocimiento. Dadas dos columnas, usan tanto una medida de coincidencia difusa (distancia de Levenshtein) como la coincidencia exacta entre los encabezados. Los resultados muestran que una coincidencia difusa devuelve en promedio 3,4 veces más tablas que una coincidencia exacta. Entre los diferentes conjuntos de tablas, las tablas web proporcionan la mayor cantidad de tablas relevantes y las tablas de Wikipedia tienden a estar sesgadas y pobladas sobre ciertos temas, como países y películas.

2.2.3. Completar datos

La tarea de completar datos para una tabla se refiere a rellenar las celdas vacías de la tabla. Podemos identificar dos subtareas aquí, consistentes en rellenar todos los datos de una columna (como haría un *join*) o solo una celda (equivalente a la tarea de *imputación de datos*). La Figura 2.4 muestra visualmente ambas tareas.

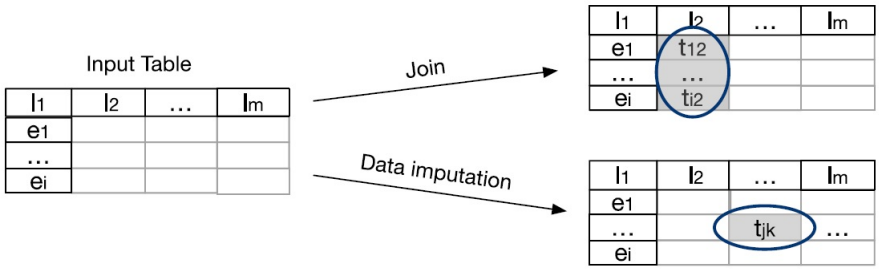


Figura 2.4: Ilustración de las tareas de completar de datos: combinación (*join*) e imputación (*data imputation*). Fuente: Zhang et al. (2020).

En cuanto a la tarea de identificar todos los valores de la columna, en el sistema InfoGather (Yakout et al., 2012) toman una tabla incompleta como entrada para buscar tablas coincidentes. Luego extraen valores de las celdas de esas tablas. Vale la pena señalar que InfoGather se centra en encontrar valores que sean entidades. Una versión ampliada del sistema (Zhang and Chakrabarti, 2013) incluye tanto valores numéricos como variables en el tiempo. Los autores utilizan modelos basados en grafos no dirigidos y crean un grafo semántico que etiqueta las columnas con unidades, escalas y marcas de tiempo, calculando las coincidencias semánticas entre columnas. Sus experimentos se realizan en tres tipos de tablas: empresas (ingresos y beneficios), países (área y tasas impositiva) y ciudades (población). Las reglas de conversión de unidades diseñadas manualmente logran una mayor cobertura que los métodos de coincidencia de esquemas basados en cadenas de texto.

La operación de extensión en el sistema OCTOPUS (Cafarella et al., 2009) permite al usuario agregar más columnas a una tabla realizando una combinación. Se necesita una consulta de palabras clave y una columna de la tabla objeto como entrada, donde la palabra clave describe la columna recién agregada. A diferencia de una combinación normal, la columna agregada no es necesariamente una columna. Puede formarse fila por fila combinando información de varias tablas relacionadas. No obstante, la metodología seguida emplea métodos simples como la distancia de edición para la coincidencia entre esquemas, lo que deja margen para la mejora.

La otra subtarea mencionada para completar datos es la de imputación. Para ello, Ahmadov et al. (2015) presentan un método de imputación híbrido que combina un enfoque basado en búsquedas en un conjunto de tablas web y un enfoque basado en modelos que utiliza aprendizaje automático para predecir el valor de una celda faltante. Sin embargo, ser capaz de decidir automáticamente cuándo realizar una búsqueda simple y cuándo emplear un modelo de aprendizaje automático sigue siendo un desafío abierto en este trabajo.

En (Zhang and Balog, 2019) se propone un marco de trabajo para abordar varios aspectos novedosos de este problema, que incluyen: (i) permitir que una celda tenga valores múltiples, posiblemente en conflicto; (ii) complementar los valores predichos con evidencia de apoyo; (iii) combinar evidencia de múltiples fuentes; y (iv) manejar el caso donde una celda debe dejarse vacía. Este marco utiliza un gran conjunto de tablas y una base de conocimientos como fuentes de datos. Consta de componentes de preprocesamiento, búsqueda de valor candidato y clasificación de valor.

2.3. Trabajos relacionados

El objetivo principal de esta tesis es el desarrollo de un sistema que permita la búsqueda e integración de datos dada una tabla de partida. Por tanto, estamos en un escenario donde la búsqueda está basada en tablas en lugar de en palabras clave. Además, al tratar las operaciones de unión (*union*) y combinación (*join*) abarcamos la área de aumento de tablas centrándonos en la extensión de filas (equivalente a *union*) y columnas (equivalente a *join*).

En esta sección se discuten aquellos trabajos del área de la búsqueda e integración de datos que usan búsquedas basadas en tablas. Se dedica una sección a cada uno de ellos. Estos enfoques difieren principalmente en la información de las tablas que se utiliza para calcular la similitud entre ellas (título de la tabla, entidades de la tabla, encabezados de columna o valores de celda), en la representación formal de estos datos y en las medidas de similitud utilizadas.

2.3.1. Búsqueda

En (Ahmadov et al., 2015) los autores usan entidades con nombre (personas, organizaciones y ubicaciones) y encabezados de columna para la recuperación de tablas. Similar a este trabajo, Lehmberg et al. (2015) compararon los nombres de las columnas entre las tablas de entrada y las tablas candidatas. Primero, utilizando la coincidencia exacta de términos seleccionaron tablas con al menos un encabezado de columna en común con la tabla de consulta. Luego, la similitud entre los encabezados de columna restantes se calculó utilizando la distancia de edición y la similitud de Jaccard.

En los trabajos de Yakout et al. (2012) y Zhang and Chakrabarti (2013) los autores aumentaron la tabla de entrada con nuevos encabezados, ejemplos y columnas. La similitud entre tablas se midió utilizando diferentes tipos de información: contexto de tabla, URL, tuplas, encabezados de columna, valores de columna, datos de tabla, valores numéricos y valores de tiempo. Posteriormente, estas medidas de similitud se incorporaron como características en un modelo de aprendizaje automático.

Los enfoques más recientes al área se basan en espacios semánticos para representar datos tabulares, aplicando medidas de similitud de vectores para calcular la relevancia entre tablas (Nguyen et al., 2015). En este sentido, Zhang and Balog (2018) combinaron dos espacios vectoriales semánticos: uno basado en una base de conocimiento (DBpedia) y el otro usando *word embeddings* previamente entrenados (Word2vec). Utilizaron toda la información disponible en las tablas para la tarea de recuperación (título, pie, encabezados y entidades).

El trabajo de Shraga et al. (2020) también utilizó Word2vec como fuente de vectores semánticos. La información de la tabla se separó en cuatro espacios semánticos: descripción (título y pie), esquema (encabezados de las columnas), registros (filas de las tablas) y facetas (columnas de las tablas). Luego, se aplicaron diferentes arquitecturas de redes neuronales a cada espacio semántico, como una red convolucional a la descripción, un perceptrón multicapa al esquema y una red neuronal convolucional 3D a registros y facetas. Finalmente, estos cuatro espacios semánticos se combinaron utilizando una unidad multimodal (Arevalo et al., 2017).

Para recuperar tablas compatibles con una tabla de entrada, Nargesian et al. (2018) intentaron estimar si el contenido de la tabla pertenecía al mismo dominio. Aplicaron tres modelos estadísticos: valores de intersección entre dos columnas, similitud semántica entre valores que mapean las columnas a clases en una ontología y el uso de *word embeddings* para medir la similitud entre valores textuales.

A diferencia de los enfoques anteriores, en (Trabelsi et al., 2019) los autores entrenaron su propio modelo de incrustación de palabras utilizando tablas de Wikipedia. Utilizaron un modelo *skip-gram* adaptado a tablas en las que se utilizaban como contexto los pies, los atributos (encabezados de columna) y el contenido (valores de las celdas).

Todos los modelos de incrustación de palabras mencionados anteriormente son no contextuales. Hasta la fecha, el único trabajo que utiliza *word embeddings* contextuales para la búsqueda basada en tablas se describió en (Chen et al., 2020). Los autores utilizaron una versión previamente entrenada de BERT, aprovechando la diferente información disponible en la tabla (tanto textual como numérica) para proporcionar a BERT un contexto: título, pie, encabezados de las columnas y valores de las celdas.

Una diferencia importante entre el presente trabajo y el de Chen et al. (2020) es el propósito de la tarea de recuperación de tablas. En su caso, era un objetivo en sí mismo, mientras que en esta tesis un paso previo a la integración. Otra diferencia es que en esta tesis se han implementado dos medidas de relevancia distintas, dependiendo de si el objetivo final es integrar tablas mediante operaciones de unión o combinación.

Finalmente, otras novedades de esta tesis incluyen la experimentación con un modelo de lenguaje contextual ajustado (*fine-tunning*) sobre un conjunto de datos tabulares, así como el estudio detallado de cómo el

contexto influye en el rendimiento del sistema. En la actualidad, aunque existen propuestas de adaptación de modelos contextuales al dominio de los datos tabulares, estos se enfocan principalmente a la tarea de búsqueda de respuestas sobre tablas y no son directamente aplicables a las tarea de búsqueda e integración que abordamos aquí (Herzig et al., 2020; Yin et al., 2020).

2.3.2. Integración

Una vez que se ha recuperado un conjunto de tablas similares, el siguiente paso de nuestro sistema es determinar si es posible ampliar la tabla de consulta con filas compatibles (*union*) o nuevas columnas (*join*).

Los enfoques anteriores para la extensión de filas han utilizado algún tipo de similitud entre tablas para encontrar su compatibilidad. Por ejemplo, Wang et al. (2015) introdujeron nombres de conceptos como entrada, junto con entidades semilla para evitar la ambigüedad léxica. El trabajo de Zhang and Balog (2017b) mejoró este método utilizando DBpedia para identificar entidades candidatas. Solo el trabajo de Zhang et al. (2019) utilizaba incrustaciones de palabras (Word2vec) para esta tarea.

En el área de la extensión de columnas, el enfoque presentado por Cafarella et al. (2008) se basó en una base de datos que incluía estadísticas de frecuencia de atributos y pares de atributos concurrentes en un gran conjunto de tablas, como ya se mencionó con anterioridad. En esta tarea, el trabajo de Zhang and Balog (2017b) aprovechó los títulos de las tablas y la similitud entre ellas. Estos dos estudios se centraron únicamente en los encabezados de las columnas y no tenían como objetivo realizar operaciones de combinación entre tablas.

Por otro lado, existen otros dos trabajos que han estudiado la operación de combinación. El enfoque propuesto por Bhagavatula et al. (2013) se basó en tablas de Wikipedia. La relación entre las tablas se estimó en función de las intersecciones de enlaces de las páginas de Wikipedia. A partir de una columna coincidente (una columna que comparte valores muy similares), encontraron una columna candidata aplicando un modelo de clasificación lineal y diferentes heurísticas, como la cantidad de datos numéricos que contenía.

Otro trabajo (Lehmberg et al., 2015) encontró nuevas columnas y datos de tablas similares utilizando la distancia de Levenshtein en los encabezados de las columnas. Luego, la operación de combinación se llevó a cabo con una serie de combinaciones externas entre la tabla de consulta y las tablas devueltas.

Finalmente, el trabajo de Flores Herrera et al. (2021) se enfocó en descubrir atributos combinables entre datos estructurados. Específicamente, definieron el concepto de *calidad de combinación* que mide tanto los valores

como la proporción de cardinalidad para determinar los mejores atributos candidatos a combinar.

Sobre la base de trabajos anteriores, el presente estudio es el primer intento de utilizar y analizar ampliamente el impacto de las incrustaciones de palabras contextuales en la tarea de integración de datos tabulares.

3

Modelos de lenguaje contextuales

En este capítulo se ofrece una introducción a la inteligencia artificial, el aprendizaje automático, el aprendizaje profundo, el procesamiento de lenguaje natural y las incrustaciones de palabras contextuales y no contextuales. El objetivo es situar al lector en las tecnologías que se han utilizado como base para el desarrollo del sistema de búsqueda e integración de datos tabulares.

3.1. Inteligencia artificial

En primer lugar vamos a describir los conceptos de inteligencia artificial, aprendizaje automático y aprendizaje profundo. La Figura 3.1 muestra la relación que existe entre estas tres disciplinas.

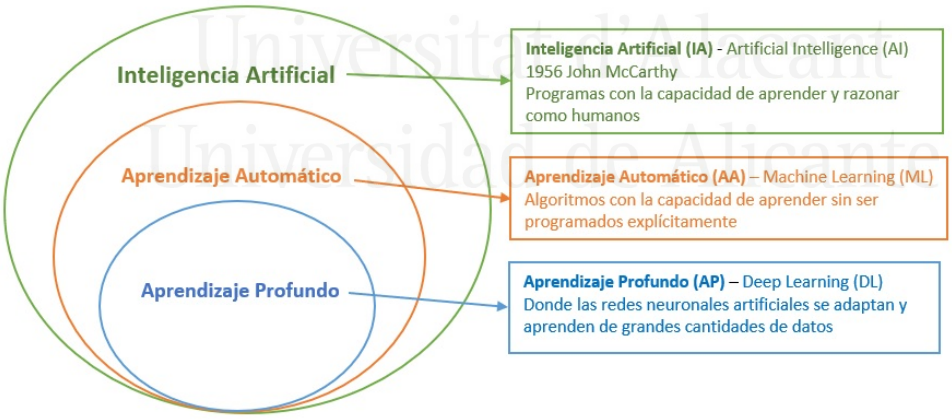


Figura 3.1: Relación entre inteligencia artificial, aprendizaje automático y aprendizaje profundo.

Dentro de las líneas que desarrolla la inteligencia artificial, tenemos categorías como el aprendizaje automático (*machine learning*) y subcategorías dentro de esta como el aprendizaje profundo (*deep learning*).

La definición de inteligencia artificial utilizada por primera vez por McCarthy en 1955 (McCarthy et al., 2006) nos dice que es una disciplina del campo de la informática que busca la creación de máquinas que puedan imitar comportamientos inteligentes. Estos comportamientos pueden ser muy diversos, tales como conducir, analizar patrones, reconocimiento de voz, análisis de sentimientos o jugar al ajedrez. Simulan de esta manera un comportamiento inteligente y en ciertas áreas logran alcanzar un rendimiento superior al humano. No obstante, si cogemos cualquiera de estos ejemplos que sobresalen en un dominio muy específico e intentamos que realice otra tarea, se verá que el resultado que obtiene no es el esperado. Se trata de inteligencia artificial *estrecha*, lejos de las capacidades de una inteligencia *general* como la que ofrecen los humanos, capaz de resolver numerosas tareas diferentes.

Dentro del campo de la inteligencia artificial podemos encontrarnos diferentes subcategorías que responden a diferentes comportamientos inteligentes. Por ejemplo, la robótica se centra en la capacidad de las máquinas de moverse y adaptarse al entorno, mientras que el PLN busca conseguir que las máquinas entiendan y se comuniquen con las personas usando el lenguaje humano.

Los primeros sistemas de inteligencia artificial estaban basados en reglas introducidas por expertos en el dominio. Esto limitaba las capacidades del sistema a lo que un humano fuera capaz de codificar y transmitirle. El aprendizaje automático vino a sustituir esta forma de trabajar, ya que se enfoca en desarrollar sistemas que pueden aprender y mejorar su rendimiento en tareas específicas sin ser programados explícitamente para realizarlas (Samuel, 1988). Esto se logra mediante el uso de algoritmos de aprendizaje automático, que permiten a los sistemas adaptarse y mejorar a medida que reciben más datos. El aprendizaje automático se puede dividir en diferentes tipos de aprendizaje, dependiendo de los datos de entrada: aprendizaje supervisado (los datos de entrada están etiquetados), aprendizaje no supervisado (los datos de entrada no están etiquetados) y aprendizaje por refuerzo (se fija un objetivo y un mecanismo de recompensa/penalización). El aprendizaje automático se utiliza ampliamente en aplicaciones como la detección de *spam*, la recomendación de productos y la conducción autónoma.

Existen numerosos algoritmos de aprendizaje automático. Entre ellos, las redes neuronales son uno de los más populares. Una red neuronal es un tipo de algoritmo (modelo matemático) de aprendizaje automático que se inspira en la forma en que funciona el cerebro humano. Una red neuronal consta de varios nodos interconectados, cada uno de los cuales representa una unidad de procesamiento llamada *neurona*. Las entradas a la red neuronal se procesan en estas neuronas y las salidas se generan en otras neuronas en función de las conexiones entre ellas. Una red neuronal puede aprender a realizar tareas específicas a medida que se le presentan datos de

entrenamiento y se ajustan los pesos de las conexiones entre las neuronas en consecuencia.

Las arquitecturas de redes neuronales suelen constar de varias capas. Cuantas más capas existan, más profunda es la red y mayor su capacidad de aprendizaje. Cuando una red neuronal artificial consta de un número elevado de capas, hablamos entonces de arquitecturas de aprendizaje profundo o *deep learning*. Estas redes neuronales profundas tienen la capacidad de extraer características de los datos de entrada de forma automática, lo que les permite aprender y generalizar de manera efectiva. El aprendizaje profundo se ha utilizado con éxito en una amplia variedad de aplicaciones, como el reconocimiento de imágenes y el PLN. A diferencia de otras técnicas de aprendizaje automático, el aprendizaje profundo puede manejar conjuntos de datos muy grandes y complejos de manera eficiente.

La Figura 3.2 muestra un resumen de los tipos de aprendizaje existentes y de distintas tipologías de redes neuronales que hay dentro del campo del aprendizaje profundo (Mahesh, 2020).

Figura 3.2: Tipologías de aprendizaje y arquitecturas de redes neuronales dentro del campo del aprendizaje profundo. Fuente: Mahesh (2020).

3.2. Redes neuronales

A continuación vamos a describir algunas de las arquitecturas más populares de redes neuronales: las *recurrent neural networks* (RNN), las *long short-term memory networks* (LSTM) y los *Transformers*.

3.2.1. RNN

Una RNN es un tipo de red neuronal que tiene conexiones que vuelven a su propia entrada. Esto significa que la salida de la red en cualquier momento también se puede usar como entrada en el futuro. Esto hace que las RNN sean especialmente adecuadas para tareas en las que se requiere tener en cuenta información temporal, como el PLN o la predicción de series temporales.

Las redes neuronales como el perceptrón multicapa permiten procesar un solo dato cada vez (como un sonido o una imagen), pero si queremos tratar una secuencia de sonidos (como una conversación) o de imágenes (como un vídeo) esta arquitectura es insuficiente. Las RNN permiten resolver esta limitación, ya que son capaces de procesar diferentes tipos de secuencias.

Para entender cómo funcionan las RNN necesitamos primero entender el concepto de secuencia. Una secuencia es, por ejemplo, un texto escrito donde para comprender su contenido no basta con que leamos cada palabra de manera individual, pues realmente nuestro cerebro concatena todas las palabras leídas hasta el momento permitiéndonos comprender la idea central de dicho texto. Por tanto, una secuencia es una serie de datos (imágenes, palabras, sonidos, etc.) que siguen un orden específico y tienen únicamente significado cuando se analizan en conjunto y no de manera individual.

3.2.2. LSTM

Las LSTM son un tipo específico de RNN que se ha diseñado para tener una memoria a largo plazo y poder manejar mejor la información temporal. Esto se logra mediante la introducción de celdas de memoria y puertas de olvido que permiten que la red controle qué información se guarda y qué información se olvida. Fueron introducidos por Hochreiter and Schmidhuber (1997).

Su característica principal es que la información puede persistir introduciendo bucles en el diagrama de la red, por lo que pueden recordar estados previos y utilizar esta información para decidir cuál será el siguiente. Mientras las RNN estándar pueden modelar dependencias a corto plazo (es decir, relaciones cercanas en la serie cronológica), las LSTM pueden aprender dependencias largas, por lo que se podría decir que tienen una memoria a más largo plazo.

3.2.3. Transformers

La arquitectura Transformer (Vaswani et al., 2019) es una arquitectura de red neuronal que se utiliza principalmente para el PLN, en tareas como la traducción automática o el resumen automático de textos. Se caracteriza por utilizar una técnica llamada *atención* para permitir que la red aprenda a enfocarse en las partes relevantes de la entrada a medida que se procesa. Estas capas de atención codifican cada palabra de una frase en función del resto de la secuencia, permitiendo así introducir el contexto en la representación matemática del texto, motivo por el cual a los modelos basados en esta arquitectura se les denomina también modelos contextuales. En la Figura 3.3 podemos ver su arquitectura.

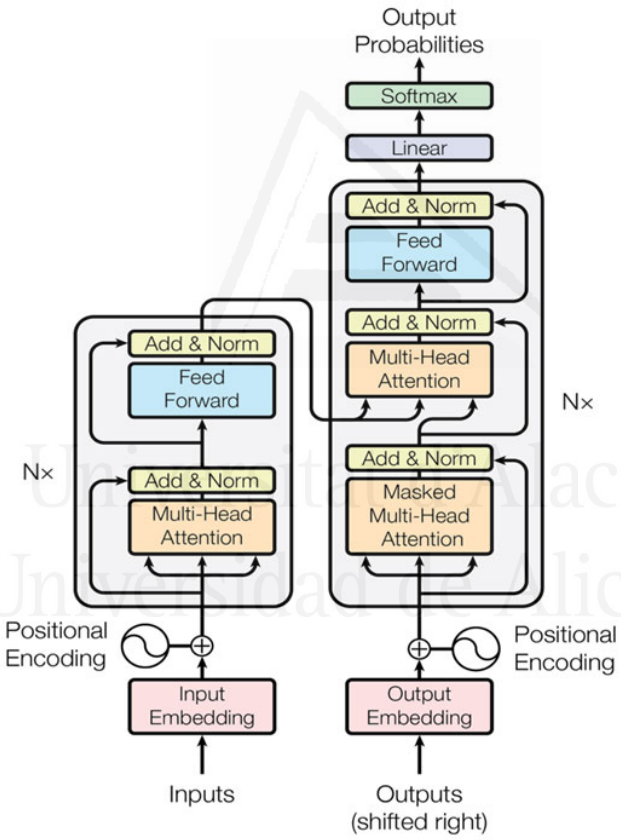


Figura 3.3: Arquitectura del modelo Transformer. Fuente: Vaswani et al. (2019).

Aunque la aplicación de esta arquitectura originalmente fue en el campo del PLN, en la actualidad se ha extendido de manera exitosa en otros problemas que involucran secuencias de datos, como el modelado de secuencias musicales y el análisis de secuencias de imágenes. En general,

puede ser útiles en cualquier tarea que implique la manipulación de secuencias de datos de longitud variable.

3.3. Procesamiento del lenguaje natural

El PLN es el campo de la inteligencia artificial que se ocupa de la interacción entre las computadoras y el lenguaje humano, tanto para que la máquina entienda el lenguaje humano como para que lo genere. Esto incluye tareas como la traducción automática, el resumen automático de texto, la generación de texto, la desambigüación de palabras y la comprensión de preguntas en lenguaje natural. El PLN es un campo interdisciplinario que combina la informática, la lingüística y la psicología cognitiva, y se basa en una amplia variedad de técnicas de aprendizaje automático y modelos de lenguaje.

En la Figura 3.4 se muestra la relación del PLN con la inteligencia artificial, la lingüística computacional, el aprendizaje automático y el aprendizaje profundo.

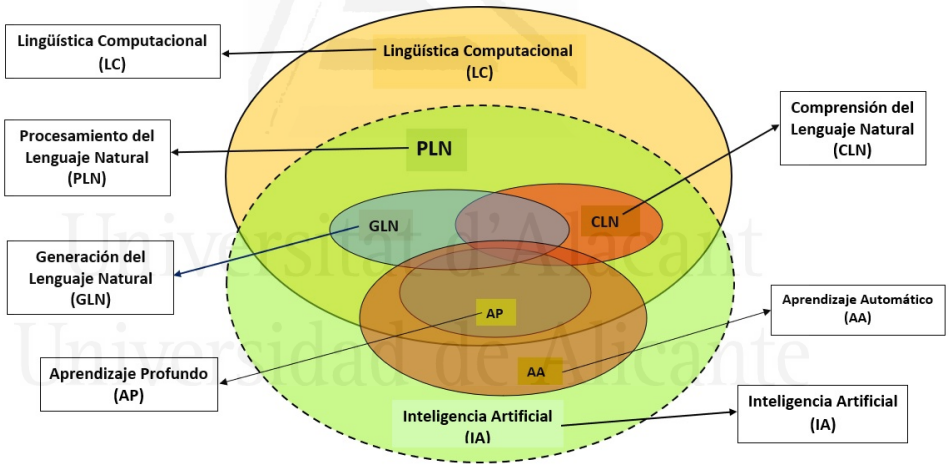


Figura 3.4: Relación entre el PLN, la inteligencia artificial, la lingüística computacional, el aprendizaje automático y el aprendizaje profundo.

El PLN utiliza métodos de varias disciplinas para permitir que las computadoras entiendan el lenguaje humano tanto en forma escrita como verbal. Esto funciona tomando datos no estructurados y convirtiéndolos en un formato de datos estructurados para permitir que las máquinas entiendan el habla/texto. Dentro del PLN se desprenden dos categorías principales: la comprensión del lenguaje natural y la generación del lenguaje natural (ver Figura 3.5).

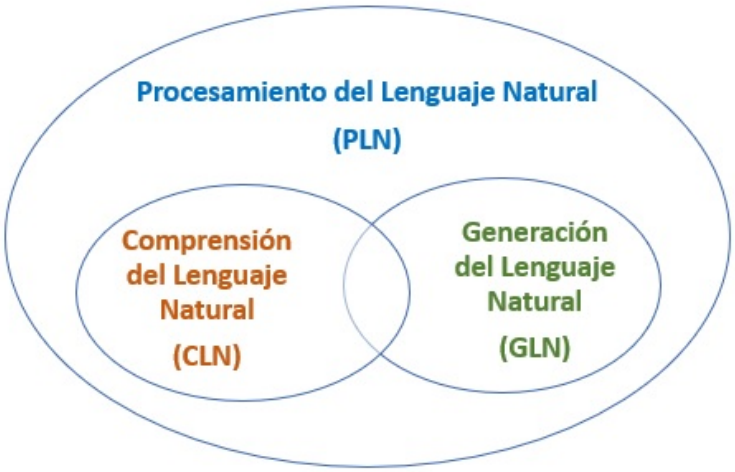


Figura 3.5: Subcategorías del PLN.

La comprensión del lenguaje natural utiliza el análisis sintáctico y semántico del texto y el habla para determinar el significado de una oración. Busca comprender el idioma que hablamos como humanos. La generación del lenguaje natural toma conjuntos de datos no estructurados para crear oraciones que los humanos podamos entender como significativas.

En el PLN las computadoras analizan el lenguaje humano, lo interpretan y dan significado para que pueda ser utilizado de manera práctica. Usando PLN podemos hacer tareas como la extracción de relaciones, análisis de sentimientos, reconocimiento del habla y clasificación de textos, entre otras.

La mayoría de algoritmos de aprendizaje automático y aprendizaje profundo existentes no pueden manejar directamente cadenas y texto sin formato. Estas tecnologías requieren convertir los datos textuales en valores numéricos antes de poder realizar cualquier tarea con ellos. Existen distintas técnicas para esto. Una forma tradicional de representar textos es mediante *one-hot encoding*, que es esencialmente un vector cuya longitud es igual al tamaño del vocabulario del corpus, donde las palabras que aparecen en el documento valen 1 y el resto valen 0. El problema de este tipo de representación es que se generan vectores muy dispersos, donde la mayoría de dimensiones están a 0.

Otra forma de obtener este vector numérico que representa al texto es usando *word embeddings*, la forma más utilizada en la actualidad para transformar los valores textuales en vectores numéricos. En la siguiente sección vamos a ver más en detalle estas tecnologías.

3.4. Incrustaciones de palabras

Las incrustaciones de palabras o *word embeddings* son vectores densos que representan el significado de una palabra como un punto en un espacio semántico (Bengio et al., 2003). Estas representaciones continuas se pueden utilizar en tareas posteriores de PLN, como la clasificación de textos (Lilleberg et al., 2015) y la búsqueda de respuestas (Shih et al., 2016). Desde una perspectiva lingüística, representan el significado distributivo de las palabras (Turney and Pantel, 2010), es decir, el significado que asume una palabra en un texto específico independientemente del significado que pueda tener en el diccionario (Harris, 1954; Firth, 1957). Por lo tanto, se aprenden representaciones similares de palabras que aparecen en contextos similares. En la literatura científica, se diferencian de los vectores semánticos tradicionales, donde el significado de una palabra se representa como un vector disperso con el peso de sus componentes calculado utilizando medidas como TF-IDF (Salton and Buckley, 1988).

La dimensionalidad de los vectores de incrustación de palabras suele oscilar entre 50 y 1.000 dimensiones, mucho más baja que la de los vectores semánticos dispersos tradicionales (Jurafsky and Martin, 2018). Esta reducción de dimensiones se basa en generalizaciones que capturan las relaciones semánticas entre palabras en función del contexto en el que aparecen.

Una vez entrenado, un *word embedding* se puede utilizar como entrada de una red neuronal u otro algoritmos para realizar tareas de PLN. A diferencia de la codificación mediante *one-hot*, los *word embeddings* son útiles porque representan las palabras en un espacio dimensional continuo y denso, lo que permite a las redes neuronales manipularlas de forma numérica y entrenarse para realizar tareas con ellas.

Collobert and Weston (2008) desarrollaron uno de los trabajos más relevantes en el uso de *word embeddings* como herramienta efectiva para diferentes tipos de tareas, presentando una arquitectura de red neuronal en la que se basan muchos de los enfoques actuales.

La Figura 3.6 presenta una línea temporal del desarrollo reciente de modelos de redes neuronales para la obtención de *word embeddings* (Peters et al., 1802; Pennington et al., 2014; Mikolov et al., 2013a; Bojanowski et al., 2017a; Ilić et al., 2018; Devlin et al., 2018; Radford et al., 2018).

Como se ve, el campo se ha estado desarrollando sorprendentemente rápido desde 2013, cuando los *word embeddings* fueron popularizados en gran parte gracias al trabajo de Mikolov et al. (2013a), quienes publicaron el modelo Word2vec. Esta técnica de aprendizaje automático se utiliza para generar *word embeddings*. Se basa en una red neuronal de dos capas que se entrena en un gran corpus de texto para predecir la palabra siguiente en una secuencia de palabras dada. Durante el entrenamiento, la red aprende a representar las palabras en un espacio vectorial de forma que las palabras

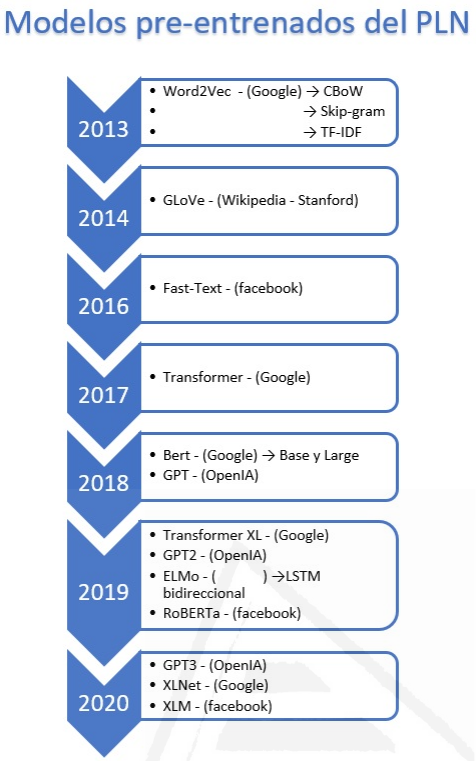


Figura 3.6: Modelos de lenguaje preentrenados para la obtención de *word embeddings*.

que tienen un significado similar estén cerca unas de otras en el espacio. Una vez entrenado, el modelo puede producir vectores de palabras para cualquier palabra del corpus de entrenamiento.

Otro modelo interesante es GLoVe (*Global Vectors for Word Representation*), aparecido en 2014 (Pennington et al., 2014). A partir de ese momento los *word embeddings* se han convertido en una de las corrientes principales dentro del PLN.

En 2016 la compañía Facebook AI Research desarrolló el modelo fastText, un método rápido y efectivo para aprender representaciones de palabras en forma de *word embeddings*. Esto es útil para lenguajes morfológicamente ricos, de modo que las representaciones de diferentes formas morfológicas de palabras se aprendan de forma independiente (Bojanowski et al., 2017b). En lugar de introducir palabras individuales en la red neuronal, fastText divide las palabras en varios n-gramas (subpalabras). En comparación con otros modelos, fastText acorta notablemente el tiempo de entrenamiento del modelo manteniendo la efectividad en la representación de los vectores numéricos generados.

Este tipo de modelos se llaman no contextuales o estáticos, ya que cada palabra que aparece durante el entrenamiento tiene una sola representación vectorial independientemente del lugar en dónde aparezca en la oración, sin tener en cuenta que las palabras tienen diferentes significados dependiendo del contexto en el que ocurren.

3.5. Modelos contextuales

En la actualidad las arquitecturas de tipo Transformer se están empleando para la obtención de *word embeddings* contextuales (Devlin et al., 2018). Pueden capturar los diferentes significados de las palabras polisémicas, ya que cada vector representa no una palabra sino un sentido. De esta manera, cada palabra se representa con diferentes *word embeddings*, una para cada contexto en el que la palabra puede aparecer. Durante el proceso de entrenamiento, las incrustaciones de palabras contextuales se han generado tomando en consideración las palabras circundantes, es decir, la secuencia de palabras en la oración o tramo de texto en el que aparece una palabra. La arquitectura Transformer incluye otras innovaciones, como los *word embeddings* posicionales, que permiten al algoritmo conocer la posición relativa de cada palabra del texto. Ejemplos de este tipo de representación son ELMo (Peters et al., 2018a), ULMFit (Howard and Ruder, 2018), BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019) y SentenceBERT (Reimers and Gurevych, 2019a), entre muchos otros (Liu et al., 2020).

En los experimentos presentados en el Capítulo 4, BERT y su variante RoBERTa se utilizan como representantes de estos modelos contextuales. BERT son las siglas de *Bidirectional Encoder Representations from Transformers*. Bidireccional significa que BERT aprende información del lado izquierdo y derecho del contexto de una palabra durante la fase de entrenamiento. Además, estos modelos basados en la arquitectura Transformer permiten la transferencia de conocimiento (*transfer learning*) en tareas de PLN. Es decir, el modelo BERT originalmente entrenado en un conjunto de datos (el modelo preentrenado) se puede usar para realizar tareas similares en otro conjunto de datos (el modelo afinado).

De esta manera, BERT, que se entrenó previamente en un gran corpus de texto sin etiquetar (incluida la Wikipedia completa y el Book Corpus), se puede ajustar para una amplia gama de tareas de PLN. Los sistemas actuales aprovechan las relaciones semánticas identificadas por los Transformers como un punto de partida para resolver un problema en lugar de construir un modelo desde cero, entrenando aún más el modelo (*fine-tuning*) en conjuntos de datos relativamente más pequeños para tareas específicas.

Otra diferencia notable entre estos modelos contextuales y sus predecesores estáticos es el uso de unidades de subpalabras en lugar de palabras

completas para representar el vocabulario del problema. Las incrustaciones de palabras se basan en el conjunto específico de tokens disponibles en el corpus utilizado para crear los vectores. Cuando aparece una palabra fuera del vocabulario (*out-of-vocabulary*) en un texto nuevo, los modelos basados en palabras no proporcionan representación para ella en el espacio semántico y, por lo tanto, la palabra se considera como desconocida.

Para manejar los extensos vocabularios comunes en corpus de lenguaje natural, BERT utiliza el algoritmo de segmentación de subpalabras *Word-Piece* (Wu et al., 2016). En él, el vocabulario se inicializa con caracteres individuales en el idioma y luego se agregan iterativamente al vocabulario las combinaciones de símbolos más frecuentes en el vocabulario. Por lo tanto, las subpalabras tienen su propia representación en el espacio semántico y a las palabras previamente desconocidas se les puede asignar una representación combinando los vectores de sus unidades de subpalabras subyacentes.

Con respecto a RoBERTa, este modelo proporciona una variante de BERT en la que la fase de preentrenamiento se optimizó con cambios en la elección de hiperparámetros, la tarea objetivo y el uso de lotes más grandes y secuencias de texto más largas. Además de eso, RoBERTa utiliza un algoritmo de segmentación diferente, *Byte-Pair Encoding* o BPE (Sennrich et al., 2015), un híbrido entre representaciones de nivel de carácter y nivel de palabra que se basa en unidades de subpalabras extraídas mediante la realización de análisis estadístico del corpus de entrenamiento. Estos cambios llevaron a mejorar los resultados de BERT en diferentes tareas de PLN, como la comprensión del lenguaje natural y la búsqueda de respuestas.

4

Búsqueda e integración de datos tabulares

Esta sección describe el sistema de búsqueda e integración de datos tabulares desarrollado. Este sistema, como se dijo, se basa en el uso de *word embeddings* para obtener una representación vectorial del contenido de las tablas y poder calcular la similitud entre ellas.

En primer lugar, vamos a describir el sistema desarrollado para la búsqueda e integración de datos tabulares. Posteriormente, mostraremos los resultados de la evaluación, tanto de la fase de búsqueda como de la fase de integración mediante unión y combinación. Se analizará también el impacto que la información de contexto tiene en el rendimiento de los modelos contextuales. Finalizaremos el capítulo proporcionando una discusión sobre los resultados experimentales obtenidos y las posteriores conclusiones.

4.1. Descripción del sistema

Aunque los desafíos de la integración de datos se han investigado durante años con un progreso significativo (Miller, 2018), solo se han hecho esfuerzos para resolver problemas específicos. Sin embargo, según Abadi et al. (2022), se requiere más trabajo para investigar cómo canalizar la integración de datos para cubrir todo el camino desde los datos sin procesar hasta el resultado deseado por el usuario final.

Esta sección describe formalmente nuestra aproximación a la búsqueda e integración de datos tabulares usando *word embeddings*. Para la fase de integración, los operadores considerados en esta tesis son los de unión (*union*) y combinación (*join*) del álgebra relacional. En aras de la simplicidad, en esta tesis estos operadores son los mismos que se usan en SQL, una conocida implementación del álgebra relacional y un estándar reconocido para consultar y manejar datos tabulares:

- El operador de unión se denota mediante el símbolo $\cup$ en álgebra relacional. Dados dos conjuntos de datos tabulares (A y B), el operador de unión tiene como objetivo obtener un conjunto de datos único que contiene filas que están en A o en B (denotadas como A $\cup$ B). A y B deben tener las mismas columnas (número, orden y tipo de datos)

para ser calculados. Además, cada columna de cada conjunto de datos debe hacer referencia al mismo concepto para que sea significativo.

- El operador de combinación se denota por el símbolo ⋈ en álgebra relacional. Dados dos conjuntos de datos tabulares A y B, el operador de combinación tiene como objetivo obtener un conjunto de datos único que incluya todas las columnas de A y B (denotadas como A ⋈ B) y contiene filas que cumplen una condición de coincidencia (aplicada a los valores de algunas columnas).

La Figura 4.1 resume el flujo de trabajo y los componentes del sistema propuesto. Para determinar las tablas más relevantes para una búsqueda, así como comprobar si se pueden llevar a cabo las operaciones de unión y combinación, hemos establecido un mecanismo basado en *word embeddings* para calcular la similitud semántica entre conjuntos de datos tabulares. Como se ha mencionado con anterioridad, el uso de *word embeddings* resuelve los problemas de los enfoques léxicos basados en similitud de cadenas: términos como "ciudad" y "ubicación" podrían considerarse muy diferentes en términos de coincidencia de cadenas, pero en el espacio vectorial creado por los *word embeddings* estos dos términos pueden estar estrechamente relacionados y considerados como muy similares.

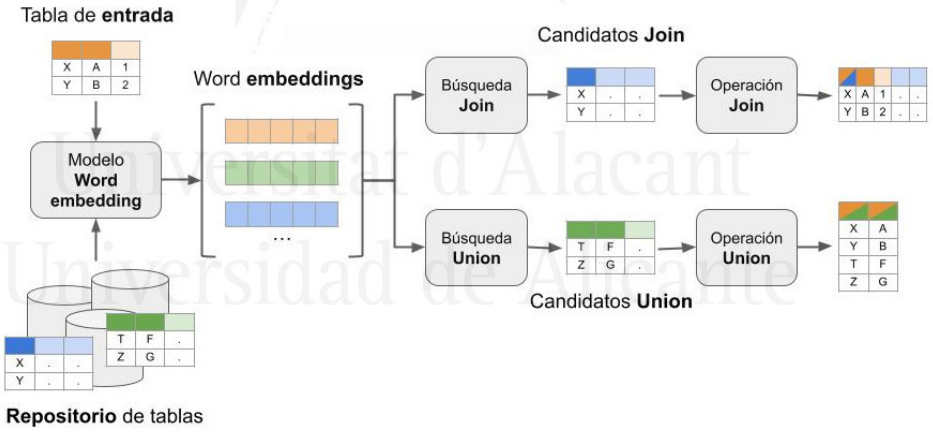


Figura 4.1: Componentes y flujo de trabajo para las operaciones de unión y combinación.

Este mecanismo de similitud toma como entrada un conjunto de datos tabulares y compara entre ellas todas las columnas de las tablas, obteniendo un valor de similitud para cada par de columnas pertenecientes a tablas diferentes. Para calcular esta similitud se tienen en cuenta dos tipos de elementos: el nombre de las columnas (encabezados) y el contenido de cada fila (valores de las celdas).

El primer paso consiste en preprocesar la información para normalizarla. Para ello se dividen las palabras en formato *CamelCase* (e.g. "VentasAnuales" se transformaría en "Ventas" y "Anuales") y aquellas separadas por guiones. Ambos fenómenos son muy comunes en los nombres de las columnas. También se eliminan los signos de puntuación y se convierte todo el texto a minúsculas. Después de eso, el modelo de lenguaje se usa para generar dos *word embeddings* para cada columna: uno se genera a partir del nombre de las columnas y el otro a partir del contenido de sus celdas. En aquellas situaciones en las que el nombre de la columna incluye más de una palabra, los vectores que representan cada palabra se promedian para obtener un único vector. El promedio de los *word embeddings* es uno de los métodos más populares para combinar este tipo de vectores, superando a otras técnicas más complejas, especialmente en escenarios generales (Gupta et al., 2020a). La misma estrategia se aplica al contenido de las celdas, donde el vector final es el resultado de calcular la media entre los vectores de cada una de las palabras que contienen.

Como en trabajos anteriores, usamos la similitud del coseno para calcular la distancia entre los vectores en el espacio generado por los *word embeddings* (Mikolov et al., 2013c):

$$sim(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|} = \frac{\sum_{i=1}^{n} \mathbf{v}_{1i} \mathbf{v}_{2i}}{\sqrt{\sum_{i=1}^{n} (\mathbf{v}_{1i})^2} \sqrt{\sum_{i=1}^{n} (\mathbf{v}_{2i})^2}}, \tag{4.1}$$

donde $\mathbf{v}_1$ y $\mathbf{v}_2$ representan el vector de incrustación de palabras del nombre de las columnas o el contenido de la columna para cada fila, y $sim(\mathbf{v}_1, \mathbf{v}_2)$ es un valor real en el rango $[-1, 1]$, donde -1 significa que no hay similitud y 1 significa máxima similitud entre los vectores considerados. Es posible que haya valores negativos del coseno, ya que los *word embeddings* pueden contener elementos negativos. No obstante, en todos los experimentos realizados, la similitud obtenida fue siempre un valor positivo. Este resultado concuerda con investigaciones anteriores que indican que los *word embeddings* no están equilibrados en torno al origen en el espacio semántico, mostrando menos similitudes del coseno negativas de las que se podría esperar de puntos aleatorios en una n-esfera (Mu and Viswanath, 2018).

En el caso de los modelos no contextuales, si este no proporciona cobertura para el nombre de la columna o su contenido (es decir, las palabras no están en el vocabulario del modelo), se utiliza la distancia de Levenshtein (Levenshtein et al., 1966) como estrategia de respaldo para garantizar que el sistema siempre devuelve un valor de similitud entre columnas. Esta métrica de cadenas se basa en el número de ediciones de un solo carácter (inserciones, eliminaciones o sustituciones) necesarias para cambiar una cadena por otra. Aplicamos la distancia de edición normalizada para obtener valores en el rango [0, 1], calculado como $(longitud - distancia)/longitud$, donde la

distancia se obtiene aplicando Levenshtein y longitud es la suma de las longitudes de las dos cadenas a comparar.

Por cada par de columnas comparadas, obtenemos un valor de similitud del nombre de la columna y un valor de similitud de su contenido. Para obtener una única puntuación final de similitud $simC$ de dos columnas c_1 y c_2, realizamos una combinación lineal de estos dos valores:

$$simC(c_1, c_2) = \alpha \cdot sim(\mathbf{c}_{1n}, \mathbf{c}_{2n}) + (1 - \alpha) \cdot sim(\mathbf{c}_{1c}, \mathbf{c}_{2c}), \tag{4.2}$$

dónde $sim(\mathbf{c}_{1n}, \mathbf{c}_{2n})$ es la similitud del coseno de los nombres de las columnas, $sim(\mathbf{c}_{1c}, \mathbf{c}_{2c})$ es la similitud del coseno de sus contenidos, y α es un parámetro en el rango $[0, 1]$ que pondera la relevancia de las dos puntuaciones de similitud en el resultado final. Un valor $\alpha = 0$ implica que solo se usa la información de las celdas, mientras que con un valor $\alpha = 1$ solo se usa el nombre de las columnas.

Para la tarea de búsqueda de tablas, se han definido dos objetivos diferentes en función de si la finalidad posterior es unir las tablas o combinarlas, es decir, dependerá de cuál de estas dos operaciones de integración queramos hacer tras el proceso de búsqueda. En la evaluación propuesta en este capítulo, la tarea de encontrar las mejores tablas para las operaciones de unión y combinación se asimila a la tarea de la búsqueda de tablas *ad hoc*: dada una consulta de búsqueda, devolver una lista ordenada de tablas relacionadas (Zhang et al., 2019), donde la consulta de búsqueda no es una secuencia de palabras clave sino una tabla (Sarma et al., 2012). Por tanto, es necesario definir una función de ordenación (*ranking function*) que clasifique las tablas en función del objetivo (unión o combinación), de modo que los resultados más relevantes aparezcan al principio de la lista de tablas recuperada.

Si el objetivo es recuperar tablas para la operación de unión, el criterio de ordenación establecido intenta priorizar tablas con un gran número de columnas muy similares que son, por tanto, buenas candidatas para la unión. Así, dado un conjunto de tablas T, la similitud $simTU$ para cada par $t_1, t_2 \in T$ se calcula como:

$$simTU(t_1, t_2) = \frac{\sum_{i=1,j=1}^{i \leq n, j \leq m} simC(c_{1i}, c_{2j})}{\|C_1\|\|C_2\|}, \tag{4.3}$$

donde $C_1 = \{c_{11}, c_{12}..., c_{1n}\}$ y $C_2 = \{c_{21}, c_{22}..., c_{2m}\}$ son el conjunto de columnas de t_1 y t_2 respectivamente. Por lo tanto, la similitud entre dos tablas se calcula como la similitud promedio entre sus columnas ($simC$).

Del mismo modo, es necesario definir una función de ordenación para recuperar datos tabulares para la operación de combinación. En este caso, la similitud $simTJ$ entre dos tablas t_1 y t_2 se calcula como:

$$simTJ(t_1, t_2) = \max_{i \leq n, j \leq m} \{simC(c_{1i}, c_{2j})\}. \tag{4.4}$$

Esta fórmula pondera la similitud entre dos tablas en función del par de columnas con mayor similitud $simC$, ya que en las operaciones de combinación nos interesan las tablas que contienen columnas que pueden coincidir con una alta probabilidad (es decir, columnas clave para realizar la combinación) mientras que las columnas restantes pueden no coincidir.

En el proceso de integración, una vez que se han recuperado las tablas relevantes el sistema tiene que indicar qué columnas específicas son candidatas para ser combinadas usando las operaciones de unión o combinación. Para decidir si se puede aplicar la operación de unión a dos columnas, se establece un umbral de similitud a partir del cual se considera que se puede realizar la operación. Por lo tanto, la unión se puede aplicar a cada par de columnas por encima de este umbral. Del mismo modo, la operación de combinación se realiza estableciendo un umbral para decidir si la operación se puede aplicar en dos columnas. En la sección de evaluación se analiza el impacto de estos umbrales en el rendimiento del sistema.

4.2. Evaluación

En esta sección se describe la evaluación realizada, que cubre los siguientes aspectos de los *word embeddings* contextuales:

- Su desempeño en la tarea de recuperar tablas relevantes para las operaciones de unión y combinación (Sección 4.2.1)
- Su desempeño en la identificación de columnas de dos tablas que se pueden combinar en operaciones de unión (Sección 4.2.2)
- Igual que antes, pero con respecto a las operaciones de combinación (Sección 4.2.3)
- Un estudio sobre cómo el contexto afecta el desempeño de los modelos contextuales (Sección 4.2.4)

El conjunto de datos utilizado en estos experimentos fue desarrollado por Nargesian et al. (2018) y está disponible públicamente.[1] Originalmente estaba destinado a la unión de tablas, pero en los siguientes experimentos se ha adaptado para evaluar también la operación de combinación. Este conjunto de datos consta de más de 5.000 tablas en formato CSV extraído de portales de datos abiertos de Estados Unidos, Canadá y el Reino Unido, proporcionando un etiquetado que identifica qué columnas de una tabla coinciden con las columnas de otra tabla. El conjunto de datos se creó a

[1] https://github.com/RJMillerLab/table-union-search-benchmark.

partir de 32 tablas base alineadas manualmente para identificar columnas coincidentes. El conjunto final se creó emitiendo primero una proyección sobre un subconjunto aleatorio de columnas de una tabla base y luego una selección sobre la tabla proyectada.

Estas tablas contienen los encabezados de las columnas y los valores de celda correspondientes, que comprenden valores de texto, numéricos y de fecha. Aunque los modelos de incrustación de palabras son especialmente adecuados para datos textuales, también brindan cobertura para columnas que contienen otros tipos de datos. En primer lugar, los nombres de las columnas son datos textuales, incluso si su contenido son números o fechas. Por lo tanto, el sistema siempre puede devolver un valor de similitud basado en los nombres de las columnas. En segundo lugar, incluso si solo se considera el contenido de las columnas, los modelos de incrustación de palabras aún brindan cobertura para muchos valores numéricos que están representados en el espacio vectorial formado. Por ejemplo, el modelo fastText proporciona cobertura para el 99,90 % de los números que van del 0 a 10.000. Esto implica que cualquier valor numérico utilizado para representar días, meses o años tiene una representación vectorial en este modelo. Además, en el caso de los modelos contextuales, el uso de algoritmos de segmentación de subpalabras BPE y WordPiece siempre proporciona una representación vectorial para cualquier valor numérico, o de cualquier otro tipo, que se encuentre en el texto.

Para realizar los experimentos se seleccionó al azar un subconjunto de 1.000 tablas. Cada tabla de este subconjunto se utilizó como una consulta para el sistema y se comparó con todas las demás tablas. Esto llevó a un total de 499.500 pares de tablas. Para cada par de tablas, todas sus columnas se compararon entre sí, obteniendo 3.249.440 comparaciones de columnas, un promedio de 6,5 comparaciones para cada par de tablas.

Los experimentos llevados a cabo en los siguientes apartados prueban el desempeño de dos modelos de incrustación de palabras no contextuales previamente entrenados (Word2vec y fastText), dos modelos de incrustación de palabras contextuales previamente entrenados (BERT y RoBERTa) y un modelo contextual entrenado específicamente (*fine-tuned*) para la tarea (WikiTables):

- Word2vec: modelo previamente entrenado en parte del conjunto de datos de Google News, que comprende alrededor de cien mil millones de palabras (Mikolov et al., 2013c). El modelo contiene vectores de 300 dimensiones para 3 millones de palabras y frases, aunque en los experimentos presentados aquí solo se consideraron palabras.

- fastText: modelo entrenado en el contenido de Wikipedia de 2017, el corpus web de UMBC y el conjunto de datos de noticias statmt.org, que comprende alrededor de 16 mil millones de palabras (Bojanowski

et al., 2017a). Como en el caso anterior, los vectores de palabra tienen 300 dimensiones.

- BERT: la versión del modelo evaluado es BERT-base, que contiene 12 capas (bloques transformadores), 12 cabezas de atención y 110 millones de parámetros. Los vectores resultantes están compuestos por 768 dimensiones (Devlin et al., 2018).

- RoBERTa: la versión evaluada es RoBERTa-base, que contiene 12 capas, 12 cabezas de atención, 125 millones de parámetros y vectores de salida de 768 dimensiones (Liu et al., 2019).

- WikiTables: modelo específico de la tarea obtenido al ajustar el modelo BERT en el corpus de tablas de Wikipedia, que contiene 1,6 millones de tablas relacionales de esta web (Bhagavatula et al., 2015a).

Como se mencionó anteriormente, el modelo BERT previamente entrenado se puede ajustar con solo una capa de salida adicional para crear modelos para una amplia gama de tareas de PLN sin modificaciones sustanciales de la arquitectura específica de la tarea. Para construir el modelo WikiTables, el corpus de las tablas de Wikipedia se preprocesó siguiendo el procedimiento mencionado anteriormente, separando palabras en formato *CamelCase* y con guiones, eliminando signos de puntuación y convirtiendo el texto a minúsculas. Para cada tabla en este corpus, se extrajeron todos los nombres de columna y se trataron como un documento de entrada para ajustar el modelo BERT. Se creó un segundo modelo para los valores de celda. En este caso, el contenido de las columnas se consideró como un documento de entrada para entrenar el modelo. Por lo tanto, en el caso de WikiTables tenemos realmente dos modelos de *word embeddings* independientes para calcular la similitud entre los nombres de las columnas y los valores de las celdas. Como en BERT, los vectores resultantes para cada palabra constan de 768 dimensiones.

La implementación de Word2vec y fastText se llevó a cabo utilizando la librería Gensim.[2] Los modelos contextuales se implementaron utilizando la librería Transformers desarrollada por Huggingface.[3]

Además de estos modelos, también se evaluaron las versiones más grandes de BERT y RoBERTa. BERT-large y RoBERTa-large contienen 24 capas y 16 cabezas de atención, produciendo *word embeddings* de 1024 dimensiones. Sin embargo, los resultados obtenidos no mejoraron las cifras presentadas en este apartado y se han dejado fuera de los resultados de evaluación.

[2] https://radimrehurek.com/gensim/.
[3] https://github.com/huggingface/transformers/.

4.2.1. Búsqueda de tablas

Esta sección describe la evaluación de los modelos al recuperar las tablas más relevantes para una tabla dada. Como se describe en la Sección 4.1, se han definido dos objetivos diferentes dependiendo de si el objetivo tras la búsqueda es unir las tablas (Sección 4.2.1.1) o combinarlas (Sección 4.2.1.2).

La precisión de los modelos (es decir, la fracción de tablas relevantes del total de tablas devueltas) se midió para las k primeras tablas devueltas, estableciendo distintos valores de k. Más específicamente, se calcularon las medidas P@1, P@10 y P@50, correspondientes a la proporción de resultados relevantes obtenidos en la primera tabla recuperada, en las 10 mejores tablas y en las 50 mejores tablas, respectivamente. Estos umbrales están en consonancia con los sistemas de recuperación de información en la web, donde miles de documentos relevantes están disponibles pero ningún usuario está interesado en leerlos todos. Las 1.000 tablas de consulta seleccionadas para los experimentos se dividieron de manera aleatoria en 10 subconjuntos, llevando a cabo 10 ejecuciones para calcular la precisión media, la desviación estándar y la prueba t de Student bimuestral ($\rho < 0{,}01$) de los resultados obtenidos.

4.2.1.1. Búsqueda para unión

El objetivo de esta tarea es recuperar las tablas más adecuadas para realizar operaciones de unión para una tabla dada. El criterio de ordenación se describió en la Ecuación 4.3. En los experimentos presentados, una tabla devuelta se considera relevante si contiene al menos una columna que podría alinearse con otra columna de la tabla de consulta en una operación de unión, tal y como se etiquetó en el conjunto de datos de Nargesian et al. (2018). Según este criterio, de los 499.500 pares de tablas mencionados anteriormente, 22.824 se consideraron relevantes (4,6 % del total).

Además de los modelos descritos al principio de esta sección, se definió un sistema base (*baseline*) utilizando el algoritmo BM25 (Robertson et al., 1995), una función de ordenación clásica ampliamente utilizada por los motores de búsqueda para estimar la relevancia de los documentos para una búsqueda determinada. La implementación de este sistema base se realizó utilizando Apache Lucene.[4]

Cada modelo se evaluó utilizando diferentes valores de α (de 0 a 1 inclusive, en incrementos de 0,1), como se describe en la Ecuación 4.2, para analizar la influencia de los nombres de columna y los valores de las celdas en el desempeño del sistema. En el caso de BM25, se emplearon tres consultas diferentes en Apache Lucene para simular el experimento de búsqueda realizado con los *word embeddings*: una consulta que usa solo los valores de las celdas (equivalente a α=0,0), una consulta que usa solo

[4] https://lucene.apache.org.

nombres de las columnas (equivalente a α=1,0) y una consulta que considera ambos a partes iguales (equivalente a α=0,5).

La Tabla 4.1 muestra el valor P@10 para los dos modelos no contextuales (Word2vec y fastText) y los tres modelos contextuales (BERT, RoBERTa y WikiTables). En este experimento, BM25 obtuvo una precisión de 0,8245 ($\alpha = 0{,}0$), 0,8349 ($\alpha = 0{,}5$) y 0.8612 ($\alpha = 1{,}0$).

Los resultados son prometedores en cuanto al rendimiento de los modelos contextuales. La mejor precisión para un modelo no contextual se obtuvo con fastText (0,9276, $\alpha = 0{,}3$). Este resulta mejor en 7,71 % el mejor valor de BM25 (0,8612, $\alpha = 1{,}0$). Por otra parte, el mejor modelo contextual fue BERT, con una precisión de 0,9737 ($\alpha = 0{,}3$). Este resultado mejora un 5 % al mejor modelo no contextual y un 13 % al sistema base. Según la prueba t de Student, estas diferencias son estadísticamente significativas. El mejor resultado para RoBERTa fue 0,9363 ($\alpha = 0{,}1$) y el mejor para WikiTables fue 0,9691 ($\alpha = 0{,}5$). En este último, la diferencia con BERT no fue estadísticamente significativa. El rendimiento de WikiTables es el más estable con respecto a las variaciones del parámetro α, con una desviación estándar $\sigma = 0{,}0175$, seguido de BERT ($\sigma = 0{,}0195$). En el otro punto del espectro está RoBERTa, con una caída marcada de rendimiento en el rango $[0,3, 0,4]$ de α, con una desviación estándar $\sigma = 0{,}0585$.

Tabla 4.1: Media (M) y desviación estándar (*standard deviation* - SD) de P@10 para los modelos de *word embeddings* en la búsqueda para unión. El mejor resultado para cada valor de α se muestra en negrita. * indica una mejora estadísticamente significativa ($\rho < 0{,}01$) del modelo contextual con respecto al mejor modelo no contextual para el correspondiente valor de α.

α	Word2vec		fastText		BERT		RoBERTa		WikiTables	
	M	SD	M	SD	M	SD	M	SD	M	SD
0.0	0.8322	0.0249	0.8664	0.0192	0,9149*	0.0145	0.8212	0.0243	**0.9174***	0.0109
0.1	0.8895	0.0166	0.9146	0.0117	**0.9616***	0.0110	0,9363*	0.0157	0,9472*	0.0069
0.2	0.9055	0.0118	0.9249	0.0084	**0.9704***	0.0117	0.9277	0.0167	0,9618*	0.0048
0.3	0.9089	0.0089	0.9276	0.0074	**0.9737***	0.0122	0.9175	0.0212	0,9647*	0.0069
0.4	0.9084	0.0110	0.9249	0.0105	**0.9726***	0.0131	0.8623	0.0224	0,9684*	0.0084
0.5	0.9076	0.0111	0.9217	0.0106	**0.9705***	0.0110	0.8520	0.0213	0,9691*	0.0081
0.6	0.9035	0.0121	0.9155	0.0121	0,9657*	0.0118	0.8456	0.0196	**0.9695***	0.0099
0.7	0.8974	0.0143	0.9082	0.0144	0,9592*	0.0113	0.8301	0.0220	**0.9681***	0.0100
0.8	0.8927	0.0147	0.9032	0.0154	0,9529*	0.0123	0.8225	0.0197	**0.9640***	0.0109
0.9	0.8851	0.0149	0.8967	0.0150	0,9383*	0.0164	0.8130	0.0167	**0.9545***	0.0127
1.0	0.8266	0.0246	0.8393	0.0213	0,9235*	0.0188	0.7701	0.0177	**0.9248***	0.0120

La medida P@1 se centra en la precisión del sistema al devolver una tabla relevante en la primera posición de la clasificación. La Figura 4.2 muestra que los tres modelos contextuales superaron a los modelos no contextuales y la sistema base, siendo estos resultados estadísticamente significativos. El modelo de mejor desempeño fue WikiTables (0,9965, $\alpha = 0{,}4$), mostrando

una alta fiabilidad en esta tarea. Este modelo mejora en un 12 % y un 23 % los mejores resultados obtenidos por fastText (0,8888, $\alpha = 0{,}4$) y Word2vec (0,8129, $\alpha = 0{,}4$) respectivamente.

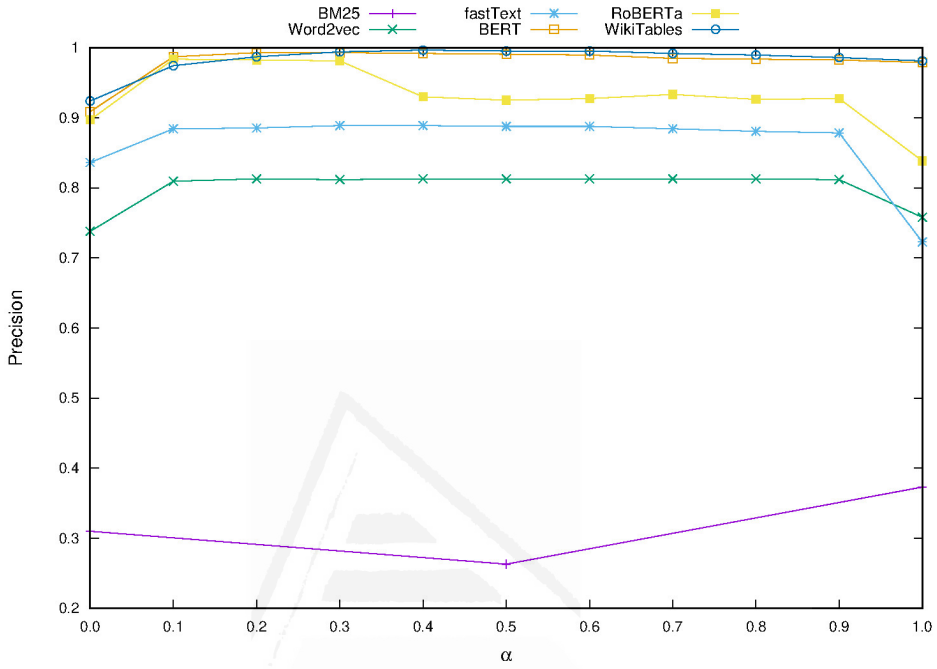


Figura 4.2: P@1 para BM25 y los modelos de *word embeddings* en la búsqueda para unión.

La Figura 4.3 muestra los resultados para P@50. El mejor modelo fue WikiTables (0,7230, $\alpha = 0{,}6$), seguido de BERT (0,7139, $\alpha = 0{,}2$), aunque esta diferencia no es estadísticamente significativa. La diferencia con fastText (0,6932, $\alpha = 0{,}4$), el mejor modelo no contextual, es estadísticamente significativa.

El rendimiento logrado fue sistemáticamente menor para cada modelo que el obtenido con P@10 como se esperaba, ya que aumentar el número de tablas recuperadas también aumenta la probabilidad de incluir falsos positivos. El desempeño de BM25 (0,6874, $\alpha = 1{,}0$) es notable en este experimento, logrando mejores resultados que los modelos no contextuales y cercano a los contextuales.

Para investigar más a fondo los valores de similitud de cada modelo, se calculó la media de similitud entre tablas asignadas por estos. La Figura 4.4 muestra que RoBERTa asigna puntuaciones de similitud significativamente altas, con un promedio de 0,9554 para todos los valores de α. En el otro extremo está Word2vec, con una similitud media de 0,3259.

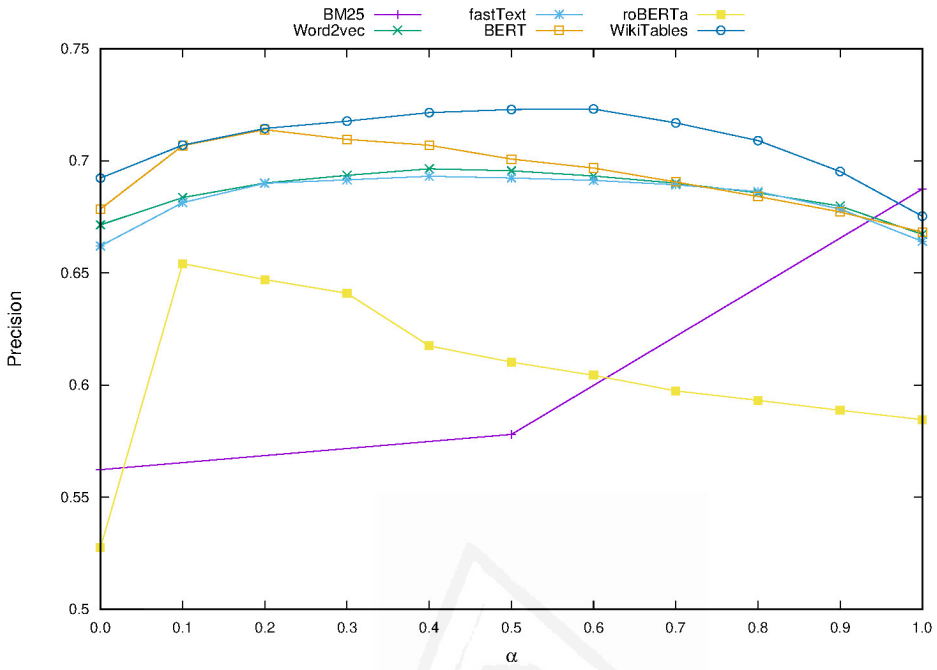


Figura 4.3: P@50 para BM25 y los modelos de *word embeddings* en la búsqueda para unión.

Este resultado también revela que la similitud media es mayor en promedio para valores más bajos de α, es decir, cuando se da más peso a los valores de celda que a los encabezados de columna. Esto implica que calcular los vectores de incrustación de muchas palabras diferentes (es decir, todos los valores de una columna) aumenta la similitud promedio entre tablas, aunque no igualmente para todos los modelos. El modelo con mayor variabilidad es BERT, con una diferencia de 0,2858 puntos entre la media más alta (0,8030, $\alpha = 0{,}0$) y la más baja (0,5172, $\alpha = 1{,}0$). El modelo más estable es WikiTables, con una diferencia de 0,0310 puntos entre la media más alta y la más baja.

La Figura 4.5 muestra con más detalle la tendencia central y la dispersión de los valores de similitud asignados por BERT. Los diagramas de caja indican que aumentar α también aumenta la dispersión, como muestra la diferencia entre los valores máximo y mínimo (los valores atípicos se excluyen en la figura). Por lo tanto, ponderar los valores de las celdas sobre los encabezados de las columnas no solo aumenta la puntuación de similitud, sino también reduce la variabilidad. Intuitivamente, esto puede llevar a BERT a una pérdida de rendimiento para valores bajos de α, ya que muchas de las tablas obtienen puntuaciones de similitud cercanas (como lo refleja la baja variabilidad). Sin embargo, el desempeño de BERT

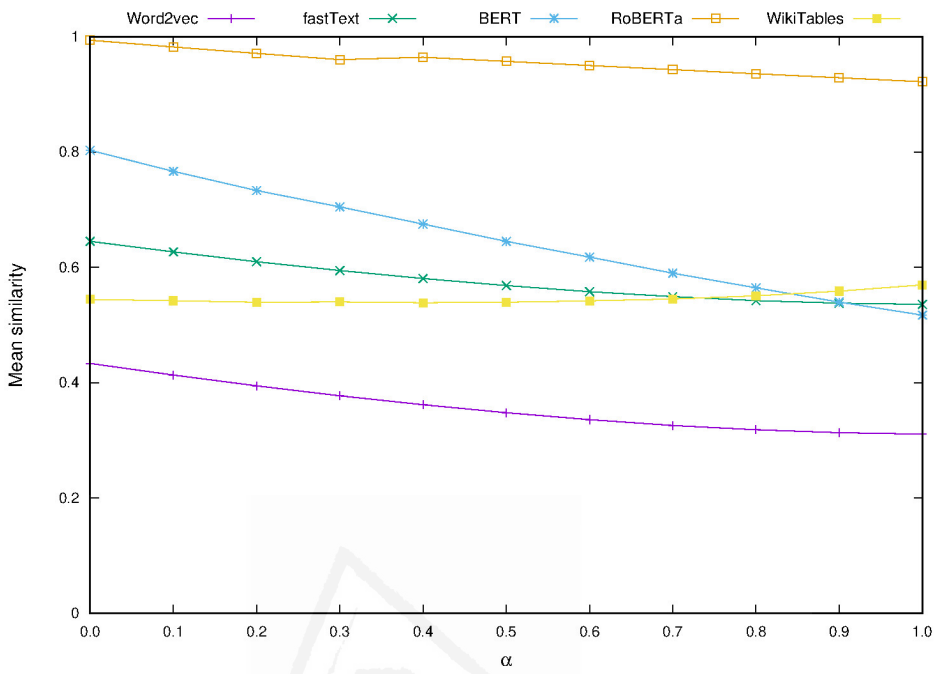


Figura 4.4: Similitud media entre tablas obtenida por los distintos modelos de incrustación de palabras.

en la Tabla 4.1, la Figura 4.2 y la Figura 4.3 aparentemente no muestra correlación con la similitud media en la Figura 4.4 basada en α. En el caso de P@10, el coeficiente de correlación de Spearman r_s entre rendimiento y similitud promedio es 0,3, lo que puede considerarse como un valor débil de correlación.

4.2.1.2. Búsqueda para combinación

El objetivo de este experimento es recuperar las tablas más relevantes para realizar operaciones de combinación (*join*). La función de ordenación para esta se definió en la Ecuación 4.4.

El conjunto de datos utilizado en el experimento de búsqueda para unión solo identifica en qué columnas se puede aplicar la operación de unión. La forma en que se obtuvo el conjunto de datos anterior se ha aprovechado aquí para evaluar también los modelos en la búsqueda para combinación. Como se mencionó anteriormente, las tablas se obtuvieron por proyección y selección de 32 tablas originales alineadas manualmente. Sobre esta base, el criterio para identificar si dos tablas del conjunto de datos se pueden combinar es verificar si ambas se obtuvieron de la misma tabla original (una

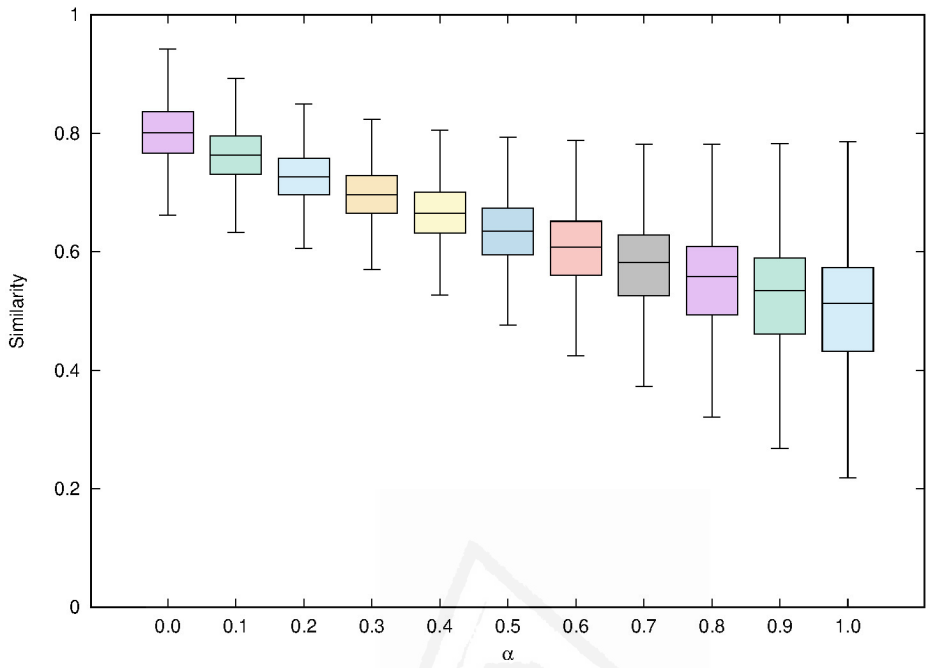


Figura 4.5: Tendencia central y dispersión de los valores de similitud obtenidos por BERT.

de las 32 mencionadas anteriormente) y si tienen al menos una columna en común con el mismo nombre.

El cumplimiento de estas condiciones garantiza que esa columna pueda servir para combinar las tablas. Por otro lado, el conjunto de datos original identifica qué columnas se pueden emparejar entre tablas. Por lo tanto, dos tablas no se pueden combinar si no tienen ninguna columna en común basada en este etiquetado original. Para los pares de tablas que no cumplan con alguna de estas condiciones, no se puede garantizar si se pueden combinar o no, por lo que se descartaron en la evaluación. Según este criterio, de los 499.500 pares de tablas, 15.137 pueden combinarse, 475.651 no pueden combinarse y 8.712 son inciertos (por lo que se descartan en la evaluación).

Dado que los encabezados de las columnas son la base para determinar si dos columnas se pueden combinar, para evaluar los modelos sin sesgos se usó un valor $\alpha = 0{,}0$, evitando de esta manera el uso de encabezados como evidencia para realizar la operación de combinación.

La Tabla 4.2 muestra el desempeño de los modelos de incrustación de palabras y el sistema base con las mismas métricas definidas en los experimentos de búsqueda para unión: P@1, P@10 y P@50.

Los resultados siguen un patrón similar a los obtenidos en los experimentos de búsqueda para unión. BM25 volvió a tener un desempeño bajo en P@1

Tabla 4.2: Media (M) y desviación estándar (*standard deviation* - SD) de P@1, P@10, y P@50 para BM25 y los modelos de *word embeddings* (α = 0,0) en la búsqueda para combinación (*join*). El mejor resultado para cada umbral de corte se muestra en negrita. * indica una mejora estadísticamente significativa ($\rho < 0{,}01$) del modelo contextual con respecto al mejor modelo no contextual para la métrica correspondiente.

Modelo	P@1		P@10		P@50	
	M	SD	M	SD	M	SD
BM25	0.2959	0.0434	0.7623	0.0376	0.4873	0.0224
Word2vec	0.6141	0.0568	0.6320	0.0324	0.4908	0.0239
fastText	0.7656	0.0508	0.7051	0.0308	0.5431	0.0255
BERT	**0.8258***	0.0311	0.7663*	0.0357	0.5699	0.0239
RoBERTa	0.8144*	0.0317	0.7625*	0.0403	0.5579	0.0247
WikiTables	0.8250*	0.0335	**0.7711***	0.0364	**0.5893***	0.0266

(0,2959) y los mejores resultados los obtuvo BERT (0,8258), seguido de cerca por WikiTables, aunque la diferencia no es estadísticamente significativa. BERT mejora 8 % a fastText, 15 % a Word2vec y 180 % a BM25. Todas estas diferencias son estadísticamente significativas.

P@10 muestra WikiTables como el modelo de mejor desempeño (0,7711), aunque la diferencia con BERT no es estadísticamente significativa. El primero mejora fastText en un 9 % y Word2vec en un 22 %. El desempeño de BM25 es notable (0,7623), muy cercano al de los modelos contextuales, lo que no fue el caso en los experimentos de búsqueda para unión. Dado que el sistema base se basa en la similitud léxica, los buenos resultados reflejan que hay pares de tablas cuyos valores de celda se superponen en columnas que se consideraron candidatas para unirse. Este no es un resultado inesperado, ya que los criterios para determinar las tablas que se pueden unir no imponen que las columnas candidatas tengan que ser claves primarias. Sin embargo, las tablas del conjunto de datos que cumplen esta condición parecen limitadas, ya que el rendimiento de BM25 cae significativamente en P@50, lo que indica que la similitud léxica disminuye (menos contenido en común) a medida que aumenta el número de tablas devueltas.

WikiTables es el mejor modelo en P@50 (0,5893), seguido de BERT (sin diferencia estadísticamente significativa). WikiTables mejora un 9 % el mejor modelo no contextual y un 21 % al sistema base. La diferencia fue estadísticamente significativa.

Estos resultados muestran que, al igual que en la tarea de búsqueda para unión, los modelos contextuales superan significativamente a los modelos no contextuales y el sistema base BM25.

4.2.2. Operación de unión

Este experimento evalúa el rendimiento de los modelos en la identificación de columnas de dos tablas que se pueden combinar en operaciones de unión. Para ello se establece un umbral de similitud para decidir si la operación se puede llevar a cabo.

El conjunto de datos proporcionado por Nargesian et al. (2018) identifica, para cada par de tablas, a qué columnas se le puede aplicar la operación de unión. De los pares de tablas en las que se evaluó la búsqueda para unión, solo se consideraron aquellos pares con al menos una columna coincidente en el conjunto de datos. Esto resultó en 303,548 comparaciones entre columnas (muestras de test), de las cuales 132,622 (43.69 %) pueden combinarse a través de la operación de unión y 170,926 (56.31 %) no.

El rendimiento de los modelos se ha evaluado en términos de precisión, cobertura y F1-score (media armónica de precisión y cobertura) para diferentes umbrales de similitud, que van de 0,1 a 1,0 en pasos de 0,1. Cada modelo fue evaluado seleccionando el valor de mejor desempeño para P@10 en el experimento de búsqueda para unión (ver Tabla 4.1).

La Tabla 4.3 resume los resultados para los umbrales de mejor desempeño en términos de F1-score. La mejor precisión se logró con fastText (0,6386), seguido de WikiTables (la diferencia no fue estadísticamente significativa). El resultado puede explicarse por la mayor similitud que los modelos contextuales asignan en promedio a las columnas comparadas, como se muestra en la Figura 4.4. Por tanto, muchas de ellas se sitúan por encima del umbral y son consideradas candidatas a unión, produciendo más falsos positivos que los modelos no contextuales, con la consiguiente pérdida de precisión.

Tabla 4.3: Media (M) y desviación estándar (*standard deviation* - SD) de precisión, cobertura y F1-score para la operación de unión, teniendo en cuenta los mejores valores para α y el umbral. El mejor resultado para cada métrica se muestra en negrita. * indica una mejora estadísticamente significativa ($\rho < 0{,}01$) del modelo contextual con respecto al mejor modelo no contextual para la métrica correspondiente.

Modelo	α	Umbral	Precisión		Cobertura		F1-score	
			M	SD	M	SD	M	SD
Word2vec	0.3	0.8	0.6125	0.0028	0.8574	0.0042	0.7146	0.0039
fastText	0.3	0.9	**0.6386**	0.0042	0.8289	0.0036	0.7214	0.0042
BERT	0.3	0.9	0.6267	0.0076	0.7630	0.0035	0.6882	0.0045
RoBERTa	0.1	0.9	0.4346	0.0057	**0.9885***	0.0022	0.6038	0.0041
WikiTables	0.6	0.8	0.6284	0.0066	0.8730*	0.0050	**0.7308**	0.0023

RoBERTa fue el mejor modelo con diferencia en términos de cobertura (0,9885). Este modelo asigna en promedio la mayor similitud a cada par de

columnas y, por lo tanto, la mayoría de ellas estaban por encima del umbral. Esto también da como resultado una baja precisión (0,4346) y la peor puntuación F1-score de todos los modelos evaluados. WikiTables superó al mejor modelo no contextual (Word2vec) y la diferencia fue estadísticamente significativa.

En cuanto a F1-score, el mejor resultado lo obtuvieron WikiTables (0,7308), seguido de fastText (no estadísticamente significativo). El primero puede considerarse como el modelo más equilibrado para el funcionamiento de la operación de unión. En el otro extremo, BERT y RoBERTa obtuvieron un rendimiento más bajo que los modelos no contextuales.

4.2.3. Operación de combinación

En este experimento, el objetivo es evaluar la capacidad de los modelos para identificar pares de columnas de diferentes tablas que se pueden combinar. Nuevamente, se estableció un umbral para tomar esta decisión.

La evaluación se centra en tablas que se pueden combinar dados los criterios descritos en la Sección 4.2.1.2. Recuerde que había 15.137 pares de tablas que se podían combinar. Este subconjunto da como resultado 214.997 pares de columnas, donde 103.414 (48,1 %) se pueden combinar siguiendo los criterios (tienen el mismo nombre de columna) y 111.583 (51,9 %) no. El conjunto de datos está bien equilibrado en términos de la cantidad de muestras positivas (se pueden combinar) y negativas (no se pueden combinar).

Al igual que con el experimento de búsqueda para unión, los modelos se probaron usando $\alpha = 0,0$ para realizar una evaluación imparcial y no sesgada. Su rendimiento se midió en términos de precisión, cobertura y F1-score para diferentes umbrales, que van de 0,4 a 1,0 en pasos de 0,05. El límite inferior (0,4) se estableció teniendo en cuenta el mínimo de similitud media obtenida por cualquiera de los modelos (en este caso 0,4335 por Word2vec).

La Figura 4.6 muestra la puntuación F1-score para los cinco modelos analizados. El mejor modelo fue Word2vec (umbral = 0,9), 0,8013 F1-score (0,7159 de precisión, 0,9098 de cobertura), seguido de fastText (umbral = 0,95), 0,7943 F1-score (0,7166 de precisión, 0,8910 de cobertura). El mejor modelo contextual fue BERT (umbral = 0,9), puntuación F1-score de 0,7842 (0,6795 de precisión, 0,9273 de cobertura). La diferencia entre estos tres modelos no fue estadísticamente significativa.

4.2.4. Relevancia del contexto

Como se mencionó en secciones anteriores, BERT y RoBERTa usan representaciones de caracteres/subpalabras (usando los algoritmos WordPiece y BEP) que permiten manejar palabras desconocidas. De esta forma, las palabras menos comunes se dividen en dos o más tokens de subpalabra para

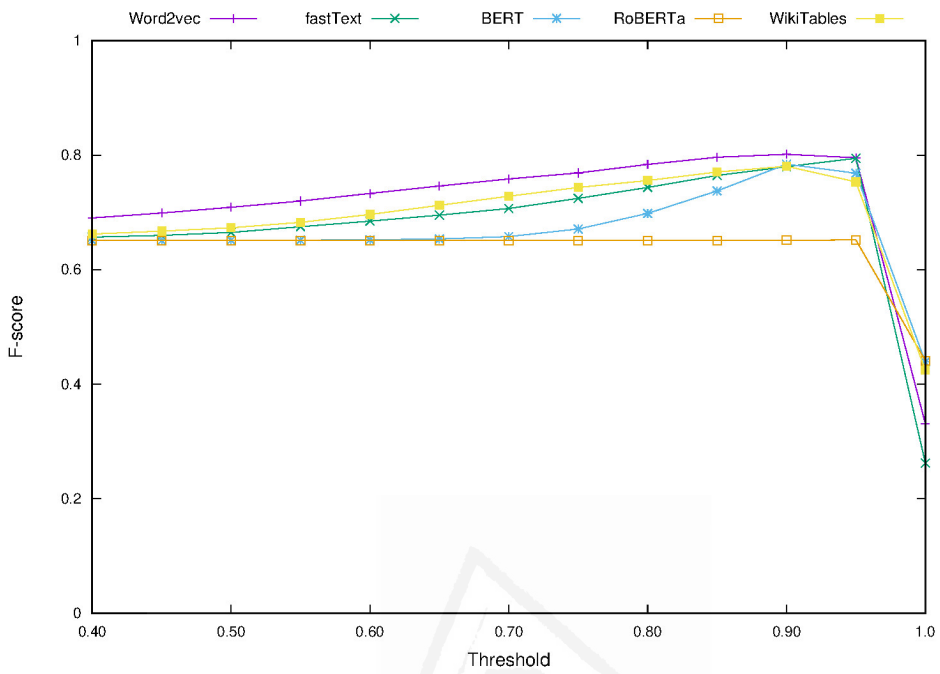


Figura 4.6: Puntuación F1-score para los cinco modelos de *word embeddings* en la operación de combinación con $\alpha = 0{,}0$.

proporcionar siempre una representación en el espacio vectorial semántico. Esta característica no está presente en Word2vec y fastText y podría ser una razón para el menor rendimiento de estos modelos con respecto a los contextuales.

Esta sección describe un estudio de ablación llevado a cabo para medir cómo la información del contexto está afectando realmente el desempeño de BERT, RoBERTa y WikiTables, dejando de lado el efecto que pueda tener el uso de los algoritmos WordPiece y BEP. El estudio se centra en los experimentos de búsqueda para unión y combinación, donde las diferencias entre los modelos contextuales y no contextuales fueron más evidentes.

En los experimentos anteriores, el vector que representa los encabezados de las columnas se obtuvo proporcionando como contexto el nombre de todas las demás columnas de la tabla. De manera similar, la representación vectorial de los valores de las celdas se obtuvo considerando también el contenido de las otras celdas de la columna.

En la evaluación que se presenta aquí, los experimentos de búsqueda para unión y combinación se recrearon utilizando una versión sin contexto de BERT, RoBERTa y Wikitables, donde se proporcionaron encabezados de columna y valores de celda al modelo sin ningún contexto adicional para obtener su representación vectorial.

La Figura 4.7 muestra la diferencia entre las versiones contextuales y no contextuales de BERT, RoBERTa y WikiTables para P@10 en los experimentos de búsqueda: (a) búsqueda para unión, $\alpha = 0{,}0$ (solo valores de celda), (b) búsqueda para unión, $\alpha = 1{,}0$ (solo encabezados de columnas), (c) búsqueda para combinación, $\alpha = 0{,}0$ (solo valores de celda) y (d) búsqueda para combinación, $\alpha = 1{,}0$ (solo encabezados de columnas).

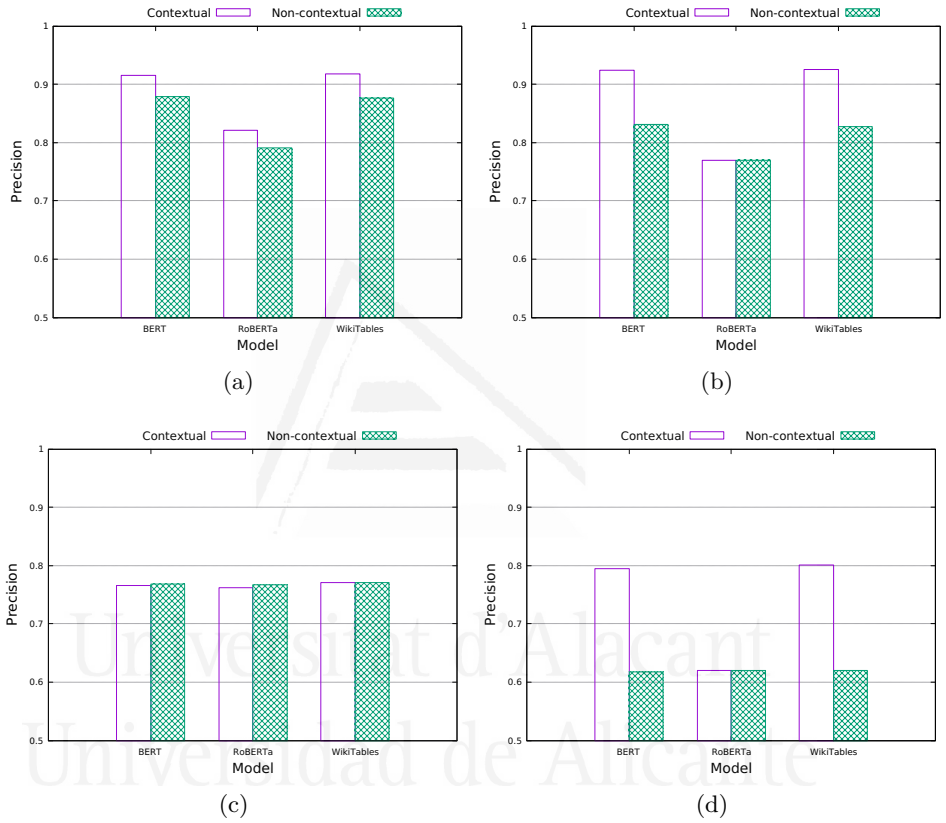


Figura 4.7: P@10 para BERT, RoBERTa y WikiTables: (a) búsqueda para unión, $\alpha = 0{,}0$, (b) búsqueda para unión, $\alpha = 1{,}0$, (c) búsqueda para combinación, $\alpha = 0{,}0$, y (d) búsqueda para combinación, $\alpha = 1{,}0$.

La Figura 4.7 (a) muestra que los tres modelos se benefician del contexto en la búsqueda para unión cuando solo se consideran los valores de celda ($\alpha = 0{,}0$). En el caso de BERT, la precisión cae de 0,9149 a 0,8785 (estadísticamente significativo) cuando se elimina la información de contexto. Esta diferencia es aún mayor cuando solo se consideran los encabezados de columna, como se muestra en la Figura 4.7 (b), donde P@10 cae de 0,9236 a 0,8313 (casi un 10 % de pérdida de desempeño). Un patrón similar se encuentra en WikiTables. Por el contrario, RoBERTa no aprovecha

el contexto en este caso y mejora menos que los otros modelos cuando se usan solo valores de celda.

En la búsqueda para combinación, la Figura 4.7 (c) refleja que cuando solo se tienen en cuenta los valores de las celdas, agregar contexto no mejora el rendimiento, lo que empeora ligeramente los resultados, aunque esta diferencia no es estadísticamente significativa. Recuerde que la función de ordenación para combinación solo tiene en cuenta el par de columnas con la mayor similitud. Esto la convierte en una medida más sensible que en el caso de la búsqueda para unión, donde se tuvieron en cuenta las similitudes de todas las columnas para la ordenación.

Para comprender mejor este resultado, se calculó la media y la desviación estándar de la similitud entre tablas obtenidas por la versión sin contexto de BERT, de manera similar a la Figura 4.5. El modelo BERT original obtuvo $\bar{x} = 0{,}8030$ y $\sigma = 0{,}0661$, mientras que la versión sin contexto logró $\bar{x} = 0{,}6919$ y $\sigma = 0{,}1261$. Tener una similitud media más baja y una desviación estándar más alta favorece el desempeño del sistema en esta tarea.

La conclusión es que aquí hay menos valores altos incorrectos (falsos positivos), lo que facilita la discriminación de columnas muy similares para la operación de combinación. El hecho de que el rendimiento del modelo con y sin contexto sea prácticamente el mismo sugiere que la pérdida que puede ocurrir al eliminar el contexto se ve compensada al tener una menor media y una mayor dispersión.

Esta hipótesis se corrobora al examinar los resultados para $\alpha = 1{,}0$ en la Figura 4.7 (d). En este caso, existen diferencias significativas en BERT y WikiTables con respecto a sus versiones no contextuales. La precisión de BERT cayó un 22 %, de 0,7946 a 0,6182. WikiTables obtuvo pérdidas similares, lo que demuestra el impacto del contexto en el rendimiento. Comparando de nuevo la media y la desviación estándar de la similitud entre tablas devuelta por BERT con y sin contexto, se observa que en el primer caso el modelo obtuvo $\bar{x} = 0{,}5292$ y $\sigma = 0{,}1539$, mientras que en el segundo caso fue $\bar{x} = 0{,}5591$ y $\sigma = 0{,}1266$. En esta situación, con $\bar{x}$ y σ muy similares, las diferencias en el rendimiento deben atribuirse al papel que juega el contexto, que resulta ser muy significativo en la búsqueda para combinación. Esta conclusión no se aplica a RoBERTa, ya que parece que el contexto no afecta en absoluto los resultados, como fue el caso de la búsqueda para unión.

4.3. Discusión

Este capítulo ha presentado el primer intento de utilizar *word embeddings* contextuales en todas las fases del proceso de búsqueda e integración de datos tabulares. La evaluación realizada ha analizado el desempeño de diferentes modelos de lenguaje e información contextual en cuatro tareas diferentes.

En la búsqueda para unión, los modelos contextuales superaron a los modelos no contextuales y a BM25. Los modelos contextuales demostraron ser extremadamente precisos para recuperar una tabla relevante en la primera posición del ranking (P@1). Curiosamente, BM25 se desempeñó mejor que Word2vec y fastText para P@50, y se acercó a los modelos contextuales cuando solo se consideraron los encabezados de columna. La conclusión que se puede extraer de este resultado es que, a medida que nos alejamos de las tablas mejor clasificadas, los modelos de *word embeddings* tienden a atribuir puntuaciones altas de similitud a tablas que no son realmente tan similares, aunque algunas de sus columnas puedan estar cerca en el espacio vectorial semántico. Mientras tanto, BM25 es más estricto en el cálculo de la similitud (considera solo la similitud léxica de los encabezados de columna), mejorando la precisión a expensas de la cobertura.

Este experimento reveló que la precisión de los modelos de lenguaje disminuyó cuando solo se consideraron los valores de celda o solo los encabezados de columna. Esto señala que ambas informaciones son complementarias y deben tenerse en cuenta, aunque sea en pequeña medida, ya que las diferencias también son significativas en $\alpha = 0{,}1$ y $\alpha = 0{,}9$.

BERT y WikiTables obtuvieron resultados similares en la búsqueda para unión, sin diferencias significativas entre ellos. Ambos funcionaron significativamente mejor que los modelos no contextuales, BM25 y RoBERTa. WikiTables fue el más estable en términos de rendimiento en función de α. Esto lo hace más adecuado para la tarea, ya que es más resistente a las variaciones en los parámetros del modelo.

En la búsqueda para combinación, BERT y WikiTables nuevamente se desempeñaron mejor en la tarea, sin diferencias significativas entre ellos, superando a los modelos no contextuales y al sistema base. Esta vez RoBERTa obtuvo resultados similares a los otros modelos contextuales. En general, el rendimiento fue menor que en la búsqueda para unión, pero siguen siendo competitivos para su uso en un proceso de integración.

Esta caída en el rendimiento puede explicarse parcialmente por la forma en que se obtuvo el conjunto de datos, lo que obliga a probar los modelos utilizando solo valores de celda para hacer una evaluación justa y sin sesgos. Como se indicó anteriormente, en la búsqueda para unión se encontró que la contribución tanto de los valores de celda como de los encabezados de columna fue fundamental para el rendimiento final. Por esta razón, se espera que evaluar con un conjunto de datos que no implique esta limitación podría conducir a mejoras en el rendimiento final de todos los modelos.

El mejor resultado en términos de F1-score para la operación de unión lo obtuvo WikiTables, superando a todos los demás modelos, pero la diferencia con fastText no fue estadísticamente significativa. El primero obtuvo una cobertura mejor pero menos precisión, aunque esta diferencia no fue significativa. Nuevamente, WikiTables podría considerarse la opción más equilibrada para esta tarea.

Los resultados de la operación de combinación revelan que los modelos contextuales no ofrecen una ventaja sobre los modelos no contextuales en esta tarea. Como se mencionó anteriormente, este resultado puede estar parcialmente condicionado por la configuración del experimento, ya que solo se tuvo en cuenta la información de los valores de las celdas. En el caso de BERT (ver Figura 4.5), esto generó una alta similitud en promedio y una baja dispersión, lo que dificulta identificar pares de columnas que se destaquen del resto, que es el objetivo final de esta tarea. RoBERTa fue el caso más extremo, ya que asignó los valores de similitud más altos en promedio (ver Figura 4.4). En la operación de unión, la cobertura de RoBERTa para el mejor ajuste de umbral (0,9) fue de 0,9885, a costa de una precisión baja (0,4346). Se obtuvieron resultados similares para la combinación: 0,9977 de cobertura y 0,4804 de precisión para un umbral de 0,95.

Esto implica que usar todos los valores de las celdas para construir el contexto agrega ruido y degrada el rendimiento, por lo que se debe hacer una selección más refinada de palabras (por ejemplo, evitar términos repetidos, eliminar palabras vacías, o mantener solo los términos más relevantes aplicando un esquema de ponderación como TF-IDF).

En la búsqueda para combinación, es interesante ver cómo el contexto no ayudó cuando se usaron solo valores de celda, pero ayudó significativamente cuando solo se consideraron los encabezados de columna. El análisis realizado mostró que en el caso de los valores de las celdas, agregar contexto aumentaba la similitud promedio entre tablas, bajando la dispersión, y esto afectaba el desempeño al neutralizar el efecto positivo de la información contextual. Una vez más, una selección detallada de palabras utilizadas en el contexto generaría un impacto positivo en esta tarea.

4.4. Conclusiones

La integración de datos ha sido un tema relevante en las ciencias de la información durante muchos años. Los avances actuales en el aprendizaje automático, y más concretamente en el área del aprendizaje profundo, han dado lugar a nuevas formas de abordar esta tarea.

En este capítulo se ha presentado un enfoque novedoso para la aplicación de *word embeddings* contextuales para la búsqueda e integración de datos. Estos modelos lingüísticos han alcanzado resultados punteros en muchas tareas de PLN en las que el foco de atención son los datos no estructurados, es decir, el texto en crudo. Sin embargo, en los últimos tiempos estos modelos se han utilizado también para comprender mejor los datos estructurados y más concretamente la información tabular.

La novedad de este trabajo ha sido la propuesta de una solución tanto para la operación de unión como para la de combinación basada en *word*

embeddings contextuales. El proceso consta de dos fases: en primer lugar, se recuperan las tablas más relevantes para la operación correspondiente; en segundo lugar, se identifican las columnas candidatas para la unión o la combinación.

Se han propuesto y analizado cuatro tareas diferentes: búsqueda para unión, búsqueda para combinación, operación de unión y operación de combinación. Cada tarea se ha evaluado utilizando tres modelos contextuales, dos modelos no contextuales y una función de clasificación clásica de referencia para las dos tareas de búsqueda. En estas cuatro tareas se ha estudiado el impacto en el rendimiento del uso de encabezados de columna y valores de celda. Los resultados revelaron que todos los modelos se beneficiaron de la combinación de ambos tipos de información, aunque los modelos obtuvieron mejores resultados cuando se dio más peso a los valores de las celdas.

En cuanto al rendimiento de los modelos contextuales, BERT y su versión refinada WikiTables mostraron más estabilidad y mejor rendimiento que RoBERTa a lo largo de todos los experimentos. Tanto en la búsqueda de tablas para unión como en la búsqueda de tablas para combinación, los modelos contextuales superaron significativamente a los no contextuales. BERT fue el mejor modelo en la búsqueda para unión (P@10 = 0,9737) y WikiTables en la búsqueda para combinación (P@10 = 0,7711), aunque la diferencia no fue estadísticamente significativa en este último caso.

WikiTables obtuvo los mejores resultados en el experimento con la operación de unión, logrando una mejora significativa con respecto a otros modelos contextuales. En esta tarea se obtuvieron resultados mixtos, con fastText y Word2vec superando a BERT y RoBERTa en términos de puntuación F1-score. Los resultados de la operación de combinación no revelaron diferencias significativas entre los modelos no contextuales y BERT. El criterio de seleccionar sólo la columna con mayor similitud en cada tabla para calcular las columnas candidatas no resultó eficaz. El rendimiento se ve muy afectado cuando los modelos asignan un valor de similitud alto de media con una dispersión baja, y este problema fue más frecuente en los modelos contextuales, como se muestra en la Figura 4.4.

5

El impacto de la eliminación de contenido

En el capítulo anterior se hizo una propuesta de sistema de búsqueda e integración de datos tabulares basado en modelos de lenguaje contextuales, utilizando representaciones de las tablas en forma de *word embeddings* para poder calcular su similitud. Estos modelos se han convertido en los sistemas más eficaces a la hora de trabajar en tareas de PLN, aunque en esta tesis el objetivo es su aplicación a datos estructurados.

Desafortunadamente, el uso de este tipo de modelos sobre grandes conjuntos de datos puede ser computacionalmente costoso, lo que puede comprometer el rendimiento de nuestro sistema de búsqueda e integración de datos tabulares en un escenario de producción.

Este capítulo plantea la hipótesis de que el contenido de una tabla puede reducirse (eliminando filas) sin alterar significativamente la representación vectorial (*word embedding*) de dicha tabla. Esto implica que se puede obtener la misma representación semántica de alta calidad utilizando solo una parte de la tabla original, reduciendo así el coste computacional del uso de modelos contextuales como BERT y otros Transformers.

Vamos a llevar a cabo un estudio para contrastar esta hipótesis inicial. Para ello se han analizado dos conjuntos de datos de diversa naturaleza. El primero es un conjunto de datos masivo que consta de casi 500.000 tablas procedentes de Wikipedia. Este tipo de tablas normalmente contiene un número reducido de filas y columnas. El segundo conjunto de datos incluye 100 tablas con cientos de miles de filas extraídas de un portal de datos abiertos.

5.1. Metodología de estudio

Como se mencionó anteriormente, el objetivo del estudio llevado a cabo en este capítulo es probar la hipótesis de que el contenido de las tablas se puede reducir sin alterar las representaciones de *word embeddings* obtenidas por los modelos de lenguaje para dichas tablas. Esta eliminación de contenido puede reducir significativamente el tiempo de cálculo y los recursos necesarios para aplicar estos modelos en grandes conjuntos de datos tabulares.

Se ha utilizado un enfoque básico para reducir tablas seleccionando aleatoriamente un porcentaje de las filas originales, obteniendo la representación de *word embedding* de estas subtablas reducidas y comparandolas con el *word embedding* de la tabla original. Una similitud alta implica que tanto la tabla original como la reducida están muy cerca en el espacio semántico generado por el modelo de lenguaje y que, por lo tanto, siguen siendo muy similares pese a la eliminación de contenido. Cuando estas representaciones de las subtablas se usen en el sistema de búsqueda e integración propuesta en el capítulo anterior, los resultados obtenidos serán muy similares a los que se obtendrían si usáramos los vectores de las tablas originales, manteniendo el rendimiento del sistema pero reduciendo notablemente el coste de procesamiento para crear los *word embeddings* al trabajar con cantidades de datos más pequeñas.

En el estudio llevado a cabo, a la hora de crear los *word embeddings* para las tablas solo se tuvo en cuenta el contenido de las celdas, que es el que se va a reducir y el que nos permite ver el efecto de esta reducción. Ni los nombres de las columnas ni ningún metadato adicional fue usado a la hora de construir estos *word embeddings*.

El procedimiento para comparar la tabla original con una subtabla consiste en obtener un *word embedding* para cada columna de estas dos tablas. Luego, cada columna en la tabla original se compara con la columna correspondiente de la subtabla calculando la similitud del coseno entre sus representaciones vectoriales. Esta comparación proporciona un valor en el rango $[0, 1]$, donde 1 implica máxima similitud y 0 ninguna similitud en absoluto. La similitud final entre ambas tablas se calcula como la similitud promedio entre sus columnas.

En este estudio se analizaron los siguientes aspectos:

- El rendimiento de modelos de lenguaje contextuales y no contextuales
- El efecto de reducir las tablas en función del tamaño (número de filas) de estas
- La cobertura de los modelos de lenguaje evaluados, es decir, el porcentaje de contenido de la tabla que tiene una representación vectorial en forma de *word embedding* en el espacio semántico definido por los distintos modelos evaluados (conocido como el *vocabulario* del problema)
- El impacto de los tipos de datos numéricos y textuales en la representación proporcionada por el *word embedding*

Con respecto a este último aspecto, los modelos de lenguaje han demostrado en gran medida sus capacidades para manejar información de cadenas, pero los estudios sobre su capacidad para representar información

numérica son más limitados. Investigaciones previas en el campo sugieren que estos modelos también capturan la semántica de los números y sus relaciones (Wallace et al., 2019; Zhang et al., 2020). En este capítulo se han realizado diferentes experimentos considerando todo el contenido de la tabla, solo columnas textuales y solo columnas numéricas, para contrastar así los resultados en función de la tipología de los datos.

5.2. Evaluación

En esta sección se describen los experimentos realizados para validar la hipótesis planteada: los modelos empleados, los conjuntos de datos recopilados y los resultados obtenidos.

5.2.1. Modelos

El estudio realizado incluye modelos de lenguaje tanto contextuales como no contextuales. El objetivo es comparar la capacidad de estos dos tipos de modelos para generar *word embeddings* de tablas que sean robustos a la reducción de contenido. Se analizaron tres modelos de lenguaje diferentes: Word2vec, fastText y SentenceBERT. Los dos primeros ya se usaron en la experimentación del capítulo anterior. A continuación se dan detalles sobre los tres modelos utilizados:

- Word2vec (Mikolov et al., 2013b): genera *word embeddings* no contextuales. El modelo proporciona vectores de 300 dimensiones para dar cobertura a tres millones de palabras y frases, aunque en los experimentos presentados aquí solo se consideraron palabras.

- fastText (Bojanowski et al., 2017b): genera *word embeddings* no contextuales. Como ocurría con Word2vec, los vectores representativos de las palabras tienen 300 dimensiones. Utilizamos aquí el modelo preentrenado que contiene información de subpalabras para aumentar la cobertura del vocabulario del modelo. Este modelo de subpalabra permite obtener una representación de palabras que no están incluídas en su vocabulario, pero que pueden componerse a partir de los vectores de las subpalabras que contiene. En este sentido actúa de manera similar a los algoritmos WordPiece y BCE que usan los modelos Transformers, permitiendo aumentar notablemente la cobertura del modelo.

- SentenceBERT (Reimers and Gurevych, 2019b): genera *word embeddings* contextuales mediante el uso de una arquitectura basada en redes siamesas y tripletas para derivar incrustaciones de oraciones semánticamente significativas. De esta manera es capaz de, dada una frase, generar un único *word embedding* que la representa, en

lugar de tener que combinar (promediando, por ejemplo) los vectores individuales de cada palabra para obtener la representación final de la oración. Este modelo fue entrenado en 1 millón de pares de oraciones. El modelo utilizado produce *word embeddings* de 384 dimensiones.[1]

Como ya se ha comentado, una diferencia notable entre estos dos tipos de modelos es el uso por parte de los modelos contextuales de unidades de subpalabras (aunque la variante de fastText que usamos en este caso usa también esta aproximación) en lugar de palabras completas para representar el vocabulario del problema. Las incrustaciones de palabras se basan en el conjunto específico de tokens disponibles en el corpus utilizado para crear los vectores. Cuando aparece una palabra fuera del vocabulario en un texto nuevo, los modelos basados en palabras no proporcionan representación para ella en el espacio semántico y, por tanto, el token se considera desconocido.

Para manejar los grandes vocabularios que se suelen dar en los corpus de lenguaje natural, SentenceBERT utiliza el algoritmo de segmentación de subpalabra WordPiece, al igual que hacía BERT. En él, el vocabulario se inicializa con caracteres individuales en el idioma y luego las combinaciones de símbolos más frecuentes en el vocabulario se agrega iterativamente al vocabulario. Por lo tanto, las subpalabras tienen su propia representación en el espacio semántico y a las palabras previamente desconocidas se les puede asignar una representación combinando los vectores de sus unidades de subpalabras subyacentes.

5.2.2. Conjuntos de datos

En este estudio se han utilizado dos conjuntos de datos para analizar diferentes escenarios. El primero consta de 492.090 tablas obtenidas de Wikipedia. Se trata de un subconjunto del corpus utilizado en el capítulo anterior, que fue originalmente recopilado por Bhagavatula et al. (2015b) y que consta de 1,6 millones de tablas de Wikipedia. En este corpus se descartaron las tablas con menos de 10 filas, ya que no resultan de utilidad para los experimentos de eliminación de contenido por tratarse de tablas muy pequeñas. Lo llamaremos el conjunto de datos *Wikitables* en los experimentos.

El segundo conjunto de datos contiene una muestra de 100 tablas del Chicago Data Portal,[2] la iniciativa de datos abiertos de esta ciudad que comprende conjuntos de datos para temas como construcción, educación, medio ambiente, eventos, seguridad pública, administración y finanzas. Lo llamamos el conjunto de datos *Chicago* en los experimentos. En este conjunto se filtraron las tablas con menos de 1.000 filas y más de 1.000.000. De esta manera proporcionamos un conjunto de datos cuantitativamente diferente al

[1] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.
[2] https://data.cityofchicago.org/.

de *Wikitables*. La Tabla 5.1 resume las principales características de ambos conjuntos de datos.

Tabla 5.1: Resumen estadístico de los conjutos de datos *Wikitables* y *Chicago*.

Característica	*Wikitables*	*Chicago*
Cantidad de tablas	492.090	100
Cantidad de filas	14.106.528	10.303.767
Cantidad de columnas	2.812.631	1.818
Cantidad de columnas numéricas	676.536	723
Cantidad promedio de filas	28,67	103.037,67
Cantidad promedio de columnas	5,85	18,18
Cantidad promedio de columnas numéricas	1,37	7,23
Máximo número de filas	4.670	975.010
Máximo número de columnas	237	119
Máximo número de columnas numéricas	115	70

Como se puede ver en las estadísticas, *Wikitables* incluye una gran cantidad de tablas de tamaño mediano, mientras que *Chicago* contiene una pequeña cantidad de tablas de tamaño grande o muy grande. El número medio de filas en *Wikitables* es 28,67, mientras que en *Chicago* este valor es 103.037,67. El número medio de columnas también es notablemente mayor en *Chicago*, 18,18 en comparación con 5,85 en el otro conjunto de datos.

5.2.3. Resultados

Para evaluar cómo cambia la representación mediante *word embeddings* de una tabla cuando se reduce su contenido, se han establecido una serie de umbrales en términos del porcentaje de filas seleccionadas aleatoriamente de la tabla original: 1 %, 5 % y múltiplos de 10 hasta 90 %, donde 1 % significa que el 99 % de las filas se eliminaron de la tabla original.

De esta manera, para cada tabla se obtuvieron 11 subtablas (del 1 % al 90 %) y se compararon con la original utilizando los tres modelos de lenguaje mencionados anteriormente: Word2vec, fastText y SentenceBERT. Como se describió en la Sección 5.2.2, las tablas con menos de 10 filas se descartaron en estos experimentos. En tablas con menos de 20 filas, no se calcularon las subtablas 1 % y 5 %, ya que no hay suficientes filas para crear estos subconjuntos. En tablas con menos de 100 filas, la subtabla del 1 % se descartó por la misma razón.

En cuanto a los modelos estudiados, en el caso de Word2vec y fastText la representación de incrustación de palabras de una columna se obtuvo promediando el vector de incrustación de palabras de cada palabra en esa columna, ya que estos modelos solo ofrecen representaciones a nivel

de palabra. Como ya se comentó en el capítulo anterior, promediar incrustaciones de palabras es uno de los métodos más populares para combinar *word embeddings*, superando técnicas más complejas especialmente en escenarios de dominio abierto (Gupta et al., 2020b). A diferencia de estos dos modelos, SentenceBERT proporciona incrustaciones de palabras también a nivel de oración con un límite de 256 palabras de entrada. Esto significa que, para tablas con 256 filas o menos (suponiendo un solo token en cada celda), el modelo proporciona automáticamente un solo vector que representa cada columna. En aquellas situaciones en las que el número de tokens superaba este límite, los tokens se procesaron en grupos de 256 tokens, obteniendo un vector de cada uno de estos grupos y finalmente promediando para obtener un único *word embedding* para la columna.

La Figura 5.1 muestra la similitud lograda (eje y) por los tres modelos probados en el corpus *Wikitables* cuando se seleccionaron diferentes porcentajes de filas (eje x). La línea discontinua recta en la parte superior corresponde al umbral de similitud del 90 %, que puede considerarse como un nivel de similitud alto.

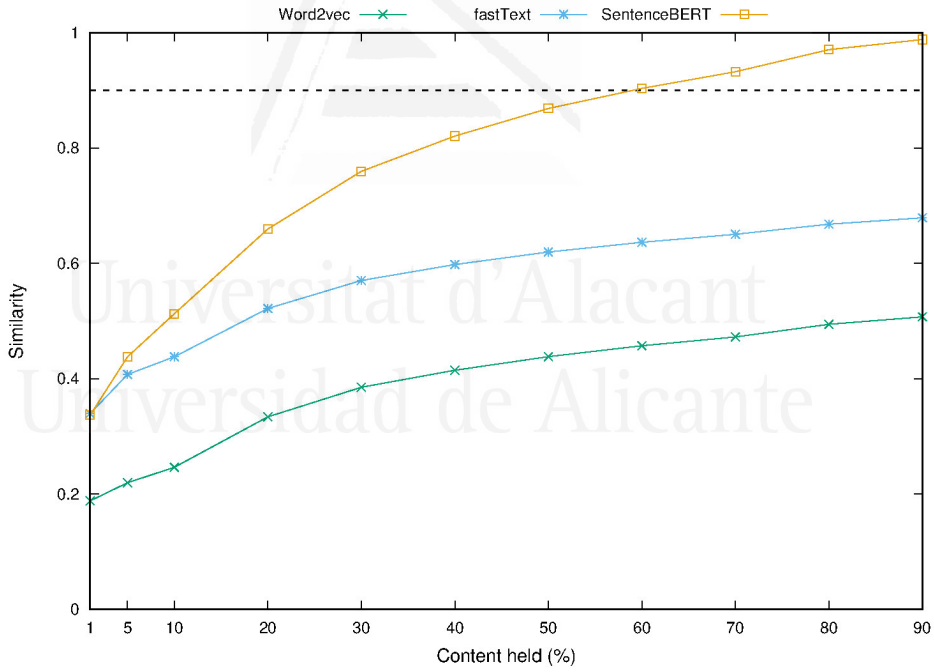


Figura 5.1: Similitud obtenida por Word2vec, fastText y SentenceBERT con diferentes porcentajes de filas en el corpus *Wikitables*. La línea discontinua en la parte superior representa el umbral de similitud del 90 %.

Los resultados muestran que SentenceBERT supera significativamente a los modelos no contextuales ($\rho < 0{,}01$ en todas las subtablas para el corpus

Chicago y en todas las subtablas por encima del 5 % en el corpus *Wikitables*), siendo los vectores de Word2vec los más afectados negativamente por la reducción de contenido. Utilizando SentenceBERT, la subtabla del 60 % superó el umbral de similitud del 90 % con respecto a la tabla original, lo que implica que se podría eliminar el 40 % de las filas y aún así obtener un *word embedding* representativo de la tabla que se pareciera un 90 % al original. Con el mismo número de filas, Word2vec obtuvo solo un 45 % de similitud y fastText un 64 %.

Se obtuvieron resultados similares en el conjunto de datos de *Chicago*, como se muestra en la Figura 5.2. En este caso, con tablas que contienen un número significativamente mayor de filas que en Wikitables, la representación de incrustación de palabras es menos sensible a la eliminación de filas. En el caso de SentenceBERT, mantener solo el 10 % de las filas permite obtener una incrustación de palabras que es un 90 % similar a la original. En este mismo escenario, Word2vec obtuvo un 44 % de similitud y fastText un 67 %.

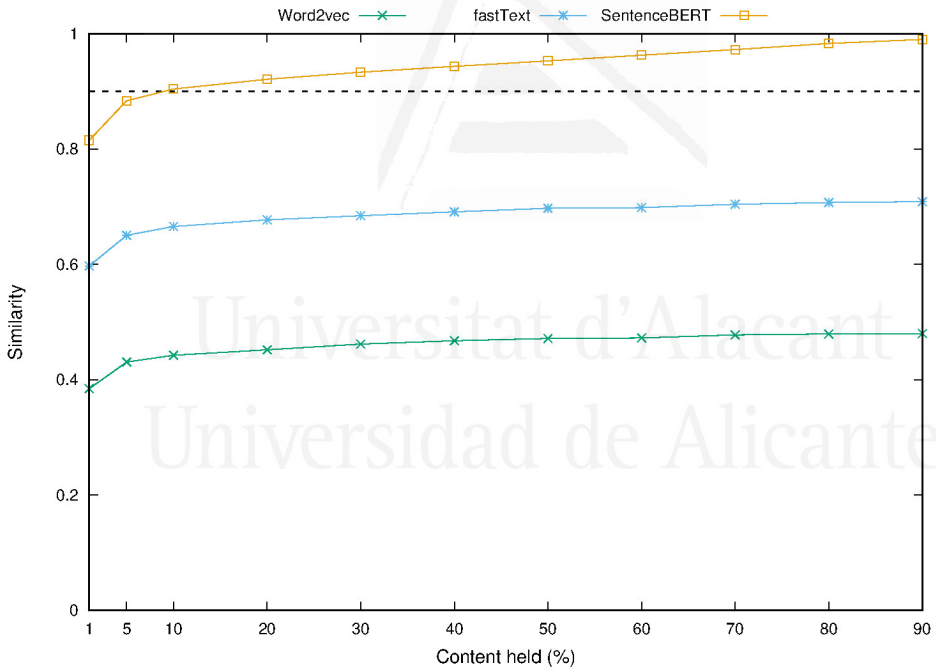


Figura 5.2: Similitud obtenida por Word2vec, fastText y SentenceBERT con diferentes porcentajes de filas en el corpus *Chicago*. La línea discontinua en la parte superior representa el umbral de similitud del 90 %.

La Figura 5.3 muestra la similitud lograda usando solo columnas numéricas en el corpus *Wikitables*, mientras que la Figura 5.4 muestra la similitud lograda en ese mismo corpus usando solo columnas textuales. Sobre

el corpus *Chicago* se obtuvieron resultados similares, que se omiten aquí por no aportar información relevante adicional.

Los resultados obtenidos en ambos casos por SentenceBERT son muy similares. En el caso de fastText, llama la atención la alta similitud obtenida usando solo valores numéricos, con una similitud cercana al 80 % para la subtabla del 20 % (esta similitud fue de solo el 48 % en la versión de solo cadena). Finalmente, Word2vec obtuvo mejores resultados en promedio usando valores de cadena.

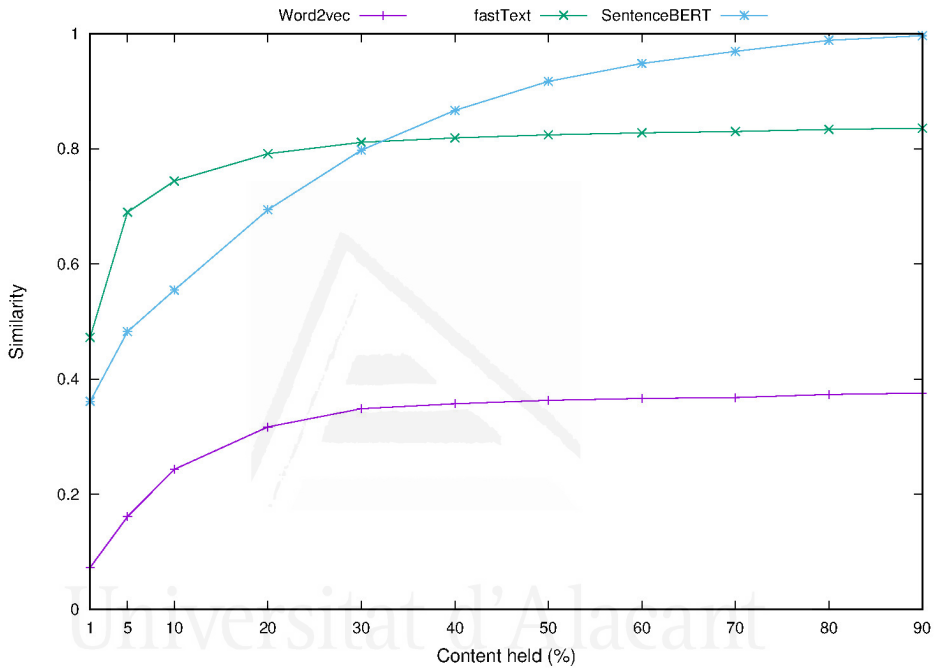


Figura 5.3: Similitud obtenida por Word2vec, fastText y SentenceBERT usando solo columnas numéricas en el corpus *Wikitables*.

Es importante tener en cuenta la cobertura dada por cada modelo, es decir, la cantidad de palabras en la tabla que forman parte del vocabulario del modelo y, por tanto, tienen una representación de incrustación de palabras en el modelo. Como SentenceBERT usa información de subpalabras, cada token tiene asegurada una representación como *word embedding*, lo que brinda una cobertura del 100 % para cada valor numérico y de cadena en los conjuntos de datos. En el caso de Word2vec, la cobertura fue del 53,24 % para columnas numéricas y del 56,13 % para columnas textuales. Esta limitada cobertura explica el bajo rendimiento alcanzado por el modelo con ambos tipos de datos. fastText, al usar su propio algoritmo de subpalabras, cubrió el 90,03 % de las columnas numéricas y el 66,05 % de las columnas textuales, mejorando la cobertura de Word2vec pero lejos del 100 % que consiguen los

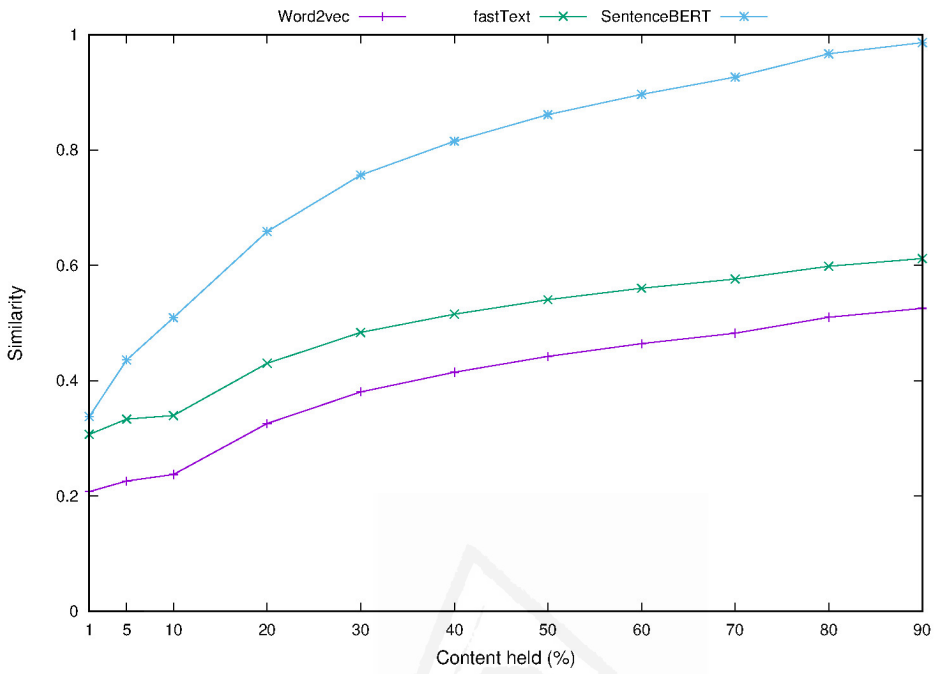


Figura 5.4: Similitud obtenida por Word2vec, fastText y SentenceBERT usando solo columnas textuales en el corpus *Wikitables*.

modelos contextuales gracias al uso de BPE y WordPiece. Estos valores de cobertura se correlacionan con el rendimiento que se muestra en la Figura 5.3 y la Figura 5.4, donde usar solo tokens numéricos en fastTet y SentenceBERT produjo resultados ($\rho < 0{,}01$) mucho mejores que usar solo tokens de texto.

5.3. Conclusiones

Este capítulo ha presentado estudio sobre cómo la eliminación de contenido afecta la representación mediante *word embeddings* de datos tabulares. El objetivo era demostrar que una parte significativa del contenido de una tabla puede eliminarse sin afectar a su representación en el espacio semántico proporcionado por las incrustaciones de palabras.

El estudio mostró que los modelos contextuales proporcionan *word embeddings* que son menos sensibles a la reducción de contenido. En el caso del conjunto de datos *Chicago* (un promedio de 103.037,67 filas por tabla), mantener solo el 10 % de las filas proporcionó una similitud promedio entre la tabla original y la reducida superior al 90 %. En el caso de *Wikitables* (29,67 filas por tabla), debe mantenerse el 60 % de las filas para lograr el mismo valor de similitud.

El impacto de la reducción de tablas fue menos significativo en tablas con muchas filas. Otros experimentos revelaron que el tipo de dato de las columnas, numérico o textual, afecta significativamente al rendimiento de Word2vec y fastText. SentenceBERT no ofreció una diferencia significativa entre ambos tipos de datos.

El análisis de la cobertura de cada modelo mostró una clara correlación con el rendimiento. Word2vec ofreció la peor cobertura y los peores valores de similitud. En el caso de fastText, hubo una mayor cobertura de números con respecto al texto, gracias a su propio algoritmo para trabajar a nivel de subpalabra, lo que se correlaciona con una similitud significativamente mayor utilizando solo valores numéricos.

Estos resultados proporcionan un camino prometedor para reducir los costos computacionales en la búsqueda e integración de tablas usando incrustaciones de palabras.

6

Desarrollo de un corpus de datos tabulares

En el Capítulo 2 vimos que existen diferentes aproximaciones que han tratado de abordar el problema de la búsqueda e integración de datos tabulares de manera automática mediante el uso de técnicas de inteligencia artificial (más concretamente, empleando algoritmos de aprendizaje automático). El problema que tienen estos algoritmos es la necesidad de grandes volúmenes de datos de entrenamiento para poder llevar a cabo de manera adecuada esta labor.

A diferencia de otras áreas de aprendizaje automático, donde resulta relativamente sencilla la adquisición de datos para el entrenamiento de estos sistemas (por ejemplo, se pueden recuperar gran cantidad de textos con opiniones en Amazon para entrenar sistemas de análisis de sentimientos), son escasos los datos disponibles para entrenar sistemas que aprendan a recuperar e integrar tablas en función de las necesidades de los usuarios.

El presente capítulo ofrece una propuesta metodológica para la creación de corpus de datos tabulares que sirvan para el entrenamiento y evaluación de sistemas de aprendizaje automático como el desarrollado en esta tesis.

Esta propuesta comprende diferentes apartados. En primer lugar se describirá el conjunto de datos de partida con el que se trabajó para la anotación. En segundo lugar se presentará una guía de anotación para revisores humanos que establezca las pautas que deben seguir para etiquetar el conjunto de tablas. En tercer lugar se mostrará la interfaz desarrollada para facilitar a los revisores la aplicación de las pautas definidas en la guía. Finalmente, se describirá una prueba piloto de anotación realizada y se mostrarán las conclusiones obtenidas.

6.1. Datos de partida

En primer lugar, se recopiló un corpus amplio de tablas para llevar a cabo su anotación. Para ello nos basamos en el corpus definido en (Bhagavatula et al., 2015a). Este conjunto de datos está compuesto por 1,6 millones de tablas relacionales extraídas de la versión en inglés de Wikipedia de noviembre de 2013. Está libremente disponible para su descarga en la página

de los autores.[1] Los fichero originales están codificados en formato JSON, presentando los siguientes campos de información:

- `id`: identificador de la tabla
- `numCols`: número de columnas de la tabla
- `numDataRows`: número de filas de la tabla
- `numHeaderRows`: número de filas que componen la cabecera de la tabla. Lo habitual es que haya una sola fila, pero en ocasiones puede haber más de una cabecera
- `numericColumns`: número de columnas que almacenan datos de tipo numérico
- `pgTitle`: título de la página de Wikipedia de donde se extrajeron los datos
- `sectionTitle`: título de la sección de Wikipedia en la que se encuentra la tabla
- `tableCaption`: texto contenido en el pie de tabla
- `tableData`: contiene la información de cada una de las filas de la tabla
- `tableHeaders`: contiene los nombres de las distintas columnas de la tabla

En la Figura 6.1 se puede ver un ejemplo de tabla de Wikipedia codificada en el formato JSON original.

A partir de esta información se generó un conjunto de ficheros en formato CSV que contenían únicamente los datos que resultaban de utilidad para nuestros objetivos: nombres de las columnas y contenido de las filas. Todo el resto de metadatos asociados a las tablas se descartó. Aquí podemos ver un ejemplo de tabla una vez transformada a formato CSV:

```
"Year Inducted",Player,"Seasons at UA","NFL Team(s)","Years"
1963,"Don Hutson",1932{34,"Green Bay Packers",1935{45
1977,"Bart Starr",1952{55,"Green Bay Packers",1956{71
1985,"Joe Namath",1962{64,"New York Jets",1965{76
1985,"Joe Namath",1962{64,"Los Angeles Rams",1977
1991,"John Hannah",1970{72,"New England Patriots",1973{85
1998,"Dwight Stephenson",1977{79,"Miami Dolphins",1980{87
1999,"Ozzie Newsome",1974{77,"Cleveland Browns",1978{90
2009,"Derrick Thomas",1985{88,"Kansas City Chiefs",1989{99
```

[1] http://websail-fe.cs.northwestern.edu/TabEL/.

```
"_id": "10000230-1",
"numCols": 1,
"numDataRows": 1,
"numHeaderRows": 1,
"numericColumns": [],
"order": 0.7341481482144445,
"pgId": 10000230,
"pgTitle": "Barton Academy (Vermont)",
"sectionTitle": "Principals",
"tableCaption": "Principals",
"tableData": [
  [
    {
      "cellID": -1,
      "textTokens": [],
      "text": "Ernest C. Benjamin - 1880 C.H. Willey - March 1893 Harry J. Stannard - 1893
        -1914 F. Jay Bates 1915-? Mr. Meacham - 1924-26 Cedric Pierce - 1931-1948 David
        Shipp 1948-1952 George MacKenzie 1952-1958+ Cedric Pierce (a second time) 1961-1963
        Rev. William Chadwick 1963-1966 Raymond Mason 1966-1967",
      "tdHtmlString": "<td colspan=\"1\" rowspan=\"1\"> Ernest C. Benjamin - 1880 C.H.
        Willey - March 1893 Harry J. Stannard - 1893-1914 F. Jay Bates 1915-? Mr. Meacham -
        1924-26 Cedric Pierce - 1931-1948 David Shipp 1948-1952 George MacKenzie 1952-1958+
        Cedric Pierce (a second time) 1961-1963 Rev. William Chadwick 1963-1966 Raymond
        Mason 1966-1967 </td>",
      "surfaceLinks": [],
      "subtableID": -1,
      "isNumeric": false
    }
  ]
],
"tableHeaders": [
  [
    {
      "cellID": -1,
      "textTokens": [],
      "text": "Other principals of Barton Academy",
      "tdHtmlString": "<th colspan=\"1\" rowspan=\"1\"> Other principals of Barton Academy
        </th>",
      "surfaceLinks": [],
      "subtableID": -1,
      "isNumeric": false
    }
  ]
],
"tableId": 1
```


Figura 6.1: Ejemplo de tabla de Wikipedia codificada en el formato JSON del conjunto de datos original.

La primera línea contiene los nombres de las cabeceras de las columnas. Las líneas siguientes representan el contenido de la tabla. Cada valor de una celda está separada de los otros mediante el uso de comas.

Durante la obtención de las tablas en formato CSV se aplicaron una serie de filtros para obtener un conjunto de datos que fuera interesante desde el punto de vista de la anotación, eliminando tablas tanto muy pequeñas, que no proporcionen suficiente información para el proceso posterior de aprendizaje, como muy grandes, que supongan una dificultad elevada para los anotadores humanos.

Concretamente, se aplicaron los siguientes filtros a la hora de obtener el conjunto de tablas final:

- Eliminar tablas con menos de 2 columnas y con más de 5
- Eliminar tablas con menos de 5 filas y más de 10
- Eliminar tablas que no tengan al menos una columna de tipo textual, ya que es en estas columnas donde puede ser más interesante el uso de *word embeddings*
- Eliminar tablas que tengan más de una fila de cabecera, para evitar tablas con estructura compleja que puedan haber sido extraídas de manera errónea
- Eliminar tablas que tengan los nombres de columnas vacíos
- Eliminar tablas que contengan celdas de texto con más de 50 palabras, ya que tamaños tan grandes pueden deberse a errores en la extracción automática de los datos de Wikipedia

Tras este filtro, el resultado final fue un subconjunto de 87.354 tablas. De este conjunto se seleccionaron de manera aleatoria 1.000 tablas para formar el conjunto de consulta, dejando las restantes como tablas objetivo.

Para cada una de estas 1.000 tablas de consulta se hizo una llamada al sistema de búsqueda de tablas para unión basado en BERT que se describió en el Capítulo 4, de manera que se devolvieran las 10 tablas más similares de entre el conjunto de tablas objetivo. Para la configuración del modelo BERT de búsqueda de tablas se usó un valor $\alpha = 0{,}0$, de manera que solo se tuviera en cuenta el contenido de las celdas, y no el nombre de las columnas, a la hora de recuperar tablas similares. Esta decisión se tomó después de un primer ensayo en el que se dio igual ponderación a contenido y columnas, ya que se vio que muchas veces las tablas que devolvían era porque tenían nombres de columnas idénticos. Esto resulta de poca ayuda para mejorar el aprendizaje de los modelos de lenguaje, ya que enseñarles que dos columnas que se llaman "localidad" son similares no les aporta demasiado. Por esta razón, interesaba más dar peso a que los valores

de las celdas fueran similares sin mirar los nombres de columnas, de manera que se devolvieran tablas similares pero que pudieran tener nombres de columnas diferentes. Por ejemplo, asociar una columna llamada "localidad" con otra llamada "ciudad" sí que aporta información de interés para que el modelo de lenguaje aprenda que estos dos términos léxicamente diferentes tienen en realidad una similitud semántica muy elevada.

6.2. Guía de anotación

Se desarrolló una guía de anotación para que todos los usuarios que intervinieran en este estudio tuvieran el mismo criterio a la hora de etiquetar los datos. La guía incluía también una introducción al uso de la interfaz para que los usuarios se familiarizaran con la herramienta de anotación. Independientemente de la técnica utilizada para anotar, era necesario aclarar algunos conceptos para que todos tuvieran claro el objetivo de la anotación.

Lo primero que se aclaró era la finalidad de la tarea: obtener un conjunto de datos consistente de pares de tablas, donde se identifique si éstas pueden ser combinadas entre sí indicando qué columnas son similares. Se aclaró el concepto de similitud de la siguiente manera: "Dos columnas son similares si se pueden juntar los contenidos de una y otra en una única columna que contenga todos los datos de ambas, de manera que el resultado sea una columna coherente en cuanto a su contenido. Por ejemplo, podemos tener una tabla con una columna que contiene nombres de ciudades y otra tabla que contenga una columna también con nombres de ciudades. El resultado de esa unión sería una columna coherente donde todos sus valores son nombres de ciudades."

Como puede verse, la tarea de anotación se limitaba a que los revisores indicaran si dos columnas se podían considerar como similares. La propuesta inicial de guía de anotación contemplaba la posibilidad de que los revisores indicaran si las tablas se podían combinar mediante operaciones de unión y combinación. Tras una primera evaluación con tres anotadores, se llegó a la conclusión de que la tarea resultaba demasiado compleja para gente que no estuviera familiarizada con estas operaciones y con el concepto de integración de datos. Por esta razón, la tarea se acabó simplificando, ya que a posteriori podemos determinar si dos tablas pueden unirse o combinarse viendo la cantidad de columnas que los usuarios han marcado como similares. En el caso de la unión, esta se puede llevar a cabo cuando todas (o una gran mayoría) de las columnas entre dos tablas son similares. En el caso de la combinación, esta se puede realizar si existe una columna similar que además comparte contenido con el de la otra tabla.

Junto a las explicaciones, se proporcionaron una serie de ejemplos de tablas de entrada y su posible anotación usando la interfaz. Una parte importante de esta guía fue el aclarar algunas situaciones que se visualizaban

como conflictivas a la hora de llevar a cabo la anotación e interpretar el concepto de *similitud* descrito arriba. Se lista a continuación estas aclaraciones:

- Tratamiento de fechas: si hubiera formatos de fecha que no coinciden en columnas de diferentes tablas, se asumirá que las columnas no son equivalentes y por tanto no serían consideradas columnas similares.
- Tratamiento de unidades de medida: si hubieran diferentes órdenes de magnitud en las unidades de medida (o diferentes unidades de medida para la misma magnitud física) que no coinciden en columnas de diferentes tablas, se asumirá que las columnas no son equivalentes y por tanto no serían consideradas columnas similares.
- Diferentes granularidades en cuanto a fechas y localizaciones: si hubiera diferentes niveles de granularidad en los valores de diferentes columnas, pero el concepto al que hace referencia la columna estuviera relacionado, no se considerarán las columnas como equivalentes. Por ejemplo, si la columna "nombre" de una tabla hiciera referencia a nombres de países no se consideraría similar a una columna "ciudad", ya que aunque ambas se refieren a localizaciones no pertenecen al mismo nivel de granularidad. Lo mismo sucedería para otro tipo de granularidades como las que ocurren en fechas (día, mes, trimestre, año, etc.).

6.3. Interfaz de anotación

Para facilitar la anotación del conjunto de tablas se ha desarrollado una interfaz propia[2] con el objetivo de que sea usable y fácilmente accesible para cualquier tipo de usuario, sin necesidad de conocimientos técnicos avanzados, permitiendo además la gestión de múltiples usuarios y conexiones simultáneas.

Esta interfaz se ha desarrollado ante la ausencia de una solución en el mercado que se ajustara a las necesidades de anotación de la guía propuesta. Sin embargo, sí que existen herramientas en el ámbito del aprendizaje automático para la anotación de datos de otras tipologías (como imágenes o textos). Una de ellas es Labelbox,[3] la cual permite etiquetar imágenes para el desarrollo de sistemas de aprendizaje supervisado de manera colaborativa. Además del etiquetado, permite aplicaciones extra como por ejemplo la gestión de *pipelines* para refinado de etiquetas o analíticas para entender qué etiquetas están funcionando mejor o peor. Otra aplicación en esta

[2] Agradecer a Gabriel De Lamo el desarrollo técnico de esta interfaz.
[3] https://labelbox.com/.

línea es SuperAnnotate.[4] Destaca su integración con Azure y Google Cloud Platfor, así como la posibilidad de ser usada como aplicación de escritorio. La aplicación Datasaur.ai[5] permite la anotación, entre otras modalidades, de texto y audio. Unfun.me[6] presenta una aproximación diferente a la anotación de textos, convirtiendo el proceso en un juego. Concretamente, existen dos modos de juego: "unfun the headline", donde los usuarios deben convertir un título de noticia satírica en uno que sea más verosímil; en el segundo modo, "real or not", el usuario recibe un titular de una noticia y debe indiar cómo de probable es que dicho título sea real o satírico, recibiendo puntos en función de los aciertos en la predicción. Otra herramienta de anotación de datos es Snorkel.ai.[7] Ofrece un enfoque programático para el anotado de datos, de manera que se pueden definir funciones para automatizar algunos aspectos del proceso de anotación. Finalmente está LabelStudio,[8] una herramienta de código abierto que permite el etiquetado de datos de audio, texto, imágenes, vídeo y series temporales, permitiendo su exportación a diferentes formatos. Se puede usar para trabajar sobre datos en crudo o para mejorar datos de entrenamiento y conseguir modelos de aprendizaje más precisos.

La interfaz desarrollada permite al usuario, dadas dos tablas, establecer relaciones entre sus diferentes columnas y enviar los resultados para ser almacenados en una base de datos online para que posteriormente puedan ser utilizados como corpus de entrenamiento para sistemas de integración de datos basados en aprendizaje automático.

La Figura 6.2 muestra la pantalla de acceso para iniciar sesión en la aplicación. De esta forma se asocia el trabajo realizado a un usuario específico facilitando su posterior análisis (por ejemplo, para el cálculo del acuerdo entre anotadores). Los usuarios se pueden dar de alta mediante correo y contraseña o utilizando una cuenta existente de Google.

Una vez iniciada la sesión, se muestra un mensaje de bienvenida como el que se puede ver en la Figura 6.3. Este mensaje es meramente informativo y sirve como paso previo a la interfaz de anotación, la cual se puede ver en la Figura 6.4.

En la interfaz de anotación, la tabla de consulta aparece en la parte superior, mientras que las tablas similares van apareciendo en la parte inferior. Cuando el usuario selecciona una columna de una tabla y una columna de la otra, estas quedan emparejadas con el mismo color. A cada pareja identificada se le asigna un color diferente. Al mismo tiempo se añade una ficha en la parte derecha de la pantalla donde se ve la relación identificada y donde el usuario puede indicar cuáles son los indicios que le han llevado a indicar que ambas columnas son similares. `Por título` indica

[4] https://www.superannotate.com/.
[5] https://datasaur.ai/.
[6] http://unfun.me/.
[7] https://snorkel.ai/.
[8] https://github.com/heartexlabs/label-studio/.

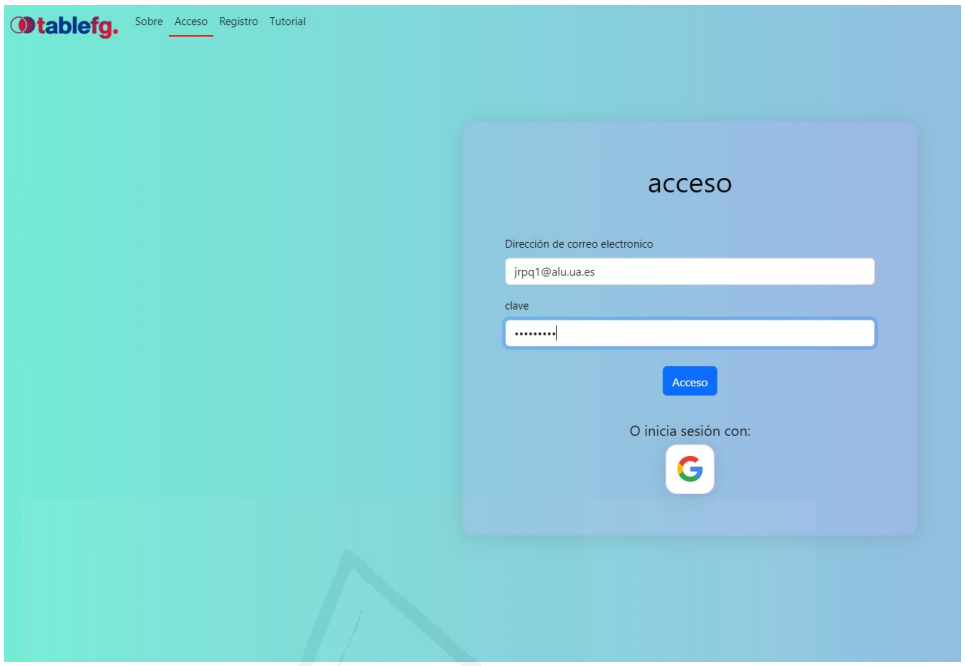


Figura 6.2: Interfaz de usuario para el inicio de sesión.

que el usuario se ha guiado por el nombre de ambas columnas para identificar que éstas son similares. `Por contenido` indica que es el contenido de las celdas lo que le ha llevado a marcar que ambas columnas son similares. Ambas opciones pueden activarse para una misma pareja de columnas. El objetivo de esta información es tener constancia explícita de si es el contenido de las celdas o el nombre de las columnas lo que lleva a los usuarios a identificar la similitud entre ellas. Esta información tiene un valor estadístico, pero podría servir también como característica de entrada para un sistema de aprendizaje automático. También se incluyó la posibilidad de realizar comentarios para cada una de las parejas de columnas identificadas, por si el usuario quisiera dejar algún tipo de aclaración para justificar su decisión.

Esta interfaz cuenta con tres botones en la parte de abajo para aquellas situaciones en las que el usuario no ha sido capaz de realizar la anotación y no empareja ninguna columna. Esto permite distinguir la anotación de aquellas situaciones en las que el usuario no ha identificado ninguna pareja porque realmente no ha visto similitud entre ellas. El botón `No sé el idioma` se puede usar en situaciones en las que los datos están en un idioma desconocido para el usuario. El botón `Desconozco el tema` permite al usuario indicar que no ha anotado las tablas porque tratan sobre una temática que desconoce y no se ve con conocimientos suficientes para poder identificar si ambas

Figura 6.3: Pantalla de bienvenida tras el inicio de sesión.

tablas tienen columnas que sean similares (por ejemplo, tablas de ámbito técnico o especializado). Finalmente, el botón `Otra razón` permite indicar que el motivo para no anotar ese par de tablas ha sido otro.

Finalmente, el botón `Terminado` permite indicar que se ha terminado de anotar esa pareja de tablas y que se desea pasar a la siguiente, guardando previamente la información de etiquetado en la base de datos del sistema.

El administrador de la herramienta puede determinar cuántas tablas de consultas y cuántas tablas similares para cada una de éstas se va a servir a cada usuario. Una barra de progreso en la parte inferior de la interfaz de anotación (ver Figura 6.4) permite ver cuántas parejas de tablas quedan pendientes por anotar.

6.4. Prueba piloto

Con el objetivo de realizar una primera aproximación al proceso de anotación y el refinamiento de la guía, se llevó a cabo una prueba piloto para la que se reclutó a 30 voluntarios, todos ellos estudiantes de Ingeniería Informática de la Facultad de Ingeniería y Ciencias Aplicadas de la Universidad Central de Ecuador.

El objetivo de esta prueba era realizar la anotación de 50 tablas del total de 1.000 del conjunto de consulta. Para cada una de esas 50 tablas se localizaron las 10 tablas más similares siguiendo el procedimiento descrito en la Sección 6.1. De esta manera, por cada tabla de consulta se generaban 10 parejas de tablas, correspondiente a la tabla de consulta en cuestión y a las 10 más similares devueltas por BERT. En total se formaron 500 parejas de tablas.

En el Apéndice A puede verse un ejemplo de tabla de consulta y las 10 tablas similares devueltas por el sistema de búsqueda de tablas basado en

Figura 6.4: Interfaz de anotación de tablas. En la parte superior se puede ver la tabla de consulta y en la parte inferior la tabla similar recuperada. A la derecha están las columnas en las que se ha identificado una relación.

BERT. En el Apéndice B se muestran las 50 tablas de consulta utilizadas en esta prueba piloto.

Para poder calcular el acuerdo entre anotadores se decidió utilizar a 3 voluntarios para cada una de las tablas, de manera que cada uno tenía que anotar 5 tablas de consulta y sus correspondientes 10 tablas más similares, para un total de 50 parejas de tablas. Este proceso llevó de promedio unos 20 minutos a cada uno de los anotadores.

El acuerdo interanotadores alcanzado fue de 0,71, lo que se considera un acuerdo sustancial entre ellos (Landis and Koch, 1977). El total de parejas de tablas que se quedó sin anotar (por al menos uno de los anotadores) fue de 87, un 17,4 % del total. De éstas, 16 fue por desconocimiento del idioma y 71 por desconocimiento de la temática. No hubo otros motivos. Este resultado refleja que trabajar con tablas en un dominio abierto, como es Wikipedia, puede llevar a que en ocasiones los anotadores desconozcan la temática de las tablas. Este problema se reduciría notablemente si trabajáramos en un dominio restringido con anotadores expertos en ese dominio.

La cantidad de tablas para las que se encontró al menos una columna similar fue del 100 % (descartando aquellas parejas de tablas que quedaron sin anotar por los motivos descritos en el párrafo anterior). Esto demuestra la eficacia del sistema de búsqueda de tablas, que fue capaz de devolver tablas con al menos una columna similar a juicio de los anotadores en todas las ocasiones.

En el transcurso de la experiencia se plantearon algunas dudas por parte de los usuarios:

- ¿Qué sucede si tenemos dos columnas que se llaman igual pero que su contenido no coincide? Por ejemplo, dos columnas que se llaman "Name" y que una contiene nombres de personas y otra nombres de películas. En ese caso no podemos indicar que las columnas son similares, ya que aunque su nombre coincida está claro que no son susceptibles de integración en ningún caso ya que contienen información de dominios diferentes. Otro ejemplo similar se encontró entre una tabla que tenía una columna "Winners" y otra tabla que tenía una columna "Champions". Por el nombre de la columna serían claramente similares, pero una contenía nombres de atletas y otra de equipos de fútbol, por lo que en este caso concreto no podemos indicar que ambas columnas son similares. Sin embargo, esa misma situación se daba entre otro par de tablas en las que las columnas se llamaban "Winners" y "Champions" pero las celdas contenían en ambos casos nombres de atletas. En esa situación las columnas sí se consideran similares.

- ¿Qué sucede si dos columnas tienen fechas, pero ninguna de las fechas de una columna se repite en la otra? En ese caso consideramos las columnas como similares por su contenido (puede que también por su título si se da el caso). Aunque el contenido no tenga extrictamente ningún valor repetido, sí que pertenece a la misma tipología (en este caso son todo fechas) y nos da una idea de que las columnas son similares.

6.5. Conclusiones

En la actualidad, la disponibilidad de conjuntos de dato para evaluar los procesos de integración de tablas son muy limitados. De hecho, no nos consta la existencia de ningún conjunto explícitamente pensado para la tarea de integración mediante combinación (*join*).

El conjunto de datos usado en los experimentos del Capítulo 4 estaba construido partiendo únicamente de 32 tablas originales, que fueron alineadas manualmente para identificar cuáles de ellas se podían integrar mediante la operación de unión. Estas tablas originales fueron luego proyectadas y seleccionadas hasta conseguir un conjunto de 5.000 tablas. Sin embargo, este conjunto no es lo suficientemente rico, a nuestro entender, para evaluar de manera satisfactoria los sistemas de búsqueda e integración de datos tabulares.

Por esta razón, planteamos en este capítulo el desarrollo de un conjunto de datos tabulares que pudiera servir para evaluar de manera más exigente estos sistemas, además de favorecer también el entrenamiento de aproximaciones basadas en aprendizaje automático a esta tarea. Se ha propuesto para ello una metodología de anotación manual y una interfaz para facilitar la tarea.

Se ha llevado a cabo una prueba piloto que ha implicado la anotación de 500 pares de tablas por un conjunto de revisores. El acuerdo entre anotadores es lo suficientemente alto como para considerar que el corpus tiene garantías de resultar de utilidad para la tarea propuesta. Más aún, si se consiguiera un conjunto de datos de mayor tamaño, podría llegar a servir de entrada para refinar un modelo de lenguaje contextual, como los utilizados en esta tesis, y obtener así un modelo mejor adaptado a las tareas de búsqueda e integración de datos tabulares.

7

Conclusiones y trabajo futuro

Este capítulo resume las conclusiones generales (Sección 7.1), las principales contribuciones de la tesis (Sección 7.2) y propone vías pendientes de investigación de cara al futuro (Sección 7.3).

7.1. Conclusiones generales

Esta primera sección resume las conclusiones que se han ido destacando ya a lo largo de los diferentes capítulos de esta tesis. El estudio del estado de la cuestión nos llevó a identificar aquellas áreas de la búsqueda e integración de datos tabulares en las que no existían trabajos relevantes sobre el uso de modelos de lenguaje contextuales. El posterior análisis de este tipo de modelos de lenguaje nos permitió encontrar los modelos que podían servir para llevar a cabo nuestro sistema.

El estudio llevado a cabo en el Capítulo 4 reveló que, en cuanto al rendimiento de los modelos contextuales, BERT y su versión refinada WikiTables mostraban más estabilidad y mejor rendimiento que RoBERTa a lo largo de todos los experimentos. Tanto en la búsqueda de tablas para unión como en la búsqueda de tablas para combinación, los modelos contextuales superaron significativamente a los no contextuales. BERT fue el mejor modelo en la búsqueda para unión y WikiTables en la búsqueda para combinación, aunque la diferencia no fue estadísticamente significativa en este último caso.

WikiTables obtuvo los mejores resultados en el experimento con la operación de unión, logrando una mejora significativa con respecto a otros modelos contextuales. En esta tarea se obtuvieron resultados mixtos, con fastText y Word2vec superando a BERT y RoBERTa en términos de puntuación F1-score. Los resultados de la operación de combinación no revelaron diferencias significativas entre los modelos no contextuales y BERT.

Por último, se vio también el impacto que el uso de encabezados de columna y valores de las celdas tenían en el rendimiento del sistema de búsqueda e integración. Los resultados revelaron que todos los modelos se beneficiaron de la combinación de ambos tipos de información, aunque los

modelos obtuvieron mejores resultados cuando se dio más peso a los valores de las celdas.

El trabajo llevado a cabo en el Capítulo 5 mostró que los modelos contextuales proporcionan *word embeddings* que son menos sensibles a la reducción de contenido. En el caso del conjunto de datos *Chicago*, mantener solo el 10 % de las filas proporcionó una similitud promedio entre la tabla original y la reducida superior al 90 %. En el caso de *Wikitables*, debía mantenerse el 60 % de las filas para lograr el mismo valor de similitud.

El impacto de la reducción de tablas fue menos significativo en tablas con muchas filas. Otros experimentos revelaron que el tipo de dato de las columnas, numérico o textual, afectaba significativamente al rendimiento de Word2vec y fastText. SentenceBERT no ofreció una diferencia significativa entre ambos tipos de datos.

El análisis de la cobertura de cada modelo mostró una clara correlación con el rendimiento. Word2vec ofreció la peor cobertura y los peores valores de similitud. En el caso de fastText, hubo una mayor cobertura de valores numéricos con respecto al texto, gracias a su propio algoritmo para trabajar a nivel de subpalabra, lo que se correlaciona con una similitud significativamente mayor utilizando solo valores numéricos.

Estos resultados proporcionan un camino prometedor para reducir los costos computacionales en la búsqueda de tablas usando incrustaciones de palabras.

Finalmente, en el Capítulo 6 se vio que la disponibilidad de conjuntos de dato para evaluar los procesos de integración de tablas son muy limitados. De hecho, no había constancia de la existencia de ningún conjunto explícitamente pensado para la tarea de integración mediante la operación de combinación. Por esta razón, planteamos en ese capítulo el desarrollo de un conjunto de datos tabulares que pudiera servir para evaluar de manera más exigente estos sistemas. Se propuso para ello una metodología de anotación manual y una interfaz para facilitar la tarea.

Se llevó a cabo una prueba piloto que implicó la anotación de 500 pares de tablas por un conjunto de 30 revisores. El acuerdo entre anotadores fue lo suficientemente alto como para considerar que el corpus tenía garantías de resultar de utilidad para la tarea propuesta.

7.2. Contribuciones

La principal contribución de este trabajo de investigación ha sido presentar un enfoque novedoso para la aplicación de *word embeddings* contextuales a la búsqueda e integración de datos tabulares. Estos modelos de lenguaje han logrado resultados punteros en muchas tareas de PLN donde el enfoque son los datos no estructurados (es decir, el texto en crudo). Sin embargo, en los últimos tiempos estos modelos también se han utilizado

para comprender mejor los datos estructurados y, más específicamente, la información tabular. Esta contribución principal ha requerido del desarrollo de una serie de contribuciones secundarias que se han ido describiendo en cada uno de los capítulos de esta tesis:

- En el Capítulo 2 se presentó un estudio sobre el estado de la cuestión en el ámbito de la búsqueda e integración de datos tabulares. Se presentaron las principales tareas que se desarrollan en cada una de estas áreas, así como los trabajos de investigación recientes. Se hizo especial hincapié en aquellas propuestas que hacían uso de modelos de lenguaje para el tratamiento de datos estructurados por su afinidad con el trabajo presentado en esta tesis. Este estudio permitió identificar los vacíos existentes a nivel de investigación por lo que respecta al uso de modelos de lenguaje contextuales para la búsqueda e integración de tablas.

- En el Capítulo 3 se hizo una revisión del estado de la cuestión de los modelos de lenguaje contextuales. Para ello se empezó definiendo la relación entre inteligencia artificial, aprendizaje automático y aprendizaje profundo. Se describieron algunas de las arquitecturas más populares en el ámbito de las redes neuronales, como son RNN, LSTM y Transformers. En esta última están basados la práctica totalidad de modelos de lenguaje contextuales existentes. Aquí se identificaron los modelos que podían ser de utilidad para la obtención de *word embeddings* que sirvieran para nuestro propósito de investigación. También se identificó en este capítulo la relación entre la inteligencia artificial, el PLN y la lingüística computacional.

- En el Capítulo 4 se presentó una solución basada en incrustaciones de palabras contextuales a la integración de datos mediante las operaciones de unión y combinación. Este proceso se dividió en dos fases: en primer lugar, se recuperan las tablas más relevantes para la operación correspondiente; en segundo lugar, se identifican las columnas candidatas a unión o combinación. Se han propuesto y analizado cuatro tareas diferentes: búsqueda para unión, búsqueda para combinación, operación de unión y operación de combinación. Cada tarea ha sido evaluada utilizando tres modelos de lenguaje contextuales (BERT, RoBERTa y WikiTables), dos modelos de lenguaje no contextuales (Word2vec y fastText) y un sistema base de recuperación de información clásico (BM25), este último únicamente para las dos tareas de búsqueda de tablas. En estas cuatro tareas se ha estudiado el impacto en el rendimiento del sistema que tiene el uso de encabezados de columna y valores de celda. Los resultados revelaron que todos los modelos se beneficiaron de la combinación de ambos

tipos de información, aunque los modelos funcionaron mejor cuando se dio más importancia a los valores de las celdas.

- En el Capítulo 5 se realizó un estudio de cómo la eliminación de contenido de las tablas puede afectar a la representación en forma de *word embedding* de estas. La hipótesis de partida de esta contribución fue que gran parte de las filas de una tabla pueden eliminarse sin que ello afecte significativamente a su representación vectorial en forma de *word embedding*, manteniendo así el rendimiento del sistema con un coste computacional y unas necesidades de almacenamiento mucho menores al inicial. Para comprobar esta hipótesis se realizó un estudio utilizando dos conjuntos de datos diferentes (uno basado en Wikipedia y otro perteneciente a un portal de datos abiertos de la administración) y tres modelos de lenguaje, dos de ellos no contextuales (Word2vec y fastText) y uno contextual (SentenceBERT). Los resultados obtenidos revelaron que, en tablas de gran tamaño, mantener solo un 10 % del contenido produce una representación como *word embedding* que es un 90 % similar a la original. Esto tiene importantes repercusiones a la hora de entrenar y evaluar este tipo de sistemas, ya que podemos prescindir de gran parte de los datos sin que esto afecte al rendimiento final.

- Finalmente, en el Capítulo 6 se propuso una metodología para el desarrollo de conjuntos de datos tabulares, etiquetados para el entrenamiento y/o evaluación de sistemas automáticos de búsqueda e integración de tablas. A diferencia de otras áreas de aprendizaje automático donde resulta sencillo adquirir datos de entrenamiento (como por ejemplo, la obtención de opiniones positivas y negativas sobre productos en Amazon), son escasos los datos disponibles para entrenar sistemas que aprendan a buscar e integrar tablas. Más aún, los conjuntos de datos existentes para la evaluación de los procesos de integración de tablas resultan poco desafiantes (caso de *union*) o incluso inexistentes (caso de *join*) para estos sistemas automáticos. Como parte de la metodología, se desarrolló una guía de anotación y una interfaz amigable para facilitar el trabajo de los revisores humanos. Esta propuesta se completó con la puesta en práctica de la metodología con un grupo de 30 anotadores que llevaron a cabo el desarrollo de un primer conjunto de datos compuesto de 500 pares de tablas.

7.3. Trabajo futuro

Tras la investigación llevada a cabo en esta tesis, se han observado un conjunto de líneas de trabajo futuro. En el caso del sistema de búsqueda e integración de datos tabulares, se vio que el criterio para determinar

si dos columnas eran susceptibles de integrarse mediante la operación de combinación no era el más adecuado. Este criterio consistía en seleccionar solo la columna con la similitud más alta en cada tabla para identificar las columnas candidatas para la combinación. El rendimiento del sistema se veía muy afectado cuando los modelos de lenguaje asignaban valores de similitud altos con una dispersión baja de manera generalizada, cosa que ocurría más a menudo en los modelos contextuales que en los no contextuales. Se debe, pues, trabajar en otros criterios para mejorar el rendimiento de los modelos contextuales en la tarea de integración usando la operación de combinación.

El trabajo realizado en el Capítulo 4 se centró principalmente en definir la aproximación a la búsqueda e integración y analizar el rendimiento de los modelos contextuales a lo largo de distintas dimensiones (información utilizada, precisión de los modelos y papel que juega el contexto). Como trabajo futuro se pretende también usar vectores generados por los modelos para entrenar algoritmos de aprendizaje automático e incluirlos en un sistema final, permitiendo inferir de manera automática los mejores parámetros (valores α y umbrales) para la tarea de integración. Otro trabajo en esta línea es el de ampliar el uso de operadores de integración más allá de la unión y la combinación. También quedó pendiente la mejora del modelo refinado WikiTables, el cual se podría entrenar con ejemplos tabulares adicionales, como los que se plantea recopilar siguiendo la metodología propuesta en el Capítulo 6.

En el caso del estudio sobre el impacto de la eliminación de contenido del Capítulo 5, como trabajo futuro se propone utilizar un criterio más informado para la eliminación de filas, ya que en la propuesta realizada se hacía de manera aleatoria. Además, otro trabajo será el llevar a cabo una evaluación extrínseca para comprobar cómo afecta la reducción de tablas al rendimiento final de un sistema de búsqueda de tablas mediante incrustaciones de palabras como el propuesto en el Capítulo 4.

En cuanto a la metodología de creación de conjuntos de datos tabulares para la integración, queda pendiente el desarrollo de un conjunto de tablas de gran tamaño que amplíe el estudio piloto llevado a cabo. Este conjunto de datos se dejará a libre disposición de la comunidad científica para que puedan evaluarse (o incluso entrenarse) de manera adecuada los sistemas basados en aprendizaje automático que están surgiendo en la actualidad para la búsqueda e integración de tablas.

8 Publicaciones

A continuación se resumen las publicaciones derivadas del trabajo realizado en esta tesis. La primera de ellas corresponde a los contenidos del Capítulo 4, la segunda al Capítulo 5 y la tercera al Capítulo 6.

- *José Pilaluisa, David Tomás, Borja Navarro-Colorado, Jose-Norberto Mazón. Contextual word embeddings for tabular data search and integration. Neural Computing and Applications, pp. 1-15, 2022.*

  Este artículo presenta un nuevo enfoque para buscar e integrar conjuntos de datos tabulares (colecciones de filas y columnas) mediante operaciones de unión (*union*) y combinación (*join*). En este trabajo, ambos procesos se llevan a cabo utilizando una medida de similitud basada en *word embeddings* contextuales, que permite encontrar tablas semánticamente similares y superar el problema de cobertura de los enfoques léxicos basados en la similitud de cadenas. Este trabajo es el primer intento de utilizar *word embeddings* contextuales en toda la cadena de búsqueda e integración de tablas, incluyendo por primera vez su uso en la operación de combinación. Se realizó un análisis exhaustivo de su rendimiento tanto en la recuperación como en la integración de conjuntos de datos tabulares, comparándolos con modelos no contextuales. Se utilizaron como información contextual los títulos de las columnas y los valores de las celdas, y se evaluó su impacto en cada tarea. Los resultados revelaron que los modelos contextuales superan significativamente a los modelos no contextuales y a un esquema de ponderación tradicional en la recuperación de tablas *ad hoc*. En la tarea de integración de datos, los modelos contextuales también mejoraron los resultados en la operación de unión en comparación con los enfoques no contextuales.

- *José Pilaluisa, David Tomás. The impact of content deletion on tabular data similarity using contextual word embeddings. In Proceedings of the 17th International Conference on Soft Computing Models in Industrial and Environmental Applications, SOCO 2022, pp. 250-259, September 5 - September 7, 2022, Salamanca, Spain.*

La recuperación de tablas es la tarea de responder a una consulta de búsqueda con una lista ordenada de tablas que se consideran relevantes para dicha consulta. Calcular la similitud de las tablas es una parte fundamental de este proceso. Los modelos lingüísticos actuales basados en Transformers se han utilizado con éxito para obtener representaciones de *word embeddings* de las tablas con el fin de calcular su similitud semántica. Desgraciadamente, obtener representaciones de *word embeddings* de tablas grandes con miles o millones de filas puede ser un proceso costoso desde un punto de vista computacional. El presente trabajo plantea la hipótesis de que gran parte del contenido de una tabla puede eliminarse (es decir, pueden suprimirse filas) sin que ello afecte significativamente a su representación como *word embedding*, manteniendo así el rendimiento del sistema con un coste computacional mucho menor. Para comprobar esta hipótesis se ha realizado un estudio utilizando dos conjuntos de datos diferentes y tres modelos lingüísticos de última generación. Los resultados obtenidos revelan que, en tablas de gran tamaño, mantener sólo un 10 % del contenido produce una representación como *word embedding* que es un 90 % similar a la original.

- *José Pilaluisa, David Tomás. Development of a dataset to train and evaluate machine learning based systems for tabular data integration. En desarrollo.*

  Existen en la actualidad numerosas aproximaciones basadas en aprendizaje automático para afrontar el problema de la búsqueda e integración de datos tabulares. Como cualquier aproximación de este tipo, el problema que tienen estos algoritmos es la necesidad de grandes volúmenes de datos para poder entrenarlos de manera óptima. A diferencia de otras áreas de aprendizaje automático, donde resulta sencillo adquirir datos de entrenamiento, son escasos los datos disponibles para entrenar sistemas que aprendan a recuperar e integrar tablas en función de las necesidades de los usuarios. Más aún, los conjuntos de datos existentes para la evaluación de los procesos de integración de tablas resultan poco desafiantes para estos sistemas automáticos. Este trabajo propone una metodología para el desarrollo de corpus de datos tabulares, etiquetados para el entrenamiento y/o evaluación de sistemas automáticos de búsqueda e integración de tablas. Se mostrará la guía de anotación desarrollada, la interfaz creada para facilitar el proceso a los revisores humanos, así como los resultados de un estudio piloto desarrollado para anotar un conjunto de datos de prueba.

A

Ejemplo de tabla de origen y tablas similares

#3 (table-0269-760): NHL FaceOff

Installments

Title	Year	Platform(s)	Cover Athlete
NHL FaceOff	1995	PlayStation	Sergei Fedorov
NHL FaceOff 97	1996	PlayStation	Paul Coffey
NHL FaceOff 98	1997	PlayStation	John LeClair
NHL FaceOff 99	1998	PlayStation	Chris Chelios
NHL FaceOff 2000	1999	PlayStation	John LeClair
NHL FaceOff 2001	2000	PlayStation	Curtis Joseph
NHL FaceOff 2002 (cancelled game)	2001	PlayStation 2	Luc Robitaille
NHL FaceOff 2003	2002	PlayStation 2	Rob Blake

#3.1 (table-1310-968): Millennium Kitchen

Games developed

Year	Game	Platform(s)
2000	Boku no Natsuyasumi	PlayStation
2002	Boku no Natsuyasumi 2	PlayStation 2
2007	Boku no Natsuyasumi 3	Playstation 3
2009	Boku no Natsuyasumi 4	PlayStation Portable
2013	Attack of the Friday Monsters	Nintendo 3DS

#3.2 (table-0634-199): Quantic Dream

Games developed

Year	Game	Platform(s)
1999	Omikron: The Nomad Soul	Microsoft Windows
2005	Fahrenheit	PlayStation 2
2010	Heavy Rain	PlayStation 3
2013	Beyond: Two Souls	PlayStation 3
TBA	Unknown	PlayStation 4

#3.3 (table-0405-318): Capcom Vancouver

as

Year	Game	Platform(s)
2007	The Bigs	PlayStation 2
2009	MLB Front Office Manager	Microsoft Windows
2009	The Bigs 2	Nintendo DS
2010	Dead Rising 2: Case Zero	Xbox 360
2010	Dead Rising 2	Microsoft Windows, PlayStation 3, Xbox 360
2010	Dead Rising 2: Case West	Xbox 360

#3.4 (table-0358-771): Elixir Studios

Games

Release Date	Titles	Genre	Platform(s)
2003	Republic: The Revolution	Strategy	Windows
2004	Evil Genius	Strategy	Windows
Canceled	Blue Vault	Real-time strategy	Windows
Canceled	Republic Dawn: The Chronicles of the Seven	Persistent Online Strategy	Windows
Canceled	Republic: The Revolution	Strategy	Xbox
Canceled	Evil Genius 2	Strategy	Windows

#3.5 (table-1125-588): Bloober Team

Games

Release Date	Titles	Genre	Platform(s)
2010	Music Master Chopin	Music	Microsoft Windows
2011	Double Bloob	Action	Nintendo DS
2011	Paper Wars: Cannon Fodder	Strategy	Wii
2013	A-Men (2013 video game)	Puzzle	PlayStation 3
2013	A-Men 2 (2013 video game)	Puzzle	PlayStation 3
2013	Basement Crawl	Action	PlayStation 4
To_be_announced	Last Flight (video game)	Action	Wii

#3.6 (table-1271-148): Cipher Prime

List of video games

Year	Title	Platform
2008	Auditorium	Flash, iOS, Playstation 3, PSP, Steam
2010	Fractal	Windows, OS X, Linux, Android, iOS
2011	Pulse	Android tablets, iPad
2012	Splice	Windows, OS X, Linux, Android, iOS
2013	Intake	Windows, OS X
TBA	Auditorium 2: Duet	Windows, OS X, Linux

#3.7 (table-1244-298): 11 bit studios

Games

Title	Year	Platform
Anomaly: Warzone Earth	2011	Microsoft_Windows
Funky Smugglers	2012	iOS
Sleepwalker's Journey	2012	iOS
Anomaly: Korea	2012	iOS
Anomaly 2	2013	Microsoft_Windows

#3.8 (table-0380-79): John Powell

1990s

Year	Title	Director	Studio(s)	Notes
1997	Face/Off	John Woo	Paramount Pictures	N/A
1998	With Friends Like These...	Philip Frank Messina	Miramax Films	N/A
1998	Antz	Eric Darnell	DreamWorks Pictures	Harry Gregson-Williams
1999	Endurance	Leslie Woodhead	Walt Disney Pictures	N/A
1999	Forces of Nature	Bronwen Hughes	DreamWorks Pictures	N/A
1999	Chill Factor	Hugh Johnson	Warner Bros. Pictures	Hans Zimmer

#3.9 (table-0694-728): Nick Pollock

Discography

Year	Title	Band	Track(s)
1992	RockHard Präsentiert Virgin Summer Slam	My Sister's Machine	"I Hate You"
1993	Crossing All Over!	My Sister's Machine	"I Hate You"
1993	Crunchy Goodness	My Sister's Machine	"Steamy Swamp Thang"
1994	New Metal Ballads	My Sister's Machine	"Empty Room"
1994	Hard Music Volume 1	My Sister's Machine	"Enemy"
2008	Unleashed 4	Soulbender	"Three Towers"
2009	Unleashed 5	Soulbender	"Clockwork and Compass"
2010	Silvertongue	Sundance Crow	"Silvertongue"

#3.10 (table-1207-416): Innerprise Software

Video games

Title	Year	Platform	Region
Nightdawn	1989	Amiga	North America
Persian Gulf Inferno	1989	Amiga	North America
Battle Squadron	1989	Amiga	North America
Plague	1990	Amiga	Europe
Globulus	1990	Amiga	Europe
Battle Squadron	1990	Sega Genesis	North America
Ms. Pac-Man	1991	Sega Genesis	North America
Sword of Sodan	1991	Sega Genesis	North America
Battle Squadron	1991	Amiga	Europe

B

Conjunto de tablas de origen

1: 2004 Canadian Grand Prix

Standings after the race

Pos	Constructor	Points
1	Ferrari	124
2	Renault	61
3	BAR	52
4	Williams	36
5	Sauber	15

2: Inner Child

Charts

Chart (1992)	Peak position
Swiss Album Charts	12
Norwegian Album Charts	16
Swedish Albums Chart	17
Austrian Album Charts	29
Billboard 200	83
Top R&B Albums	13

3: NHL FaceOff

Installments

Title	Year	Platform(s)	Cover Athlete
NHL FaceOff	1995	PlayStation	Sergei Fedorov
NHL FaceOff 97	1996	PlayStation	Paul Coffey
NHL FaceOff 98	1997	PlayStation	John LeClair
NHL FaceOff 99	1998	PlayStation	Chris Chelios
NHL FaceOff 2000	1999	PlayStation	John LeClair
NHL FaceOff 2001	2000	PlayStation	Curtis Joseph
NHL FaceOff 2002 (cancelled game)	2001	PlayStation 2	Luc Robitaille
NHL FaceOff 2003	2002	PlayStation 2	Rob Blake

4: United States presidential election in Oregon, 2004

By congressional district

District	Bush	Kerry	Representative
1st	44%	55%	David Wu
2nd	61%	38%	Greg Walden
3rd	33%	67%	Earl Blumenauer
4th	49%	49%	Peter DeFazio
5th	50%	49%	Darlene Hooley

5: Athletics at the 1928 Summer Olympics – Men's 100 metres

Heat 9

Place	Name	Time
1	Georg Lammers	10.8 seconds
2	André Théard	unknown
3	János Paizs	unknown
4	Leo Jørgensen	unknown
5	Jean Moulin	unknown
6	Renos Frangoudis	unknown

6: Ontario University Athletics women's ice hockey

All-star teams

Player	Position	School
Glenda Rosen	G	McMaster
Sue Scherer	D	Guelph
Sophie Radecki	D	Toronto
Carolyn Aylesworth	C	Queens
Marjot Verlaan	RW	McMaster
Heather Ginzel	LW	Toronto

7: Ha (kana)

Stroke order

Form	Romanization_of_Japanese	Hiragana	Katakana
Normal h- (は行 ha-gyō)	ha	は	ハ
Normal h- (は行 ha-gyō)	haa hā, hah	はあ, はぁ はー	ハア, ハァ ハー
dakuten	ba	ば	バ
dakuten	baa bā, bah	ばあ, ばぁ ばー	バア, バァ バー
handakuten	pa	ぱ	パ
handakuten	paa pā, pah	ぱあ, ぱぁ ぱー	パア, パァ パー

8: 2008 BWF Grand Prix Gold and Grand Prix

Grand Prix Gold

Category	Winners	Runners-up	Score
Men's singles	Boonsak Ponsana	Chetan Anand	21–16, 21–12
Women's singles	Zhou Mi	Lu Lan	21–14, 21–14
Men's doubles	Guo Zhendong	Chan Chong Ming	19–21, 21–14, 21–12
Women's doubles	Chien Yu-chin	Miyuki Maeda	21–17, 21–16
Mixed doubles	He Hanbin	Kristof Hopp	21–18, 21–9

9: 2009–10 Los Angeles Kings season

Miscellaneous

Player	New team	Contract terms
Matt Moulson	New York Islanders	1 year
Vladimir Dravecky	HC Košice	1 Year
Derek Armstrong	St. Louis Blues	1 year, 2-way NHL/AHL
Dan Taylor	Syracuse Crunch	1 year, two-way AHL/ECHL
Kyle Calder	Anaheim Ducks	1 year, two-way NHL/ECHL

10: 2004 Little League World Series qualification

State	City	LL Organization	Record
Kentucky	Owensboro	Southern	4-0
Ohio	Hamilton	West Side	2-2
Indiana	Highland	Highland	2-2
Illinois	Mundelein	Mundelein National	2-2
Wisconsin	Appleton	Einstein	1-3
Michigan	Grand Rapids	Western	1-3

11: 1983 World Championships in Athletics – Men's hammer throw

Abbreviations

Q	automatic qualification
q	qualification by rank
DNS	did not start
NM	no mark
WR	world record
AR	area record
NR	national record
PB	personal best
SB	season best

12: Under the Hood

Other on-screen talent

Role:	Name:
Commentators	Bryce Remsburg
Commentators	Gavin Loudspeaker
Commentators	Leonard F. Chikarason
Commentators	Wink Vavasseur
Ring announcers	Gavin Loudspeaker
Referees	Bryce Remsburg
Referees	Daniel Yost
Referees	Jonathan Barber

13: 2009 World Series of Poker Europe

Final Table

Place	Name	Prize
1st	Erik Cajelais (1/1)	£104677
2nd	Mats Gavatin	£64705
3rd	Robin Keston	£47858
4th	Men Nguyen	£35412
5th	Richard Gryko	£26619
6th	Chris Bjorin	£20106
7th	Hoyt Corkins	£15302
8th	Ian Frazer	£11732
9th	Howard Lederer	£9117

14: Nepenthes angasanensis

Differences between N. angasanensis, N. mikei and N. tobaica (Salmon & Maulder, 1999)

Character	N.angasanensis	N.mikei	N.tobaica
Habit	Produces offshoots from underground rhizomes	No rhizomes	No rhizomes
Spur	Forked	Fasciculate	Filiform
Inner margin of peristome	Teeth to 1.5–2mm long	Teeth to 0.2-0.4mm long	Teeth < 0.2mm
Stem cross section	Cylindrical	Cylindrical	Cylindrical to obtusely triangular
Bracteoles	Sometimes near base of lowest pedicel only	Half way up every pedicel	At base or slightly below pedicel attachment, few
Pitcher glands	300 / cm²	150-180 / cm²	200-250 / cm²
Pedicels	1-flowered	1-flowered	2-flowered
Inflorescence (female)	55–125mm long, 9-17 flowers	40–80mm long, 4-10 flowers	195–400mm long, 30-50 flowers

15: Miss May I

Music videos

Year	Song	Director
2009	"Architect" (demo version)	Thunder Down Country
2009	"Forgive and Forget"	Spencer Nicholson
2009	"Forgive and Forget (Saw VI Soundtrack)"	Spencer Nicholson
2010	"Relentless Chaos"	Thunder Down Country
2010	"Our Kings"	Thunder Down Country
2011	"Masses of a Dying Breed"	Thunder Down Country
2011	"Relentless Chaos (Live)"	Cole Dabney
2012	"Hey Mister"	Thunder Down Country
2012	"Day By Day"	Thunder Down Country

16: Simply Deep

Release history

Region	Date
Australia	October 22, 2002
Canada	October 22, 2002
Italy	October 22, 2002
Mexico	October 22, 2002
United States	October 22, 2002
Japan	January 22, 2003
Austria	February 3, 2003
France	February 3, 2003
Ireland	February 3, 2003
Germany	February 10, 2003

18: 1989–90 Greek Cup

Group 7

Team	Pts
Panionios	6
Panetolikos	6
Atromitos	4
Eordaikos	3
Pontioi Veria	1

19: Ruby on Rails

Version history

Version	Date
1.0	Cannot handle non-empty timestamp argument! 13, 2005
1.2	Cannot handle non-empty timestamp argument! 19, 2007
2.0	Cannot handle non-empty timestamp argument! 7, 2007
2.1	Cannot handle non-empty timestamp argument! 1, 2008
2.2	Cannot handle non-empty timestamp argument! 21, 2008
2.3	Cannot handle non-empty timestamp argument! 16, 2009
3.0	Cannot handle non-empty timestamp argument! 29, 2010
3.1	Cannot handle non-empty timestamp argument! 31, 2011
3.2	Cannot handle non-empty timestamp argument! 20, 2012
4.0	Cannot handle non-empty timestamp argument! 25, 2013

20: Comparison of layout engines (web typography)

Support in SVG documents

Format	Trident	Gecko	WebKit	Presto
(EOT)	No	No	No	No
(TTF)	No	Yes	Yes	2.2
(OTF)	No	18.0	No	2.2
(SVG)	No	No	525	2.2
(WOFF)	No	No	No	2.7.81

21: 2011 Southern Conference football season

Preseason Poll Results

Media	Coaches
Georgia Southern (15)	Appalachian State (5)
Appalachian State (13)	Georgia Southern (3)
Wofford (2)	Wofford
Chattanooga	Chattanooga (1)
Furman	Elon
Elon	Furman
Samford	Samford
Western Carolina	The Citadel
The Citadel	Western Carolina

22: Endlessly (album)

Year-end charts

Chart (2010)	Position
Danish Albums Chart	34
Swedish Albums Chart	61
Chart (2011)	Position
Belgian Albums Chart (Flanders)	84
Belgian Albums Chart (Wallonia)	89
Danish Albums Chart	100
Dutch Albums Chart	98
Swedish Albums Chart	77

23: 2008 The National

Pool C

Team	W	L
Stoughton	5	0
Ferbey	3	2
Ferland	3	2
Koe	2	3
Jordison	1	4
Ursel	1	4

24: Sioux Falls Storm

Passing

Name	TD Passes	Opponent	Date
Terrance Bryant	8	Lexington Horsemen	July 29, 2006
Chris Dixon	8	Lehigh Valley Steelhawks	June 23, 2012
Ryan Aulenbacher	5	Utah Warriors	July 12, 2003
Chris Dixon	5	Omaha Beef	June 25, 2011
Chris Dixon	5	Green Bay Blizzard	June 30, 2012
Terrance Bryant	4	Louisiana Swashbucklers	August 2, 2008
Chris Dixon	4	Tri-Cities Fever	July 14, 2012
Terrance Bryant	4	Nebraska Danger	June 29, 2013
7 TIED	3	–	–

25: David Paul (actor)

Filmography

Film	Role	Year
D.C. Cab	Buzzy (as David Barbarian)	1983
The Flamingo Kid	Turk (uncredited)	1984
The Barbarians	Gore	1987
The Road Raiders	Black	1989
Think Big	Victor	1989
Ghost Writer	Marco	1989
Double Trouble	David Jade	1992
Natural Born Killers	The Hun Brothers (scenes deleted)	1994
Twin Sitters	David Falcone	1994
Souled Out	Ace Stevens	2005

26: General Government

Distribution of food in General Government as of December, 1941

Nationality	Daily calorie intake
Germans	2310
Foreigners	1790
Ukrainians	930
Poles	654
Jews	184

27: 2004 CAF Champions League

Second Round

Team 1	Agg.	Team 2	1st leg	2nd leg
Supersport United	2-0	Al-Hilal Omdurman	2-0	0-0
Orlando Pirates	2-2	Bakili Bullets	2-1	0-1
APR FC	1-1	Africa Sports National	1-0	0-1
Espérance	3-1	Coton Sport FC	3-0	0-1
p	2-2	Asante Kotoko	2-0	0-2
Jeanne d'Arc	3-2	Canon Yaoundé	2-0	1-2
Hearts of Oak SC	1-1	Etoile du Sahel	1-0	0-1
Enyimba	3-2	Petro Atlético	1-1	2-1

28: Table tennis at the 2011 All-Africa Games

Medal summary

Event	Gold	Silver	Bronze
Men's Singles	Omar Assar	El-sayed Lashin	Ahmed Ali Saleh
Women's Singles	Offiong Edem	Olufunke Oshonaike	Han Xing
Men's Doubles	Egypt	Egypt	Republic of the Congo
Women's Doubles	Nigeria	Nigeria	Egypt
Mixed Doubles	Nigeria	Nigeria	Egypt
Men's Team	Egypt	Nigeria	Algeria
Women's Team	Egypt	Nigeria	Republic of the Congo

29: The Omar Khayyam Show

Episode	Title	Original broadcast date
1	Ned Kelly	27 December 1963
2	The Ashes	3 January 1964
3	The Prime Minister's Trousers	10 January 1964
4	The Flying Dustman	17 January 1964
5	The America Cup	24 January 1964
6	Waltzing Matilda	5 February 1964

30: 2002 European Aquatics Championships

Women's events

Event	Gold	Silver	Bronze
1 m springboard	Heike Fischer	Vera Ilyina	Natalya Umyskova
3 m springboard	Yuliya Pakhalina	Ditte Kotzian	Conny Schmalfuß
10 m platform	Anke Piper	Tania Cagnotto	Olga Leonova
3 m springboard synchro	Ditte Kotzian	Vera Ilyina	Tania Cagnotto
10 m platform synchro	Ditte Kotzian	Olga Leonova	Yevgeniya Olshevskaya

31: Puerto Rico Islanders statistics

All-Time Minutes Leaders

Rank	Name	Career	Min
1	Noah Delgado	2005 - current	9319
2	Marco Vélez	2005 - 2007; 2009 - current	7699
3	Alexis Rivera Curet	2004 - current	7333
4	Edwin Miranda	2006 - current	7179
5	Petter Villegas	2005 - current	7076

32: British Rail regional multiple unit numbering

London Midland Region

Range	Classes
003-005	Class 128
012-014 / 050-062	Class 122
021-027	Class 114
301-315	Class 117
316-318	Class 118
319-342	Class 116
401-421	Class 115
501-545	Class 116
600-629	Class 116

33: Joel Feeney

Music videos

Year	Video
1991	"Diamonds"
1992	"If Anything Could Be"
1995	"What Kind of Man"
1995	"Life Is Just a Dream"
1998	"A Little Bit of Your Love"
1998	"Leslie's Wedding Day"

34: Let's Go! (Wang Chung song)

Charts

Chart	Peak position
Billboard Hot 100	9
Hot Dance Club Play	4
Mainstream Rock Tracks	18
UK Singles Chart	81
Canada RPM Top 100 Singles	9

35: Scarborough Athletic F.C.

Average league attendance

Season	Average	Lowest	Highest
2007–08	485	376	605
2008–09	493	304	791
2009–10	435	266	701
2010–11	415	259	725
2011–12	419	254	748
2012–13	447	321	950
2013–14	398	287	518

36: 2005 Masters Tournament

Third round (Saturday)

Place	Player	Country	To par	Hole
1	Chris DiMarco	United States	–13	9
2	Tiger Woods	United States	–9	9
3	Thomas Bjørn	Denmark	–8	9
T4	Mark Hensby	Australia	–4	9
T4	Vijay Singh	Fiji	–4	10
T4	Rod Pampling	Australia	–4	12

37: The Franchise (TV series)

Season 2: 2012

No.	Title	Original air date
0	"Miami Marlins: Special Preview"	April21,2012
1	"Miami Marlins: Episode 1"	July11,2012
2	"Miami Marlins: Episode 2"	July18,2012
3	"Miami Marlins: Episode 3"	July25,2012
4	"Miami Marlins: Episode 4"	August1,2012
5	"Miami Marlins: Episode 5"	August8,2012
6	"Miami Marlins: Episode 6"	August15,2012
7	"Miami Marlins: Episode 7"	August22,2012

38: Der Golem (opera)

Roles

Role	Voice type	Premiere cast 14 November 1926 (Conductor: Clemens Krauss)
Der Golem	bass	Jean Stern
Rabbi Loew	baritone	Robert vom Scheidt
His apprentice	tenor	Hans Brandt
Lea, Rabbi Loew's foster daughter	soprano	Elisabeth Kandt
Emperor Rudolf II	bass	Walter Schneider
First Jew	tenor	Hans Brandt
Second Jew	baritone	Robert vom Scheidt

39: 2007 European Junior Curling Challenge

Standings

Country	G	W	L
Italy	4	4	0
Sweden	4	3	1
Czech Republic	4	3	1
Hungary	4	2	2
Germany	4	2	2
Finland	4	2	2
Poland	4	0	4
Spain	4	0	4

40: 1934 Masters Tournament

Third round

Place	Name	Country	Score	To par
1	Horton Smith	United States	70-72-70=212	–4
2	Billy Burke	United States	72-71-70=213	–3
T3	Craig Wood	United States	71-74-69=214	–2
T3	Ed Dudley	United States	74-69-71=214	–2
5	Paul Runyan	United States	74-71-70=215	–1
T6	Walter Hagen	United States	71-76-70=217	+1
T6	Willie Macfarlane	Scotland	74-73-70=217	+1
T8	Jimmy Hines	United States	70-74-74=218	+2
T8	Macdonald Smith	Scotland	74-70-74=218	+2
10	Al Watrous	United States	74-74-71=219	+3

41: 1999 World Championships in Athletics – Men's pole vault

Abbreviations

Q	automatic qualification
q	qualification by rank
DNS	did not start
NM	no mark
WR	world record
AR	area record
NR	national record
PB	personal best
SB	season best

42: Temptation (Heaven 17 song)

Brothers in Rhythm Remix

Chart (1992)	Peak position
UK Singles Chart	4
Irish Singles Chart	9
Chart (1993)	Peak position
Dutch Top 40	16
German Singles Chart	42
GfK	18

43: Come with Us

Release history

Region	Release date	Label	Format	Catalogue
Japan	21 Cannot handle non-empty timestamp argument! 2002	Virgin Japan	CD	VJCP-68367
UK	28 Cannot handle non-empty timestamp argument! 2002	Freestyle Dust	CD	XDUSTCD5
UK	28 Cannot handle non-empty timestamp argument! 2002	Freestyle Dust	CD	XDUSTCDX5
UK	28 Cannot handle non-empty timestamp argument! 2002	Freestyle Dust	LP	XDUSTLP5
UK	28 Cannot handle non-empty timestamp argument! 2002	Freestyle Dust	MC	XDUSTMC5
USA	29 Cannot handle non-empty timestamp argument! 2002	Astralwerks	CD	ASW11682-2
USA	29 Cannot handle non-empty timestamp argument! 2002	Astralwerks	CD	ASW11895-2
USA	29 Cannot handle non-empty timestamp argument! 2002	Astralwerks	LP 2×LP	ASW11682-1
Japan	29 Cannot handle non-empty timestamp argument! 2002	Virgin Japan	CD	VJCP-68408

44: UEFA Euro 2016 bids

Schedule

Date	Notes
15 December 2008	Applications formally invited
9 March 2009	Closing date for registering intention to bid
3 April 2009	Announcement of bidders
15 February 2010	Submission of detailed bids
7–15 April 2010	Bid visits
28 May 2010	Bid presentation and announcement of hosts

45: So You Think You Can Dance (U.S. season 4)

Week 4 (July 3, 2008)

Dancer	Style	Music	Result
Kourtni Lind	Contemporary	"No Man"—Nina Storey	Eliminated
Matt Dorame	Contemporary	"Sweet Contentment"—Bradley James and the Roadies	Eliminated
Courtney Galiano	Jazz	Maria Mena	Safe
Gev Manoukian	B-boying	Everybody Loves a Carnival	Safe
Comfort Fedoke	Hip-Hop	Twista	Safe
Thayne Jasperson	Contemporary	I Want To Break Free	Safe

46: 1981 San Francisco Giants season

Farm system

Level	Team	League	Manager
AAA	Phoenix Giants	Pacific Coast League	Rocky Bridges
AA	Shreveport Captains	Texas League	Jack Mull
A	Fresno Giants	California League	Wayne Cato
A	Clinton Giants	Midwest League	Wendell Kim
Rookie	Great Falls Giants	Pioneer League	Ernie Rodriguez

47: Solid HarmoniE

Music videos

Year	Title	Director(s)
1996	"Got 2 have ya"	—
1997	"I'll be there for you"	Gerry Wenner
1998	"I want you to want me"	Roger Pomphrey
1998	"I wanna love you"	Max & Dania
1998	"To love once again"	Paul Morgans

48: 2010–11 Greek Basket League

Greek League Best Five

Player	Team
Player	Team
Vassilis Spanoulis	Olympiacos
Dimitris Diamantidis	Panathinaikos
Rawle Marshall	PAOK
Mike Batiste	Panathinaikos
Ioannis Bourousis	Olympiacos

49: Lane County, Oregon

Presidential elections results

Year	Republican	Democratic
2012	36.7% 60,979	60.1% 99,891
2008	34.9% 63,835	62.3% 114,037
2004	40.4% 75,007	58.0% 107,769
2000	40.4% 61,578	51.6% 78,583
1996	34.5% 48,253	49.7% 69,461
1992	27.5% 41,789	48.8% 74,083
1988	39.7% 47,563	58.4% 69,883
1984	48.9% 61,493	50.9% 63,999
1980	43.6% 54,750	41.6% 52,240

50: Diving at the 1924 Summer Olympics – Men's 3 metre springboard

Final

Place	Diver	Points	Score
1	Albert White	7	696.4
2	Pete Desjardins	8	693.2
3	Clarence Pinkston	15	653
4	Edmund Lindmark	22	599.1
5	Dick Eve	26	564.3
6	Adolf Hellquist	30	544.9
7	Curt Sjöberg	34	538.3
8	Henk Hemsing	39	490.8
9	Julius Balasz	44	463.1

Bibliografía

Abadi, D., Ailamaki, A., Andersen, D., Bailis, P., Balazinska, M., Bernstein, P. A., Boncz, P., Chaudhuri, S., Cheung, A., Doan, A., et al. (2022). The seattle report on database research. *Communications of the ACM*, 65(8):72–79. 1, 4.1

Ahmadov, A., Thiele, M., Eberius, J., Lehner, W., and Wrembel, R. (2015). Towards a hybrid imputation approach using web tables. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, pages 21–30. IEEE. 2.1.2, 2.2.3, 2.3.1

Arevalo, J., Solorio, T., Montes-y Gómez, M., and González, F. A. (2017). Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*. 2.3.1

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155. 3.4

Bhagavatula, C. S., Noraset, T., and Downey, D. (2013). Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD workshop on interactive data exploration and analytics*, pages 18–26. 2.1.1, 2.2.2, 2.3.2

Bhagavatula, C. S., Noraset, T., and Downey, D. (2015a). Tabel: Entity linking in web tables. In *International Semantic Web Conference*, pages 425–441. Springer. 2.2.1, 4.2, 6.1

Bhagavatula, C. S., Noraset, T., and Downey, D. (2015b). Tabel: Entity linking in web tables. In *The Semantic Web - ISWC 2015*, pages 425–441, Cham. Springer International Publishing. 5.2.2

Bizer, C. and Schultz, A. (2009). The berlin sparql benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(2):1–24. 1

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017a). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146. 3.4, 4.2

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017b). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146. 3.4, 5.2.1

Bron, M., Balog, K., and Rijke, M. d. (2013). Example based entity search in the web of data. In *European Conference on Information Retrieval*, pages 392–403. Springer. 2.2.1

Cafarella, M. J., Halevy, A., and Khoussainova, N. (2009). Data integration for the relational web. *Proceedings of the VLDB Endowment*, 2(1):1090–1101. 2.1.1, 2.2.3

Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., and Zhang, Y. (2008). Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549. 2.1.1, 2.2.2, 2.2.2, 2.3.2

Chen, Z., Trabelsi, M., Heflin, J., Xu, Y., and Davison, B. D. (2020). Table search using a deep contextualized language model. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 589–598. 2.3.1

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. 3.4

Dai, Z. and Callan, J. (2019). Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 985–988. 1

Das Sarma, A., Fang, L., Gupta, N., Halevy, A., Lee, H., Wu, F., Xin, R., and Yu, C. (2012). Finding related tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 817–828. 2.1.2

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 3.4, 3.5, 4.2

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*. 3.4

Flores Herrera, J. d. J., Nadal Francesch, S., and Romero Moral, Ó. (2021). Towards scalable data discovery. In *Advances in Database Technology: EDBT 2021, 24th International Conference on Extending Database Technology: Nicosia, Cyprus, March 23-26, 2021: proceedings*, pages 433–438. OpenProceedings. 2.3.2

Gupta, R. and Sarawagi, S. (2009). Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment*, 2(1):289–300. 2.1.1

Gupta, S., Kanchinadam, T., Conathan, D., and Fung, G. (2020a). Task-optimized word embeddings for text classification representations. *Frontiers in Applied Mathematics and Statistics*, 5:67. 4.1

Gupta, S., Kanchinadam, T., Conathan, D., and Fung, G. (2020b). Task-optimized word embeddings for text classification representations. *Frontiers in Applied Mathematics and Statistics*, 5:1–10. 5.2.3

Halevy, A., Rajaraman, A., and Ordille, J. (2006). Data integration: The teenage years. In *Proceedings of the 32nd international conference on Very large data bases*, pages 9–16. 1

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162. 3.4

He, Y. and Xin, D. (2011). Seisa: set expansion by iterative similarity aggregation. In *Proceedings of the 20th international conference on World wide web*, pages 427–436. 2.2.1

Herzig, J., Nowak, P. K., Müller, T., Piccinno, F., and Eisenschlos, J. M. (2020). Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*. 1, 2.3.1

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. 3.2.2

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*. 3.5

Ilić, S., Marrese-Taylor, E., Balazs, J. A., and Matsuo, Y. (2018). Deep contextualized word representations for detecting sarcasm and irony. *arXiv preprint arXiv:1809.09795*. 3.4

Jurafsky, D. and Martin, J. H. (2018). Ch 24: Dialog systems and chatbots. speech and language processing: An introduction to natural language processing. *Computational Linguistics, and Speech Recognition, 3rd (draft) edition. Prentice Hall PTR, Upper Saddle River, NJ, USA*. 3.4

Khodabakhsh, M. and Bagheri, E. (2021). Semantics-enabled query performance prediction for ad hoc table retrieval. *Information Processing & Management*, 58(1):102399. 1

Konstantinou, N., Abel, E., Bellomarini, L., Bogatu, A., Civili, C., Irfanie, E., Koehler, M., Mazilu, L., Sallinger, E., Fernandes, A. A., et al. (2019). Vada: an architecture for end user informed data preparation. *Journal of Big Data*, 6(1):1–32. 1

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*. 3.5

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174. 6.4

Lehmberg, O., Ritze, D., Ristoski, P., Meusel, R., Paulheim, H., and Bizer, C. (2015). The mannheim search join engine. *Journal of Web Semantics*, 35:159–166. 2.1.2, 2.2.2, 2.3.1, 2.3.2

Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union. 4.1

Lilleberg, J., Zhu, Y., and Zhang, Y. (2015). Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE. 3.4

Limaye, G., Sarawagi, S., and Chakrabarti, S. (2010). Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3(1-2):1338–1347. 2.1.2

Liu, Q., Kusner, M. J., and Blunsom, P. (2020). A survey on contextual embeddings. *arXiv preprint arXiv:2003.07278*. 3.5

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*. 1

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 3.5, 4.2

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386. (document), 3.1, 3.2

McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12. 3.1

Metzger, S., Schenkel, R., and Sydow, M. (2013). Qbees: query by entity examples. In *Proceedings of the 22nd acm international conference on information & knowledge management*, pages 1829–1832. 2.2.1

Metzger, S., Schenkel, R., and Sydow, M. (2014). Aspect-based similar entity search in semantic knowledge graphs with diversity-awareness and relaxation. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 60–69. IEEE. 2.2.1

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 1, 3.4, 3.4

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, Red Hook, NY, USA. Curran Associates Inc. 5.2.1

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26. 4.1, 4.2

Miller, R. J. (2018). Open data integration. *Proc. VLDB Endow.*, 11(12):2130–2139. 1, 4.1

Mu, J. and Viswanath, P. (2018). All-but-the-top: Simple and effective postprocessing for word representations. In *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada. 4.1

Nargesian, F., Zhu, E., Pu, K. Q., and Miller, R. J. (2018). Table union search on open data. *Proceedings of the VLDB Endowment*, 11(7):813–825. 2.1.2, 2.3.1, 4.2, 4.2.1.1, 4.2.2

Neumaier, S., Umbrich, J., and Polleres, A. (2016). Automated quality assessment of metadata across open data portals. *Journal of Data and Information Quality (JDIQ)*, 8(1):1–29. 1

Nguyen, T. T., Nguyen, Q. V. H., Weidlich, M., and Aberer, K. (2015). Result selection and summarization for web table search. In *2015 IEEE 31st International Conference on Data Engineering*, pages 231–242. IEEE. 2.1.2, 2.3.1

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. 3.4, 3.4

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations. arxiv preprint arxiv: 180205365. 3.5

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (1802). Deep contextualized word representations. corr abs/1802.05365 (2018). *arXiv preprint arXiv:1802.05365*. 3.4

Peters, M. E., Neumann, M., Zettlemoyer, L., and Yih, W.-t. (2018b). Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*. 1

Pimplikar, R. and Sarawagi, S. (2012). Answering table queries on the web using column keywords. *arXiv preprint arXiv:1207.0132*. 2.1.1

Plaza-del Arco, F. M., Molina-González, M. D., Urena-López, L. A., and Martín-Valdivia, M. T. (2021). Comparing pre-trained language models for spanish hate speech detection. *Expert Systems with Applications*, 166:114120. 1

Qu, C., Yang, L., Qiu, M., Croft, W. B., Zhang, Y., and Iyyer, M. (2019). Bert with history answer embedding for conversational question answering. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 1133–1136. 1

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training. 3.4

Rastan, R., Paik, H.-Y., and Shepherd, J. (2019). Texus: A unified framework for extracting and understanding tables in pdf documents. *Information Processing & Management*, 56(3):895–918. 1

Reimers, N. and Gurevych, I. (2019a). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*. 3.5

Reimers, N. and Gurevych, I. (2019b). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics. 5.2.1

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995). Okapi at trec-3. *Nist Special Publication Sp*, 109:109. 4.2.1.1

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523. 3.4

Samuel, A. L. (1988). Some studies in machine learning using the game of checkers. ii—recent progress. *Computer Games I*, pages 366–400. 3.1

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. 3.5

Sarma, A. D., Fang, L., Gupta, N., Halevy, A. Y., Lee, H., Wu, F., Xin, R., and Yu, C. (2012). Finding related tables. 2.1.2, 2.2.1, 2.2.2, 4.1

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*. 3.5

Shih, K. J., Singh, S., and Hoiem, D. (2016). Where to look: Focus regions for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4613–4621. 3.4

Shraga, R., Roitman, H., Feigenblat, G., and Cannim, M. (2020). Web table retrieval using multimodal deep learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1399–1408. 2.3.1

Trabelsi, M., Davison, B. D., and Heflin, J. (2019). Improved table retrieval using multiple context embeddings for attributes. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1238–1244. IEEE. 2.3.1

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188. 3.4

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2019). Attention is all you need. *Advances in neural information processing systems*, 30. (document), 3.2.3, 3.3

Venetis, P., Halevy, A. Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., and Miao, G. (2011). Recovering semantics of tables on the web. 2.2.1

Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. (2019). Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics. 5.1

Wang, C., Chakrabarti, K., He, Y., Ganjam, K., Chen, Z., and Bernstein, P. A. (2015). Concept expansion using web tables. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1198–1208. 2.2.1, 2.3.2

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*. 3.5

Yakout, M., Ganjam, K., Chakrabarti, K., and Chaudhuri, S. (2012). Infogather: entity augmentation and attribute discovery by holistic

matching with web tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 97–108. 2.1.2, 2.2.1, 2.2.2, 2.2.3, 2.3.1

Yin, P., Neubig, G., Yih, W.-t., and Riedel, S. (2020). Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*. 1, 2.3.1

Zhang, L., Zhang, S., and Balog, K. (2019). Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1029–1032. 2.1.1, 2.2.1, 2.2.2, 2.3.2, 4.1

Zhang, M. and Chakrabarti, K. (2013). Infogather+ semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 145–156. 2.1.2, 2.2.3, 2.3.1

Zhang, S. and Balog, K. (2017a). Design patterns for fusion-based object retrieval. In *European Conference on Information Retrieval*, pages 684–690. Springer. 2.1.1

Zhang, S. and Balog, K. (2017b). Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 255–264. 2.2, 2.2.1, 2.2.2, 2.3.2

Zhang, S. and Balog, K. (2018). Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 world wide web conference*, pages 1553–1562. 2.1, 2.1.1, 2.3.1

Zhang, S. and Balog, K. (2019). Auto-completion for data cells in relational tables. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 761–770. 2.2.3

Zhang, S. and Balog, K. (2020). Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–35. 2, 2.1

Zhang, X., Ramachandran, D., Tenney, I., Elazar, Y., and Roth, D. (2020). Do language embeddings capture scales? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4889–4896, Online. Association for Computational Linguistics. (document), 2.1, 2.2, 2.3, 2.4, 5.1