Neural Computing and Applications (2023) 35:9319–9333
https://doi.org/10.1007/s00521-022-08066-8

S.I.: LATIN AMERICAN COMPUTATIONAL INTELLIGENCE

Contextual word embeddings for tabular data search and integration

José Pilaluisa[1] · David Tomás[2] · Borja Navarro-Colorado[2] · Jose-Norberto Mazón[2]

Received: 1 April 2022 / Accepted: 16 November 2022 / Published online: 30 November 2022

Abstract
This paper presents a new approach to retrieve and further integrate tabular datasets (collections of rows and columns) using union and join operations. In this work, both processes were carried out using a similarity measure based on contextual word embeddings, which allows finding semantically similar tables and overcome the recall problem of lexical approaches based on string similarity. This work is the first attempt to use contextual word embeddings in the whole pipeline of table search and integration, including for the first time their use in the join operation. A comprehensive analysis of their performance was carried out on both retrieving and integrating tabular datasets, comparing them with context-free models. Column headings and cell values were used as contextual information and their impact on each task was evaluated. The results revealed that contextual models significantly outperform context-free models and a traditional weighting schema in ad hoc table retrieval. In the data integration task, contextual models also improved the results on union operation compared to context-free approaches.

Keywords Tabular data · Contextual word embedding · Information search · Data integration · Open data

1 Introduction

The goal of data science is to move from raw data to knowledge through different steps, including data integration, wrangling, analysis, and visualisation. Importantly, data integration comprises more than 80% of the effort for a data scientist [1] due to its complex nature. This task involves several types of transformations (e.g. cleansing, combination or normalisation) that uses operations (e.g. join or union) to offer a unified view of a set of heterogeneous data from different sources. In such a scenario, it is essential for data scientists to have tools to support the integration process.

Data integration is even more complex within an open data scenario, since the data to be consumed are not known a priori and more effort is required to retrieve and understand them. Moreover, most open data available on the Web are published in open data portals as tabular data (i.e. CSV files). Tabular data is a common way of storing, classifying and disseminating information. Although tables are widely used, it is difficult to extract and manipulate automatically their content. Using previously unknown tabular open data is therefore a time-consuming task for data scientists, who have to properly understand the data (e.g. finding out their relationships) before retrieving other data that are potentially useful for integration.

This paper presents an application of neural network language models to support data scientists in finding

✉ David Tomás
dtomas@dlsi.ua.es

José Pilaluisa
jpilaluisa@uce.edu.ec

Borja Navarro-Colorado
borja@dlsi.ua.es

Jose-Norberto Mazón
jnmazon@dlsi.ua.es

[1] Faculty of Engineering, Physical Sciences and Mathematics, Central University of Ecuador, Avenida Universitaria, 170129 Quito, Ecuador

[2] Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n, 03690 San Vicente del Raspeig, Spain

relationships between data in order to retrieve and integrate relevant tabular data. The method proposed involves two steps. In the first one, the objective is to retrieve tabular datasets[1] that have similar columns and are therefore more likely to be integrated. Once this set of similar tables is obtained, the second step aims to determine which specific columns should be involved in the final integration process. The present study is focused on union and join operators, since they are specially relevant for data integration, facilitating the consumption of data from disparate sources [2]. It is interesting to note that considering these operators provides the basis for including others widely used in data integration, such as filtering and sorting.

To carry out the retrieval and integration processes, a (semantic) similarity measure between tabular data has been defined. This measure leverages contextual word embeddings [3] to compute the similarity between tables using column headings and cell contents. Word embeddings is a technique that allows mapping words or phrases to vectors of real numbers, capturing semantic regularities in vector spaces. When comparing two terms, word embeddings overcome the problems of lexical approaches based on string similarity: terms such as "city" and "location" could be considered as being very different in terms of string matching, but in a word embedding space these two terms may be closely related and considered as highly similar.

Traditional word embeddings provide the same vector representation for a word regardless of the context in which it occurs. Unlike these static embeddings, recent contextual models capture the different meanings of a word, providing different vector representations depending on the surrounding context. For instance, in the case of tabular data, the column header "place" should have a different representation if it is surrounded by the words "athlete" and "time" (it refers to the position in the final ranking), or if it appears next to "coordinates" and "postal code" (in which case it refers to a location). Another example is the occurrence of "Francis Bacon" in the cell content. The vector representation would be different if it appears in a column with "Jackson Pollock" and "Pablo Picasso" (Francis Bacon the artist), or with "Thomas Malet" and "Samuel Eyre" (Francis Bacon the statesman).

Incorporating context into word embeddings has led to significant improvements in virtually every Natural Language Processing (NLP) task, representing the current state-of-the-art in the area. Recently, these models are also been applied to table retrieval and, to a lesser extent, to tabular data integration. This work presents the first attempt to provide a pipeline for tabular data search and integration including contextual word embeddings in all its phases. The main objectives of this paper are:

- To conduct a comprehensive analysis of the performance of neural networks-based contextual word embeddings in the task of retrieving and integrating (union and join operations) tabular data.
- To compare the performance between contextual and context-free models to determine the impact of context in these tasks.

The rest of the article is structured as follows: Sect. 2 presents previous works in the field of table retrieval and integration using word embeddings; Sect. 3 describes the main features of contextual models; Sect. 4 shows the procedures and measures proposed to implement the retrieval and integration processes; Sect. 5 reports the evaluation carried out; Sect. 6 discusses the main outcomes of the experiments; conclusions and future work are presented in Sect. 7.

2 Related work

This section summarises existing work in the area of table retrieval and integration, with a focus on systems that make use of word embeddings.

The goal of table retrieval (or table search) is to answer a search query with a ranked list of tables that are considered as relevant to that query [4]. It is an important task on its own, but also a fundamental step in table integration [5].

Depending on the type of query, table retrieval may be classified as keyword-based or table-based search [5]. In the former, a set of keywords form the query, as is the case with traditional search engines such as Google. In the latter, the query is also a table, and the goal is to compute a similarity score between the input and candidate tables.

The work in [6] is an example of a keyword-based table retrieval system that uses table metadata to create its vector representation, using a TF-IDF model to further retrieve relevant tables.

Since in the present work the goal is to integrate tabular data, tables are used as queries to retrieve related tables that are suitable candidates for further integration, performing a table-based search on the dataset. For this reason, the remainder of this section focuses on this category. The approaches presented differ mainly in what information from tables is used to calculate the similarity between them (table title, table entities, column headings or cell values), in the formal representation of the data, and in the similarity measures used.

An early approach was presented by Ahmadov et al. [7], which only used named entities (e.g. people, organisations,

[1] The proposal can be extended to any tabular format, although only CSV files were used in the experiments.

and locations) and column headings. More recent approaches relied on semantic spaces to represent tabular data, applying vector similarity measures to calculate the relevance between tables [8]. In this vein, Zhang and Balog [4] combined two semantic vector spaces: one based on a knowledge base (DBpedia) and the other using pre-trained word embeddings (Word2vec). They used all the information available in the tables for the retrieval task (e.g. title, caption, headings, and entities).

The work in [9] also used Word2vec as the source for semantic vectors. The information of the table was separated in four semantic spaces: description (title and caption), schema (column headings), records (table rows), and facets (table columns). Then, different neural network architectures were applied to each semantic space, including recurrent convolutional neural network (description), multilayer perceptron (schema), and 3D convolutional neural network (records and facets).

To retrieve tables compatible with an input table, Nargesian et al. [10] tried to estimate if the table contents belonged to the same domain. They applied three statistical models: intersecting values between two columns, semantic similarity between values mapping the columns to classes in an ontology, and using word embeddings to measure similarity between textual values.

All the word embedding models mentioned above are non-contextual. The works presented in the following paragraphs use contextual word embeddings for table-based search. In [11] a survey on contextual embeddings is presented.

In [12], the authors used a pre-trained version of BERT [13], leveraging different information available in the table (both textual and numerical) to provide BERT with context: title, caption, column headings, and cell values. An important difference between the present work and that of Chen et al. [12] is the purpose of the table retrieval task. In their case, it was a goal in itself, while in this work it is a step prior to integration. Another difference, is that we implement two different relevance measures, depending on whether the final goal is to integrate tables by means of join or union operations. Finally, another novelty in this paper is the experimentation with a fine-tuned contextual language model. Although there are recent proposals of contextual word embedding models trained on tabular data [14, 15], they are focused on answering natural language questions from tables, but do not address the retrieval and integration of this type of data.

Focusing on table retrieval, in [16] tables were considered as 2D images, and data were then handle by traditional neural approaches to image processing (e.g. CNNs). Another image-based neural representation approach was presented in [17], where an image-based method was combined with a graph-based approach in order to get the best out of each one. The authors proposed to use WordNet structure as a graph, in such a way that cell texts (tokens) from the table were represented by their synsets in WordNet. With this information, they built a graph that captured lexical similarities between text cells. However, with this approach it is not possible to represent contextual and distributional relations between data.

The aforementioned approaches considered tables as a whole. In the present work, the goal of data integration requires a more fine grained representation, working at column level, where each one is represented as a vector, as described in Sect. 4.

An alternative to represent tabular data is the use of graph-based approaches. The work in [18] represented data from relational databases as a graph with three types of nodes: data (token), column id (header) and record id (file name). A sentence was derived from these relations and a contextual vector was obtained for these sentences.

The representation model for tabular data in [19] is similar to that presented here, where each column is encoded by a transformer model. They focused only on the representation of data, whereas in the present work BERT is fine-tuned with tabular data and further applied to table retrieval and integration.

Moving forward in the data integration pipeline, once a set of similar tables has been retrieved, the next step is to determine whether it is possible to extend the query table with compatible rows (union) or new columns (join) [5].

Row extension is mainly done in relational tables, in which there is a core column with the key concepts of the table and the remaining columns contain the attributes. This way, the row extension task is similar to concept expansion, where an initial set of entities has to be completed with additional entities [5]. Previous approaches to row extension have used some sort of similarity between tables to find their compatibility. For example, Wang et al. [20] introduced concept names as input, together with seed entities to prevent lexical ambiguity. As far as the authors know, only the work by Deng et al. [21] previously used word embeddings (Word2vec) in this task.

In the area of *column extension* (also known as *attribute discovery*), the approach presented in [22] was based on a database that included frequency statistics of attributes and co-occurring attribute pairs in a large table corpus (5.4 million unique attribute names). In this task, the authors of [23] took advantage of table captions and similarity between tables. These two studies were focused only on column headings and were not aimed to carry out join operations between tables. The approach proposed in [24] was based on Wikipedia tables. The relatedness between tables was estimated based on the link intersections of Wikipedia pages.

On the basis of previous work, the present study is the first attempt to widely use and analyse the impact of contextual word embeddings in the task of tabular data integration. The following section describes the particularities of these vector representations and how they differ from traditional context-free word embeddings.

3 Contextual word embeddings

Word embeddings are dense vectors that represent the meaning of a word as a point in a semantic space. These continuous representations can be used in downstream NLP tasks, such as text classification and question answering. They represent the distributional meaning of words, that is, the meaning that a word assumes in a specific text regardless of the meaning it may have in the dictionary. Thus, similar representations are learnt from words appearing in similar contexts.

The dimensionality of word embedding vectors usually ranges from 50 to 1000 dimensions, much lower than that of traditional sparse semantic vectors. This reduction in dimensions is based on generalisations that capture the semantic relations between words based on the context they appear. Examples of this type of word representations are Word2vec [25] and Glove [26].

These word embedding techniques build a global vocabulary using unique words in the documents, assigning a single representation for each word and ignoring that they can have different meanings or senses in different contexts. They are considered as *static* representations unable to capture the different senses of a word. On the other hand, recent contextual word embeddings [13] are able to capture the different meanings of polysemous words, since each vector represents not a word but a sense. In this way, each word is represented with different word embeddings, one for each context in which the word can occur. During the training process, contextual word embeddings are generated taking into consideration the surrounding words, that is, the sequence of words in the sentence or text span in which a word appears. Examples of these type of representation are ELMo [27], ULMFit [28] and BERT [13], among others.

In the experiments presented in Sect. 5, BERT and its variant RoBERTa [29] were used as representative of these contextual models. BERT stands for Bidirectional Encoder Representations from Transformers. Bidirectional means that BERT learns information from both left and right side of a word's context during the training phase. It uses the now ubiquitous transformer architecture [30] allowing transfer learning in NLP tasks, i.e. the BERT model originally trained on a dataset (the *pre-trained model*) can be used to perform similar tasks on another dataset (the *fine-tuned* model).

In this way, BERT, which was pre-trained on a large corpus of unlabelled text (including the entire Wikipedia and Book Corpus), can be fine-tuned for a wide range of NLP tasks. Current NLP state-of-the-art systems leverage the semantic relationships identified by transformers as a starting point to solve a problem rather than building a model from scratch, further training the model (fine-tuning) on relatively smaller datasets for specific tasks.

Finally, another remarkable difference between these contextual models and its static predecessors is the use of subword units instead of full words to represent the vocabulary of the problem. Word embeddings are built on the specific set of tokens available in the corpus used to create the vectors. When an out-of-vocabulary word occurs in a new text, word-based models provide no representation for it in the semantic space and thus the token is considered as *unknown*. In order to handle the large vocabularies common in natural language corpora, BERT uses the WordPiece subword segmentation algorithm [31]. In it, the vocabulary is initialised with individual characters in the language and then the most frequent combinations of symbols in the vocabulary are iteratively added to the vocabulary. Thus, subwords have their own representation in semantic space, and previously unknown words can be assigned a representation by combining the vectors of their underlying subword units.

Regarding RoBERTa, this model provides a variant of BERT where the pre-training phase was optimised with changes in the choice of hyperparameters, the objective task, and the use of bigger batch sizes and longer text sequences. Besides that, RoBERTa uses a different segmentation algorithm, Byte-Pair Encoding (BPE) [32], a hybrid between character-level and word-level representations that relies on subwords units extracted by performing statistical analysis of the training corpus. This changes led to improving the results of BERT in different NLP tasks, such as natural language understanding and question answering.

4 Retrieval and integration of tabular data

This section formally describes how table retrieval and integration are carried out using word embeddings.[2] In the case of integration, the operators considered in this work for handling input tabular data are union and join from relational algebra. For the sake of simplicity, in this article these operators are borrowed from SQL, the well-known

[2] The source code is available at: https://github.com/d-tomas/data-integration.

implementation of relational algebra and a recognised standard for querying and handling tabular data:

- Union operator is denoted by $\cup$ symbol in relational algebra. Given two tabular datasets (A and B), union operator aims to get a unique dataset that contains rows that are in A or in B (denoted as A $\cup$ B). A and B must have the same columns (number, order, and datatype) to be computed. Also, each column of each dataset must refer to the same concept to be meaningful.
- Join operator is denoted by $\bowtie$ symbol in relational algebra. Given two tabular datasets A and B, join operator aims to get a unique dataset that includes every column from A and B (denoted as A $\bowtie$ B) and contains rows that fulfil a matching condition (applied to values of some columns).

Figure 1 summarises the workflow and the components of the system proposed. To determine the most relevant tables for a search, and whether union or join operations can be performed, the system proposed relies on word embeddings to calculate the semantic similarity of two tabular datasets. This similarity mechanism takes as an input a set of tabular data and compares with each other all the columns of the tables, obtaining a similarity measure for each pair of columns belonging to different tables. To calculate this similarity two elements are taken into account: name of the columns (headings) and their content for each row (cell values).

This information is pre-processed and normalised by splitting CamelCase and hyphenated words (very common in column headings), removing punctuation, and converting text to lowercase. After that, the word embedding model is used to extract two vectors for each column: one represents the name of the column and the other the cell values of that column. In those situations where the name of the column includes more than one word, vectors representing each word are averaged to get a single vector. Averaging word embeddings is one of the most popular methods for combining embedding vectors, outperforming more complex techniques especially in out-of-domain scenarios [33]. The same strategy is applied to cell values, where the final vector is the result of calculating the mean between vectors of each value.

As in previous works, cosine similarity is used to compute the distance between vectors in the embedding space [25]:

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

where $\mathbf{v}_1$ and $\mathbf{v}_2$ represent the word embedding vectors of the name of the columns or the content of the column for each row, and $sim(\mathbf{v}_1, \mathbf{v}_2)$ is a float value in the range $[-1, 1]$, where -1 means no similarity and 1 means maximum similarity between the vectors considered. Negative cosine values are possible because word embedding vectors can contain negative elements. Nevertheless, in all the experiments carried out the cosine similarity obtained was always a positive value. This result is consistent with previous research suggesting that word vectors are not balanced around the origin in the semantic space, showing fewer negative cosine similarities than expected from points in a random n-sphere [34].

In the case of static word embeddings, if the model does not provide coverage for column headings or its content (i.e. their tokens are not in the vocabulary of the model), the Levenshtein distance [35] is used as a backup strategy to ensure that the system always returns a similarity value between columns. This string metric is based on the number of single-character edits (insertions, deletions or substitutions) required to change one string into the other.

For each two columns compared, a similarity value of the column headings and a similarity value of their content is obtained. A linear combination of these two values is performed to obtain the final score $simC$ of two columns c_1 and c_2:

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

where $sim(\mathbf{c}_{1n}, \mathbf{c}_{2n})$ is the cosine similarity of the names of the columns, $sim(\mathbf{c}_{1c}, \mathbf{c}_{2c})$ is the cosine similarity of their contents, and α is a parameter in the range [0, 1] that weights the relevance of these two scores in the final result.

For the table retrieval task, two different objectives have been defined depending on whether the goal is to further union the tables or join them. In the evaluation carried out in Sect. 5, the task of finding the best tables for union and join operations is assimilated to the task of *ad hoc* table-based retrieval: answering a search query with a ranked list of tables [21], where the search query is not a sequence of keywords but a table itself [36]. It is thus necessary to define a ranking function that sorts tables according to the goal, so that most relevant results appear early in the list retrieved.

If the goal is retrieving tables for union operation, the ranking criterion established tries to prioritise tables with a large number of highly similar columns, which are hence good candidates for union. Thus, given a set of tables T, the similarity $simTU(t_1, t_2)$ for every pair $t_1, t_2 \in T$ is computed as:

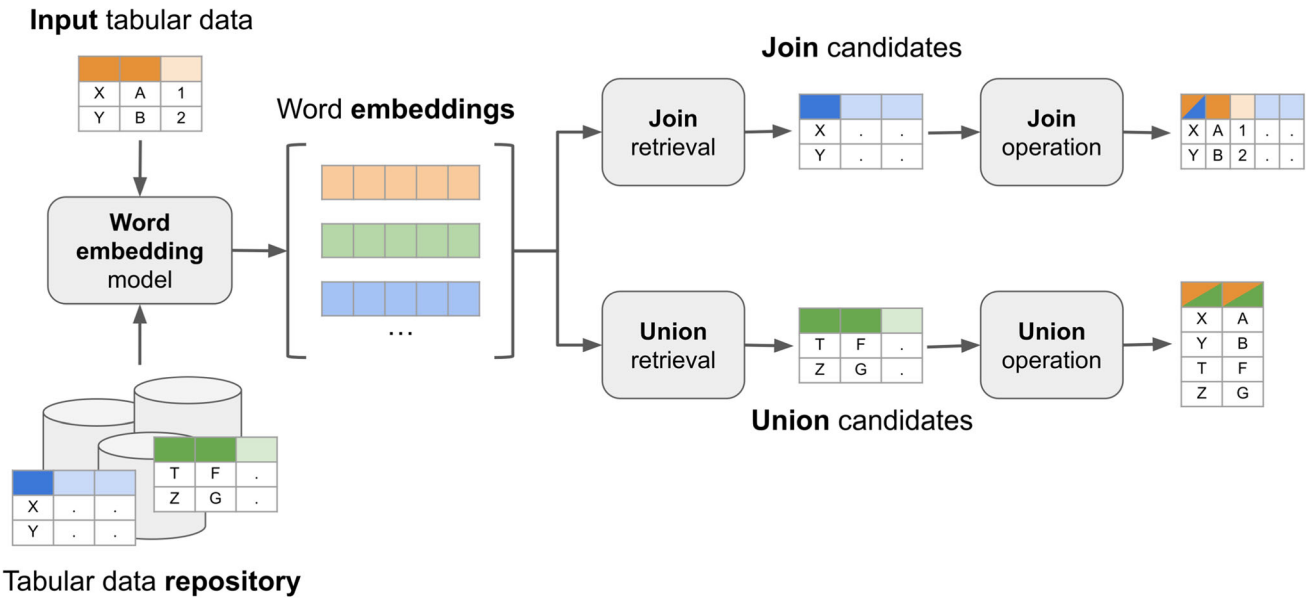

Fig. 1 Components and workflow for the union and join operations

$$simTU(t_1, t_2) = \frac{\sum_{i=1,j=1}^{i \leq n, j \leq m} simC(c_{1i}, c_{2j})}{\|C_1\|\|C_2\|}, \tag{3}$$

where $C_1 = \{c_{11}, c_{12}..., c_{1n}\}$ and $C_2 = \{c_{21}, c_{22}..., c_{2m}\}$ are the set of columns of t_1 and t_2, respectively. Therefore, the similarity between two tables is computed as the average similarity between their columns (*simC*).

In the same vein, it is necessary to define a ranking function to retrieve tabular data for the join operation. In this case, the similarity between two tables is computed as:

$$simTJ(t_1, t_2) = \max_{i \leq n, j \leq m} \{simC(c_{1i}, c_{2j})\}. \tag{4}$$

This formula weights two tables based on the pair of columns with the highest *simC* similarity, since for join operations the goal is to find tables containing columns that can be matched with a high probability (i.e. key columns to perform join) while the remaining columns may not match.

In the integration pipeline, once the relevant tables have been retrieved, the system has to indicate which specific columns are candidates to be combined using the union or join operations. To decide whether the operation of union can be applied on two columns, a similarity threshold is established at which it is considered that the operation can be carried out. Thus, union can be applied to every pair of columns over this threshold. Similarly, the join is performed by establishing a threshold to decide if the operation can be applied on two columns. The impact of these threshold values in the performance of the system is also analysed in the evaluation section.

5 Evaluation

This section describes the evaluation carried out, which covers the following aspects of contextual word embeddings: (i) their performance in the task of retrieving relevant tables for union and join operations (Sect. 5.1), (ii) their performance in identifying columns from two tables that can be combined in union operations (Sect. 5.2), and finally (iii) same as before but with respect to join operations (Sect. 5.3).

The dataset used in these experiments was developed by Nargesian et al. [10] and it is publicly available. It was originally intended for table union search, but in the following experiments it has been adapted to also evaluate join operations. This dataset consists of more than 5000 tables in CSV format extracted from USA, Canada, and UK open data portals, providing a ground truth that identifies which columns of a table match the columns of another table. The dataset was built starting with 32 base tables manually aligned to identify matching columns. The final set was created by first issuing a projection on a random subset of columns of a base table, and then a selection with some limit and offset on the projected table.

These tables contain the column headings and the corresponding cell values, comprising text, numeric, and date values. Although word embedding models are specially suitable for textual data, they also provide coverage for columns containing other data types. First, the names of the columns are textual data, even if their contents are numbers or dates. Thus, the system can always return a similarity value based on the column names. Secondly, even if only the content of the columns is considered, the word

embedding models still provide coverage for many numeric values that are represented in the embedding space. For instance, the fastText model described below provides coverage for 99.90% of the numbers ranging from 0 to 10,000. This implies that any numeric value used to represent days, months or years has a vector representation in this model. Moreover, in the case of contextual models, the use of BPE and WordPiece subword segmentation algorithms always provides a vector representation for any numeric or data value found in text.

To perform the experiments, a subset of 1,000 tables was randomly selected. Every table in this subset was used as a query to the system and compared with all the other tables in the subset. This led to a total of 499,500 pairs of tables. For each pair, all their columns were compared with each other, obtaining 3,249,440 column comparisons, an average of 6.5 comparisons for each pair of tables.

The experiments carried out in the next sections test the performance of two pre-trained non-contextual word embedding models (Word2vec and fastText), two pre-trained contextual models (BERT and RoBERTa) and one task-specific (fine-tuned) contextual model (WikiTables):

- Word2vec: embedding vectors pre-trained on part of Google News dataset, comprising about 100 billion words [25]. The model contains 300-dimensional vectors for 3 million words and phrases, although in the experiments presented here only words were considered.
- fastText: embedding vectors pre-trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset, comprising about 16 billion words [37]. As in the previous case, vectors have 300 dimensions.
- BERT: the version of the model evaluated is BERT-base, containing 12 layers (transformer blocks), 12 attention heads, and 110 million parameters. The resulting vectors are composed of 768 dimensions [13].
- RoBERTa: the version evaluated is RoBERTa-base, containing 12 layers, 12 attention heads, 125 million parameters, and producing vectors of 768 dimensions [29].
- WikiTables: task-specific model obtained by fine-tuning BERT-base on the Wikipedia table corpus, which contains 1.6 million Wikipedia relational tables [38].

As mentioned before, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks without substantial task-specific architecture modifications. To build the WikiTables model, the Wikipedia table corpus was pre-processed splitting CamelCase and hyphenated words, removing punctuation, and converting text to lowercase. For every table in this corpus, all the column headings were extracted and treated as an input document to fine-tune BERT. A second model was created for the cell values. In this case, the content of column was considered as an input document to train the model. Thus, two separate word embedding models are used to calculate the similarity between column headings and cell values. As in BERT-base, vectors have 768 dimensions.

The implementation of Word2vec and fastText was carried out with the Gensim library.[3] The contextual models were implemented using the Transformers library developed by Huggingface.[4]

5.1 Table retrieval

This section describes the evaluation of the embedding models in retrieving the most relevant tables for a given one. As described in Sect. 4, two different objectives have been defined depending on whether the goal is to further union the tables (Sect. 5.1.1) or join them (Sect. 5.1.2).

The precision of the models (fraction of relevant instances among retrieved instances) was measured on the top-k tables retrieved at different k values, a procedure widely used in information retrieval systems on the web [39]. More specifically P@1, P@10, and P@50 where computed, corresponding to the proportion of relevant results obtained in the first table retrieved, in the top 10 tables, and in the top 50 tables respectively. These thresholds are in accordance with web-scale information retrieval systems, where thousands of relevant documents are available but no user is interested in reading all of them. The 1,000 queries selected for the experiments were randomly split into 10 subsets, carrying out 10 runs to calculate the average precision, standard deviation, and two-tailed independent t-test ($\rho < 0.01$) of the results obtained.

5.1.1 Union retrieval

The goal of this task is to retrieve the most appropriate tables to perform union operations for a given table. The ranking criterion was described in Eq. 3.

In the experiments presented, a ranked table is considered to be relevant if it contains at least one column that could be aligned with another column of the query table in a union operation, as specified in the ground truth provided in [10]. Although the union retrieval similarity function in Eq. 3 prioritise tables with a large number of highly similar columns, the criterion established here takes into account that having just one similar column can be enough for the subsequent union operation. According to this criterion, out

[3] https://radimrehurek.com/gensim/.

[4] https://github.com/huggingface/transformers/.

of 499,500 table pairs, 22,824 were considered as relevant (4.6% of the total).

In addition to the word embedding models described at the beginning of this section, a baseline was defined using BM25 [40], a classical ranking function widely employed by search engines to estimate the relevance of documents to a given search query. The implementation of this baseline was done using Apache Lucene.[5]

Each embedding model was evaluated using different values of α (from 0 to 1 inclusive, in 0.1 increments), as described in Eq. 2, to analyse the influence of the column headings and the cell values in the performance of the system. In the case of BM25, three different queries were employed in Apache Lucene to simulate the ranking experiment done with the embedding vectors: a query that uses only cell values (equivalent to $\alpha = 0.0$), a query that uses only names of the columns (equivalent to $\alpha = 1.0$), and a query considering both (equivalent to $\alpha = 0.5$).

Table 1 shows P@10 for the two non-contextual models (Word2vec and fastText) and the three contextual models (BERT, RoBERTa and WikiTables). In this experiment, BM25 scored 0.8245 ($\alpha = 0.0$), 0.8349 ($\alpha = 0.5$) and 0.8612 ($\alpha = 1.0$).

The results are encouraging in terms of performance of contextual models. The best precision for non-contextual models was obtained by fastText (0.9276, $\alpha = 0.3$). This result improves 7.71% the best performance of BM25 (0.8612, $\alpha = 1.0$). On the other hand, the best contextual model was BERT, with 0.9737 precision ($\alpha = 0.3$). The result improves 5% the best non-contextual model and 13% the baseline. According to t-test, these differences are statistically significant. The best result by RoBERTa was 0.9363 ($\alpha = 0.1$) and the best for WikiTables was 0.9691 ($\alpha = 0.5$). In the later, the difference with BERT was not statistically significant. Performance of WikiTables is the most stable regarding variations in the parameter α, with a standard deviation $\sigma = 0.0175$, followed by BERT ($\sigma = 0.0195$). At the other end of the scale is RoBERTa, with a sharp drop in performance in the α range [0.3, 0.4] and a standard deviation $\sigma = 0.0585$.

P@1 puts the focus on how precise the system is in returning a relevant table at the first position of the ranking. Figure 2a shows that all three contextual models surpassed the context-free models and the baseline, being these results statistically significant. The best performing model was WikiTables (0.9965, $\alpha = 0.4$), showing high reliability in this task. This model improves 12% and 23% the best results obtained by fastText (0.8888, $\alpha = 0.4$) and Word2vec (0.8129, $\alpha = 0.4$), respectively.

Figure 2b shows the results for P@50. The performance of WikiTables (0.7230, $\alpha = 0.6$) and BERT (0.7139, $\alpha = 0.2$) was very similar, with no statistically significant difference. On the contrary, the difference with fastText (0.6932, $\alpha = 0.4$), the best non-contextual model, is statistically significant. The performance achieved was systematically lower for every model than that obtained with P@10 as expected, since increasing the number of tables retrieved also increases the likelihood of including false positives. The performance of BM25 (0.6874, $\alpha = 1.0$) is remarkable in this experiment, achieving better results than non-contextual models and close to contextual ones.

The work in [10] also addressed the task of union retrieval in the same dataset used in this section. In their case, the authors tried to maximise the performance on their own definition of "unionability" (involving all the unionable columns of a table), whereas the goal in the current work is to address both union and join operations where tables have at least one matching column in common. Although these differences make the performance of the systems not directly comparable, their results are shown here as a reference: their system obtained a P@10 value around 0.8 with an approach based on non-contextual word embeddings, which improved to 0.95 when combined with semantic information from YAGO.[6] In Table 1, the highest result obtained by BERT ($\alpha = 0.3$) was 0.9737.

To further investigate similarity scores, the mean similarity between tables assigned by different models and α values were computed. Figure 3a shows that RoBERTa assigns significantly high similarity scores, averaging 0.9554 for all α values. On the other extreme is Word2vec, with an average similarity of 0.3259.

This result also reveals that mean similarity is higher on average for lower α values, that is, when more weight is given to cell values than to column headings. This implies that computing the embedding vectors from many different tokens (i.e. all the values of a column) increases the average similarity between tables, although not equally for all models. The model with the highest variability is BERT, with a difference of 0.2858 points between the highest (0.8030, $\alpha = 0.0$) and the lowest (0.5172, $\alpha = 1.0$) mean. The most stable model is WikiTables, with a difference of 0.0310 points between the highest and lowest mean.

Figure 3b shows in more detail the central tendency and dispersion of similarity values assigned by BERT. The box plots indicate that increasing α also increases the dispersion, as indicated by the difference between the maximum and minimum values (outliers are excluded in the figure). Therefore, weighting cell values over column headings not only increases the similarity score, but also reduces variability. Intuitively, this may lead BERT to a loss of performance for low α values, since many of the tables obtain

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

[6] https://yago-knowledge.org/.

Table 1 Mean (M) and standard deviation (SD) of P@10 for word embedding models in union retrieval

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

The best result for each α value is bold-faced

* Indicates statistically significant improvement ($\rho < 0.01$) of the contextual model with respect to the best non-contextual model for the corresponding α value

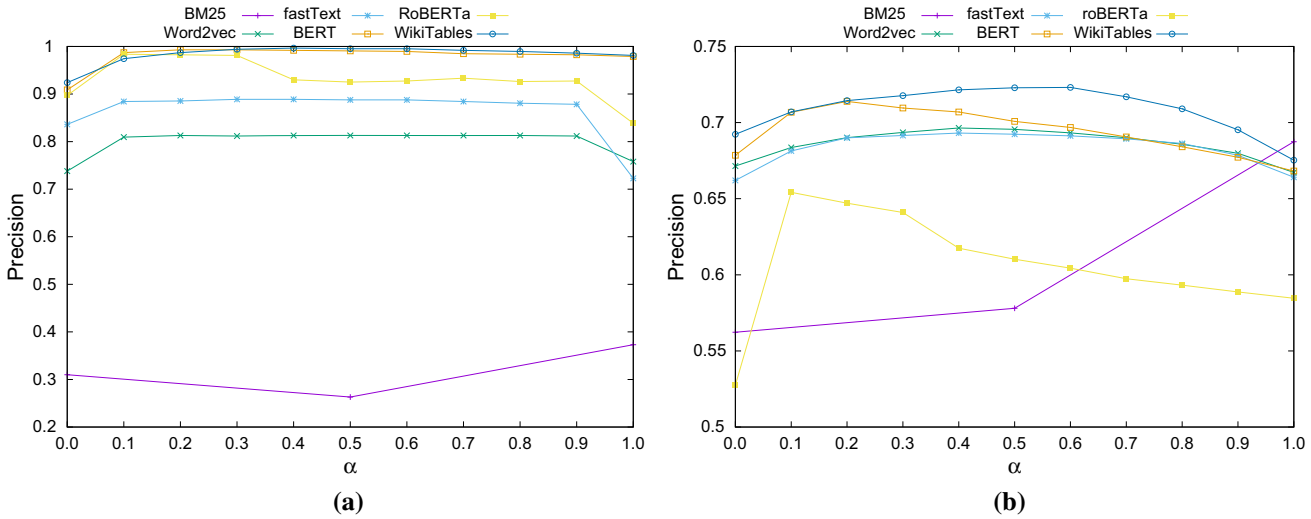

Fig. 2 P@1 (**a**) and P@50 (**b**) for BM25 and word embedding models in union retrieval

close similarity scores (as reflected by the low variability). However, the performance of BERT in Table 1 and Fig. 2 apparently shows no correlation with the mean similarity in Fig. 3a based on α. In the case of P@10, the Spearman correlation coefficient r_s between performance and average similarity is 0.3, which can be considered as a weak correlation.

5.1.2 Join retrieval

The objective in this experiment is to retrieve the most relevant tables to perform join operations. The ranking function for this task is defined in Eq. 4.

The ground truth used in the union retrieval experiment only identifies on which columns the union operation can be applied. The way in which the previous dataset was obtained has been leveraged here to also evaluate the models in join retrieval. As mentioned above, tables were obtained by projection and selection of 32 original tables manually aligned. On this basis, the criterion to identify whether two tables from the dataset can be joined is to verify if they were both obtained from the same original table (one of the 32 mentioned before), and if they have at least one column in common with the same name. Meeting these conditions ensures that the tables can be joined by that column. On the other hand, the original ground truth identifies what columns can be matched

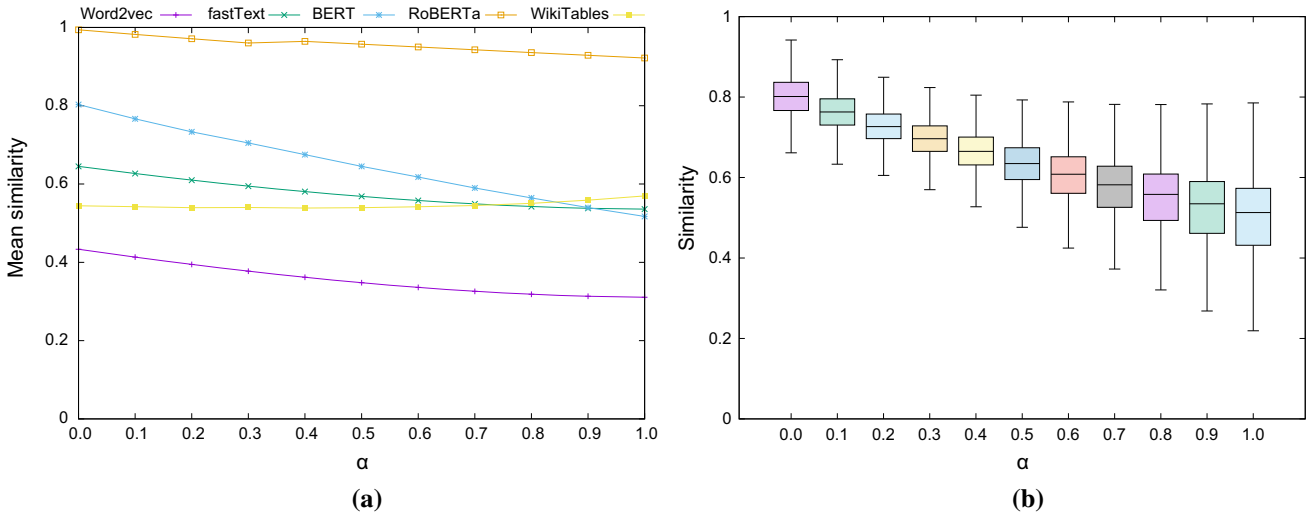

Fig. 3 Mean similarity between tables as a function of α for word embedding models (**a**); central tendency and dispersion of similarity values obtained by BERT (**b**)

between tables. Thus, two tables cannot be joined if they do not have any columns in common based on this ground truth. For the pairs of tables that do not fulfil any of these conditions, it cannot be guaranteed whether they can be joined or not, so they were discarded in the evaluation. According to this criteria, out of 499,500 pairs of tables, 15,137 can be joined, 475,651 cannot, and 8,712 are uncertain (and are therefore discarded in the evaluation).

Since column headings are chosen as a basis to determine whether two columns can be joined, to conduct an unbiased assessment models were evaluated using $\alpha = 0.0$, avoiding the use of headings as an evidence to perform the join operation.

Table 2 shows the performance of the baseline and word embedding models at the same cut-off thresholds defined in union retrieval experiments (P@1, P@10 and P@50).

Results follow a similar pattern to those obtained in the experiments for union retrieval. BM25 had again a low performance at P@1 (0.2959). BERT (0.8258) and WikiTables (0.8250) had a similar performance and the difference is not statistically significant. BERT improves 8% fastText, 15% Word2vec, and 180% BM25. All these differences are statistically significant.

P@10 shows WikiTables (0.7711) and BERT (0.7663) as the best performing models. The difference between them is not statistically significant. The former improves fastText by 9% and Word2vec by 22%. The performance of BM25 is remarkable (0.7623), very close to that of the contextual models, which was not the case in the union retrieval experiments. Since this baseline relies on lexical similarity, good results reflect that there are pairs of tables whose cell values overlap in columns that were considered as candidates for join. This is not an unexpected result, as the criteria for determining tables that could be joined did not impose that candidate columns had to be primary keys. However, tables in the dataset that meet this condition seems limited, as the performance of BM25 drops significantly at P@50, indicating that lexical similarity decreases (less content in common) as the number of tables returned increases.

Table 2 Mean (M) and standard deviation (SD) of P@1, P@10, and P@50 for BM25 and word embedding models ($\alpha = 0.0$) in join retrieval

Model	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

The best result for each cut-off threshold is bold-faced

*Indicates statistically significant improvement ($\rho < 0.01$) of the contextual model with respect to the best non-contextual model for the corresponding metric

WikiTables (0.5893) and BERT (0.5699) obtained the best results at P@50 (no statistically significant difference between them). WikiTables improves 9% the best non-contextual model and 21% the baseline. The difference was statistically significant in both cases.

These results show that, as in the union retrieval task, for the join retrieval task contextual models significantly outperform non-contextual models and the baseline established.

5.2 Union operation

This experiment evaluates the performance of the models in identifying columns from two tables that can be combined in union operations. A similarity threshold is set to decide if the operation can be carried out.

The ground truth provided in [10] identifies, for each pair of tables, to which columns the union operation can be applied. Of the pairs of tables on which union retrieval was evaluated, only those pairs with at least one matching column in the ground truth were considered. This resulted in 303,548 comparisons between columns (test samples), of which 132,622 (43.69%) can be combined through union operation and 170,926 (56.31%) cannot.

The performance of the models has been evaluated in terms of precision, recall and F1-score (harmonic mean of precision and recall) for different similarity thresholds, ranging from 0.1 to 1.0 in 0.1 steps. Each model was evaluated by selecting the best performing α value for P@10 in the union retrieval experiment (see Table 1).

Table 3 summarises the results for the best performing thresholds in terms of F1-score. The best precision was achieved by fastText (0.6386), followed by WikiTables (the difference was not statistically significant). The result can be explained by the higher similarity that contextual models assign on average to the columns compared, as shown in Fig. 3a. Therefore, many of them lie above the threshold and are considered as candidates for union, producing more false positives than context-free models and a subsequent loss of precision.

RoBERTa was the best model by far in terms of recall (0.9885). This model assigns on average the highest similarity to every pair of columns, and thus most of them were above the threshold. This also results in low precision (0.4346) and the worst F1-score of all the models evaluated. WikiTables outperformed the best non-contextual model (Word2vec) and the difference was statistically significant.

Regarding F1-score, the best result was obtained by WikiTables (0.7308), followed by fastText (no statistically significant). The former can be considered as the most balanced model for the union operation. At the other extreme, BERT and RoBERTa obtained lower performance than non-contextual models.

5.3 Join operation

In this experiment, the goal is to evaluate the ability of the models to identify pairs of columns from different tables that can be joined. Again, a threshold was established to take this decision.

The evaluation is focused on tables that can be joined given the criteria described in Sect. 5.1.2. Recall that there were 15,137 pairs of tables that could be joined. This subset results in 214,997 column pairs, where 103,414 (48,1%) can be joined following the criteria (they have the same column name) and 111,583 (51,9%) cannot. The dataset is well balanced in terms of the number of positive (can be joined) and negative samples (cannot).

As with join retrieval experiment, models were tested using $\alpha = 0.0$ to conduct an unbiased evaluation. Their performance was measured in terms of precision, recall and F1-score for different thresholds, ranging from 0.4 to 1.0 in 0.05 steps. The lower limit (0.4) was set by taking into account the minimum average similarity obtained by any of the models (0.4335 by Word2vec).

Table 3 Mean (M) and standard deviation (SD) of Precision, Recall and F1-score for union operation taking the best α and threshold values

Model	α	Threshold	Precision		Recall		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

The best result for each metric is bold-faced

*Indicates statistically significant improvement ($\rho < 0.01$) of the contextual model with respect to the best non-contextual model for the corresponding metric

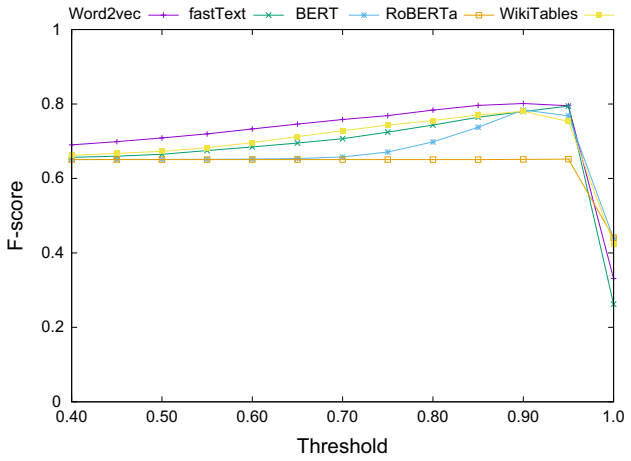

Fig. 4 F1-score for join operation and $\alpha = 0.0$ for five models for word embeddings

Figure 4 shows the F1-score for the five word embedding models analysed. The highest performance was achieved by Word2vec (threshold = 0.9), 0.8013 F1-score (0.7159 precision, 0.9098 recall), followed by fastText (threshold = 0.95), 0.7943 F1-score (0.7166 precision, 0.8910 recall). BERT obtained the highest score of contextual models (threshold = 0.9), 0.7842 F1-score (0.6795 precision, 0.9273 recall). Nevertheless, the difference between these three models was not statistically significant.

Given a model, the F1-score obtained is almost the same for all the threshold that fall below the average similarity presented in Fig. 3a. In those circumstances most column pairs will be candidate to join, obtaining a recall close to 1 but a low precision. The F1-score improves as the threshold increases (less false positives), dropping significantly at 1.0, where only columns with 100% similarity are joined.

The worst results were obtained by RoBERTa, with an almost constant F1-score around 0.65 that drops sharply to 0.44 at 1.0 threshold. Again, the high similarity assigned by RoBERTa on average provides high recall but low precision, which affects the final F1-score.

6 Discussion

This work has presented the first attempt to use contextual word embeddings in all the phases of the table retrieval and integration process. The evaluation carried out has analysed the performance of different word embedding models and contextual information in four different tasks.

In union retrieval, contextual models outperformed context-free models and BM25. Contextual models proved to be extremely precise in retrieving a relevant table in the first position of the ranking (P@1). Interestingly, BM25 performed better than Word2vec and fastText for P@50, and close to contextual models when only the column headings were considered. The conclusion that can be drawn from this result is that, as we move away from the top ranked tables, word embedding models tend to attribute high similarity scores to tables that are not really that similar, even though some of their columns may be close in the semantic vector space. Meanwhile, BM25 is more strict in calculating the similarity (considers only the lexical similarity of column headings), improving accuracy at the expense of recall.

This experiment revealed that precision of word embedding models dropped when only cell values or only column headings were considered. This points out that both information is complementary and should be taken into account, even to a small degree, since the differences are also significant at $\alpha = 0.1$ and $\alpha = 0.9$.

BERT and WikiTables obtained similar results in union retrieval, with no significant difference between them. Both performed significantly better than non-contextual models, BM25 and RoBERTa. WikiTables was the most stable in terms of performance as a function of α. This makes it more suitable for the task as it is more robust to variations in model parameters.

In join retrieval, BERT and WikiTables again performed best in the task, with no significant difference between them, outperforming non-contextual models and the baseline. This time, RoBERTa obtained similar results to the other contextual models. Overall, the performance was lower than in union retrieval, but are still competitive for their use in an integration pipeline. This drop in performance can be partially explained by the way in which the dataset was obtained, forcing to test the models using only cell values to make a fair assessment. As stated above, in union retrieval it was found that the contribution of both cell values and column headings was critical to the final performance. For this reason, it is expected that evaluating with a dataset that does not imply this limitation could lead to improvements in the final performance for all models.

The best results in terms of F1-score for union operation were obtained by WikiTables and fastTest (no statistically significant difference between them), surpassing all the other models. The former obtained significant better recall but less precision, although this difference was not significant. Again, WikiTables could be considered the most balanced option for this task.

Join operation results reveal that contextual models do not offer an advantage over non-contextual models in this task. As mentioned before, this result may be partly conditioned by the setup of the experiment, as only information from cell values was taken into account. In the case of BERT (see Fig. 3b) this led to high similarity on average and low dispersion, which makes it difficult to identify pairs of columns that stand out from the rest, which is the

ultimate goal of this task. RoBERTa was the most extreme case, as it assigned the highest similarity values on average (see Fig. 3a). In union operation, the recall of RoBERTa for the best threshold setting (0.9) was 0.9885, at the expense of a low precision (0.4346). Similar results were obtained for join: 0.9977 recall and 0.4804 precision for 0.95 threshold. This means that almost every pair of columns was assigned a similarity score over these thresholds, and was thus considered as good candidate for union/join, obtaining high recall but low precision.

7 Conclusions and future work

Data integration has been a relevant topic in information sciences for many years. The current advances in machine learning, and more specifically in the area of deep learning, has led to new ways to address this task.

This paper has presented a novel approach to the application of contextual word embeddings to data integration. These language models have achieved state-of-the-art results in many NLP tasks where the focus is unstructured data (i.e. raw text). However, in recent times these models have been used also to better understand structured data, and more specifically tabular information.

The novelty of this work has been the proposal of a solution to both union and join operations based on contextual word embeddings. The process consists of two phases: first, the most relevant tables for the corresponding operation are retrieved; secondly, candidate columns for union or join are identified.

Four different tasks have been proposed and analysed: union retrieval, join retrieval, union operation, and join operation. Each task has been evaluated using three contextual models, two non-contextual models, and a classic ranking function baseline for the two retrieval tasks. In these four tasks, the use of column headings and cell values have been studied for their impact in performance. Results revealed that all models benefited from the combination of both types of information, although models performed better when more weight was given to cell values.

Regarding the performance of the contextual models, BERT and its fine-tuned version WikiTables showed more stability and better performance than RoBERTa throughout all the experiments. In both union and join table retrieval, contextual models significantly outperformed non-contextual models. BERT was the best model for union retrieval (P@10 = 0.9737), and WikiTables for join retrieval (P@10 = 0.7711), although the difference was not statistically significant in the latter.

WikiTables obtained the best results in the union operation experiment, achieving a significant improvement with respect to other contextual models. In this task, there were mixed results, with fastText and Word2vec surpassing BERT and RoBERTa in terms of F1-score. Join operation results revealed no significant difference between static models and BERT. The criterion of selecting only the column with the highest similarity in each table to calculate candidate columns was not effective. Performance is strongly affected when models assign a high similarity value on average with a low dispersion, and this issue was more prevalent in contextual models, as shown in Fig. 3a. Different criteria should be explored in future work to improve performance in join operation.

This paper mainly focused in establishing the approach to data integration and analysing performance of contextual models in different dimensions (information used, performance of models, and role of context). As a future work, it is also planned to use vectors generated by the models to train a machine learning algorithm, and then include it in a final pipeline that automatically infers the best parameters (α and thresholds) for the data integration task at hand.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This research has been partially funded by project "Desarrollo de un ecosistema de datos abiertos para transformar el sector turístico" (GVA-COVID19/2021/103) funded by Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital de la Generalitat Valenciana (Spain); and by projects "CHAN-TWIN" (TED2021-130890B-C21), "COnscious natuRal TEXt generation (CORTEX)" (PID2021-123956OB-I00) and "Technological Resources for Intelligent VIral AnaLysis through NLP (TRIVIAL)" (PID2021-122263OB-C22), funded by MCIN/AEI/ 10.13039/501100011033 and by the European Union NextGenerationEU/PRTR.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abadi D, Ailamaki A, Andersen D, Bailis P, Balazinska M, Bernstein P, Boncz P, Chaudhuri S, Cheung A, Doan A et al

(2020) The seattle report on database research. ACM SIGMOD Rec 48(4):44–53. https://doi.org/10.1145/3385658.3385668
2. Miller RJ (2018) Open data integration. Proc Very Large Data Base Endow 11(12):2130–2139. https://doi.org/10.14778/3229863.3240491
3. Peters ME, Neumann M, Zettlemoyer L, Yih W-t (2018) Dissecting contextual word embeddings: Architecture and representation. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp. 1499–1509. Association for computational linguistics, Brussels, Belgium. https://doi.org/10.18653/v1/D18-1179
4. Zhang S, Balog K (2018) Ad hoc table retrieval using semantic similarity. In: Proceedings of the 2018 world wide web conference, Lyon, France, pp. 1553–1562. https://doi.org/10.1145/3178876.3186067
5. Zhang S, Balog K (2020) Web table extraction, retrieval, and augmentation: a survey. ACM Transact Intelli Syst Technol 11(2):1–35. https://doi.org/10.1145/3372117
6. Trabelsi M, Davison BD, Heflin J (2019) Improved table retrieval using multiple context embeddings for attributes. In: 2019 IEEE international conference on big data (Big Data), pp. 1238–1244. IEEE, Los Angeles, CA, USA. https://doi.org/10.1109/BigData47090.2019.9005681
7. Ahmadov A, Thiele M, Eberius J, Lehner W, Wrembel R (2015) Towards a hybrid imputation approach using web tables. In: Proceedings of the IEEE 2nd international symposium on big data computing, pp. 21–30. https://doi.org/10.1109/BDC.2015.38
8. Nguyen TT, Nguyen QVH, Matthias W, Karl A (2015) Result selection and summarization for web table search. In: Proceedings of the 31st international conference on data engineering (ISDE'15), pp. 231–242. IEEE, Seoul, South Korea. https://doi.org/10.1109/ICDE.2015.7113287
9. Shraga R, Roitman H, Feigenblat G, 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, pp. 1399–1408. Association for computing machinery, Virtual. https://doi.org/10.1145/3397271.3401120
10. Nargesian F, Zhu E, Pu KQ, Miller RJ (2018) Table union search on open data. Proc Very Large Data Base Endow 11(7):813–825. https://doi.org/10.14778/3192965.3192973
11. Liu Q, Kusner MJ, Blunsom P (2020) A survey on contextual embeddings. CoRR arXiv:abs/2003.07278
12. Chen Z, Trabelsi M, Heflin J, Xu Y, Davison BD (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, pp. 589–598. Association for computing machinery, Virtual. https://doi.org/10.1145/3397271.3401044
13. Devlin J, Chang M-W, Lee K, Toutanova K (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics, pp. 4171–4186. Association for computational linguistics, Minneapolis, MN, USA. https://doi.org/10.18653/v1/N19-1423
14. Herzig J, Nowak PK, Müller T, Piccinno F, Eisenschlos J (2020) TaPas: Weakly supervised table parsing via pre-training. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp. 4320–4333. Association for computational linguistics, Virtual. https://doi.org/10.18653/v1/2020.acl-main.398
15. Yin P, Neubig G, Yih W-t, Riedel S (2020) TaBERT: Pretraining for joint understanding of textual and tabular data. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp. 8413–8426. Association for computational linguistics, Virtual. https://doi.org/10.18653/v1/2020.acl-main.745
16. Agarwal V, Bhardwaj A, Rosso P, Cudré-Mauroux P (2021) Convtab: A context-preserving, convolutional model for ad-hoc table retrieval. In: 2021 IEEE International conference on big data (Big Data), pp. 5043–5052. https://doi.org/10.1109/BigData52589.2021.9671828
17. Du L, Gao F, Chen X, Jia R, Wang J, Zhang J, Han S, Zhang D (2021) Tabularnet: a neural network architecture for understanding semantic structures of tabular data. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery and data mining. KDD '21, pp. 322–331. Association for computing machinery, New York, NY, USA. https://doi.org/10.1145/3447548.3467228
18. Cappuzzo R, Papotti P, Thirumuruganathan S (2021) Embdi: generating embeddings for relational data integration. In: Greco S, Lenzerini M, Masciari E, Tagarelli A (eds.) Proceedings of the 29th Italian symposium on advanced database systems, SEBD 2021, Pizzo Calabro, Italy, September 5-9, 2021. CEUR Workshop Proceedings, vol. 2994, pp. 331–338. http://ceur-ws.org/Vol-2994/paper36.pdf
19. Huang X, Khetan A, Cvitkovic MW, Karnin ZS (2021) Tabtransformer: tabular data modeling using contextual embeddings. In: Workshop on weakly supervised learning (WeaSuL 2021). ICLR 2021 Workshop., Virtual. https://weasul.github.io/papers/7.pdf
20. Wang C, Chakrabarti K, He Y, Ganjam K, Chen Z, Bernstein PA (2015) Concept expansion using web tables. In: Proceedings of the 24th international conference on World Wide Web, Florence, Italy, pp. 1198–1208. https://doi.org/10.1145/2736277.2741644
21. Deng L, Zhang S, 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, pp. 1029–1032. Association for computing machinery, Paris, France. https://doi.org/10.1145/3331184.3331333
22. Cafarella MJ, Halevy A, Wang DZ, Wu E, Zhang Y (2008) Webtables: Exploring the power of tables on the web. In: Proceedings of the very large data base endowment, pp. 538–549. https://doi.org/10.14778/1453856.1453916
23. Zhang S, Balog K (2017) Entitables: smart assistance for entity-focused tables. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, pp. 255–264. Association for computing machinery, Shinjuku, Tokyo, Japan. https://doi.org/10.1145/3077136.3080796
24. Bhagavatula CS, Noraset T, Downey D (2013) Methods for exploring and mining tables on wikipedia. In: Proceedings of the ACM SIGKDD workshop on interactive data exploration and analytics. IDEA '13, pp. 18–26. Association for computing machinery, Chicago, Illinois. https://doi.org/10.1145/2501511.2501516
25. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) 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, pp. 3111–3119. Curran Associates Inc., Lake Tahoe, Nevada
26. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Empirical methods in natural language processing (EMNLP), Doha, Qatar, pp. 1532–1543. https://doi.org/10.3115/v1/D14-1162. http://www.aclweb.org/anthology/D14-1162
27. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human

language technologies, Volume 1 (Long Papers), pp. 2227–2237. Association for computational linguistics, New Orleans, Louisiana. https://doi.org/10.18653/v1/N18-1202
28. Howard J, Ruder S (2018) Universal language model fine-tuning for text classification. In: Proceedings of the 56th annual meeting of the association for computational linguistics, pp. 328–339. Association for computational linguistics, Melbourne, Australia. https://doi.org/10.18653/v1/P18-1031
29. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: A robustly optimized bert pretraining approach. CoRR arXiv:abs/1907.11692
30. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, vol. 30, pp. 5998–6008. Curran Associates, Inc., Long Beach, CA, USA
31. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K, Klingner J, Shah A, Johnson M, Liu X, Kaiser L, Gouws S, Kato Y, Kudo T, Kazawa H, Stevens K, Kurian G, Patil N, Wang W, Young C, Smith J, Riesa J, Rudnick A, Vinyals O, Corrado G, Hughes M, Dean J (2016) Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv CoRR arXiv:abs/1609.08144
32. Sennrich R, Haddow B, Birch A (2016) Neural machine translation of rare words with subword units. In: Proceedings of the 54th annual meeting of the association for computational linguistics, pp. 1715–1725. Association for computational linguistics, Berlin, Germany. https://doi.org/10.18653/v1/P16-1162
33. Gupta S, Kanchinadam T, Conathan D, Fung G (2020) Task-optimized word embeddings for text classification representations. Front Appl Math Stat 5:1–10. https://doi.org/10.3389/fams.2019.00067
34. Mu J, 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
35. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. Soviet Phys Dokl 10:707–710
36. Sarma AD, Fang L, Gupta N, Halevy A, Lee H, Wu F, Xin R, Yu C (2012) Finding related tables. In: Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD'12), pp. 817–828. Association for computing machinery, Scottsdale, Arizona, USA. https://doi.org/10.1145/2213836.2213962
37. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Transact Assoc Comput Linguist 5:135–146. https://doi.org/10.1162/tacl_a_00051
38. Bhagavatula CS, Noraset T, Downey D (2015) Tabel: entity linking in web tables. In: The semantic web - ISWC 2015, pp. 425–441. Springer, Cham. https://doi.org/10.1007/978-3-319-25007-6_25
39. Kobayashi M, Takeda K (2000) Information retrieval on the web. ACM Comput Surv 32(2):144–173. https://doi.org/10.1145/358923.358934
40. Robertson SE, Walker S, Jones S, Hancock-Beaulieu MM, Gatford M (1994) Okapi at trec-3. In: Proceedings of the third TREC conference, vol. 500-225, pp. 109–126. National Institute of Standards & Technology, Gaithersburg, Maryland, USA