Table2Vec: Neural Word and Entity Embeddings for Table Population and Retrieval

Li Deng
University of Stavanger
ninalx1991@gmail.com

Shuo Zhang
University of Stavanger
shuo.zhang@uis.no

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

ABSTRACT

Tables contain valuable knowledge in a structured form. We employ neural language modeling approaches to embed tabular data into vector spaces. Specifically, we consider different table elements, such caption, column headings, and cells, for training word and entity embeddings. These embeddings are then utilized in three particular table-related tasks, row population, column population, and table retrieval, by incorporating them into existing retrieval models as additional semantic similarity signals. Evaluation results show that table embeddings can significantly improve upon the performance of state-of-the-art baselines.

CCS CONCEPTS

• **Information systems → Environment-specific retrieval**; *Retrieval models and ranking*; Search in structured data;

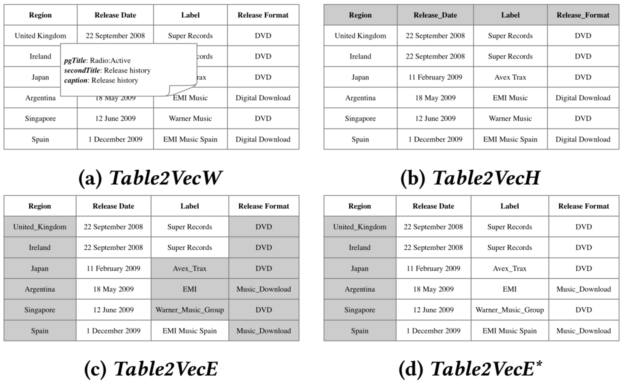
KEYWORDS

Neural embeddings; table retrieval; table population; table2vec

ACM Reference Format:
Li Deng, Shuo Zhang, and Krisztian Balog. 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 (SIGIR ’19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331333

1 INTRODUCTION

Tables contain a vast amount of useful information in the form of structured data. Recently, a growing body of work has developed around leveraging tabular data in various applications [1, 3, 4, 9, 10, 12–16]. In this paper, we focus on three particular table-related tasks: row population, column population, and table retrieval. All three tasks are performed on *relational tables*, which describe a set of entities placed in a *core column*, along with their attributes in additional columns. *Table population* is the task of populating a given seed table with additional elements [9, 14]. Specifically, we address the *row population* and *column population* tasks proposed in [14]. The former aims to complement the core column of a relational table with additional entities, while the latter aims to complement the header row with additional headings. *Table retrieval* is the task of returning a ranked list of tables for a keyword query [15].



Region	Release Date	Label	Release Format
United Kingdom	22 September 2008	Super Records	DVD
Ireland		cords	DVD
Japan		rax	DVD
Argentina	18 May 2009	EMI Music	Digital Download
Singapore	12 June 2009	Warner Music	DVD
Spain	1 December 2009	EMI Music Spain	Digital Download

pgTitle: Radio:Active
secondTitle: Release history
caption: Release history

(a) ***Table2VecW***

Region	Release_Date	Label	Release_Format
United Kingdom	22 September 2008	Super Records	DVD
Ireland	22 September 2008	Super Records	DVD
Japan	11 February 2009	Avex Trax	DVD
Argentina	18 May 2009	EMI Music	Digital Download
Singapore	12 June 2009	Warner Music	DVD
Spain	1 December 2009	EMI Music Spain	Digital Download

(b) ***Table2VecH***

Region	Release Date	Label	Release Format
United_Kingdom	22 September 2008	Super Records	DVD
Ireland	22 September 2008	Super Records	DVD
Japan	11 February 2009	Avex_Trax	DVD
Argentina	18 May 2009	EMI	Music_Download
Singapore	12 June 2009	Warner_Music_Group	DVD
Spain	1 December 2009	EMI Music Spain	Music_Download

(c) ***Table2VecE***

Region	Release Date	Label	Release Format
United_Kingdom	22 September 2008	Super Records	DVD
Ireland	22 September 2008	Super Records	DVD
Japan	11 February 2009	Avex_Trax	DVD
Argentina	18 May 2009	EMI	Music_Download
Singapore	12 June 2009	Warner_Music_Group	DVD
Spain	1 December 2009	EMI Music Spain	Music_Download

(d) ***Table2VecE****

Figure 1: Illustration of different Table2Vec embeddings.

Prior table-related work has considered embeddings, both pre-trained ones and task-specific ones. For example, Zhang and Balog [15] use pre-trained word and entity embeddings for table retrieval. Ghasemi-Gol and Szekely [6] develop table embeddings for table classification and Gentile et al. [5] train table embeddings for web table entity matching. However, to the best of our knowledge, no studies have been conducted on training table embeddings specifically for table population and retrieval tasks. To fill the gap, we propose Table2Vec, a neural language modeling approach to map different table elements into semantic vector spaces, with specific table-oriented tasks in mind.

In this study, we train four variants of table embeddings by utilizing different table elements. Specifically, word embeddings (Table2VecW) consider all the terms within a table, and are leveraged for table retrieval. The method employing Table2VecW outperforms a start-of-the-art baseline by over 10% in terms of NDCG@10. Interestingly, this is on par with using pre-trained Word2Vec embeddings using Google News data. Two different entity embeddings are obtained by considering only core column entities (Table2VecE*) and all table entities (Table2VecE). Table2VecE* is employed for the row population task. We show that it significantly outperforms all baselines. Combining with an effective baseline can lead to further improvements. Table2VecE is employed in table retrieval and can yield minor improvements, albeit those are not statistically significant. Heading embeddings (Table2VecH) are generated for the column population task by utilizing table headings. Table2VecH results in substantial and significant improvements over the baseline. Especially, when the number of seed headings becomes larger, it achieves 40% relative improvement in NDCG@10 over the baseline.

The resources developed in this work are made publicly available at https://github.com/iai-group/sigir2019-table2vec.

SIGIR ’19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331333

2 TRAINING TABLE2VEC EMBEDDINGS

In this section, we first introduce the neural model for training embeddings (Sect. 2.1), and then detail four variants of table embeddings (Sect. 2.2).

2.1 Neural Model for Training Embeddings

We base the training of our embeddings on the skip-gram neural network model of *Word2Vec* [7]. It is a computationally efficient two-layer neural language model that learns the meaning of terms from raw sequences and maps those terms to a vector space, such that similar terms close to each other.

More formally, given a sequence of training terms $t_1, t_2, \ldots, t_n$, the objective is to maximize the average log probability:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{-c\leq j\leq c, j\neq 0}\log p(t_{i+j}|t_i)\,, \tag{1}$$

where c is the size of training context, and the probability $p(t_{i+j}|t_i)$ is calculated using the following softmax function:

$$p(t_o|t_i) = \frac{\exp(\vec{v}'_{t_o}{}^{\top}\vec{v}_{t_i})}{\sum_{t=1}^{V}\exp(\vec{v}'_t{}^{\top}\vec{v}_{t_i})}\,, \tag{2}$$

where V is the size of vocabulary, and $\vec{v}_{t_i}$ and $\vec{v}'_{t_o}$ are the input and output vector representations of term t, respectively. Semantically similar terms share more similar vector representations; accordingly, the dot product between those vectors results in higher values, which means higher probabilities after softmax.

In our scenario, we consider terms to be words, entities, or headings in a table. We also employ negative sampling, to make the training of our models computationally more efficient.

2.2 Four Variants

We train four different table embeddings, using different table elements as input; these are summarized in Table 1 and illustrated in Fig. 1. All embeddings are trained using the same neural model, but they differ in (i) what constitutes as a term and (ii) which table elements are used for training.

Table2VecW This method takes all the words appearing in a table into consideration. Specifically, it considers the page title, section title, table caption, table headings, and all table cells; see Fig. 1a.

Table2VecH Instead of using single words, we further leverage the table structure and represent tables as sequences of headings. Each heading is treated as a single term, as is shown in the shadowed area in Fig. 1b.

Table2VecE Tables often contain entities, which are semantically more meaningful than words. Thus, we take sequences of entities as input, by extracting all entities that appear within table cells; see the shadowed area in Fig. 1c.

Table2VecE* Relational tables describe a set of entities as well as their attributes in the columns. These entities are placed in the *core column*. Table2VecE* considers only entities in the core column of the table, as is shown in Fig. 1d.

3 UTILIZING TABLE2VEC EMBEDDINGS

In this section, we extend previous table population and retrieval methods by incorporating the Table2Vec embeddings that we introduced in Sect. 2. For all tasks, we keep our focus on relational tables. It is assumed that entities mentioned in the table are recognized and disambiguated by linking them to entries in a knowledge base [2]. The table population task is considered in two flavors: row population and column population. We shall refer to the input table T as *seed table*, in which the set of entities from the core column are referred as seed entities E, and the set of headings are denoted as seed headings L.

Table 1: Table2Vec embeddings.

Method	Input	Semantic repr.
Table2VecW	all table data	word embeddings
Table2VecH	table headings	heading embeddings
Table2VecE	all entities	entity embeddings
Table2VecE*	core column entities	entity embeddings

3.1 Row Population

Row population is a task of returning a list of entities, based on their likelihood of being added to the core column of the seed table T in the next row. The ranking is established based on the similarity of a candidate entity e to the seed table entities E. In this task, we measure entity similarity by two approaches: using a knowledge base and using Table2Vec embeddings.

3.1.1 Baselines. We employ three probabilistic ranking methods from [14] as our baselines, which rank candidate entities according to $P(e|E)$. Candidate entity selection is done as in [14].

BL1 Entity similarity is measured based on the similarity of relations of e, obtained from RDF triples, and those of the seed tables entities E.

BL2 It uses the Wikipedia Link-based Measure [8] to estimate the semantic relatedness of entities based on their outgoing links (in the knowledge base).

BL3 It relies on the Jaccard similarity between outgoing links of entities.

3.1.2 Using Table2Vec embeddings. Recall that we have two entity embeddings, Table2VecE and Table2VecE*. The former is trained on all entities contained in the table, while the latter considers only entities in the core column. Given that the row population task focuses on the core column, we employ the Table2VecE* embeddings here. We measure the similarity of each candidate entity e, against the seed entities $e' \in E$, using the cosine similarity of their respective embedding vectors:

$$sim(e,E) = \frac{1}{|E|}\sum_{e'\in E} sim(e,e') = \frac{1}{|E|}\sum_{e'\in E}\frac{\vec{v}_e\cdot\vec{v}_{e'}}{\|\vec{v}_e\|\|\vec{v}_{e'}\|}\,, \tag{3}$$

where $|E|$ is the size of seed entity set, and $\vec{v}_e$ and $\vec{v}_{e'}$ are the embedding vectors of the candidate and seed entities, respectively.

We then combine the baseline similarity with the Table2Vec-based similarity using the following linear mixture:

$$P(e|E) = \alpha\, P_{KB}(e|E) + (1-\alpha)\, P_{emb}(e|E)\,, \tag{4}$$

where P_{KB} is the similarity measured using the knowledge base and P_{emb} is based on table embeddings, and equals to Eq. (3).

3.2 Column Population

Column population is the task of returning a ranked list of headings, $l_1, \ldots, l_k$, given a seed table T. The returned headings are ranked based on their relevance to the seed headings L. Similarly to row population, we consider two heading similarity measures.

3.2.1 Baseline. The baseline method, using a table corpus, is taken from [14]. First, relevant tables are retrieved from the table corpus. Then, the probability of a candidate heading being relevant $P(l|L)$ is estimated based on the occurrences of that heading in relevant tables.

3.2.2 Using Table2Vec embeddings. We use embeddings trained on table headings, Table2VecH, for heading relevance estimation. Similarly to row population, we measure the cosine similarity between the embedding vectors of the candidate heading l and seed headings $l' \in L$. Then, the baseline estimate is combined with the embedding-based similarity using:

$$P(l|L) = \alpha \, P_{KB}(l|L) + (1 - \alpha) \, P_{emb}(l|L) \, . \tag{5}$$

3.3 Table Retrieval

Table retrieval is the task of returning a ranked list of tables in response to a keyword query q, based on their relevance to q. For this task, we employ a feature-based method as a baseline, which is referred to as the LTR method in [15]. We utilize the word-based and entity-based table embeddings, Table2VecW and Table2VecE, to compute additional semantic matching features. Specifically, each type of embedding contributes four features, for each of the similarity methods in [15].

Given that both the table and query are vectors now, we compute cosine similarity to measure relevance. For comparison purposes, we employ both methods in [15]: *early fusion* and *late fusion*. For the former method, query-table relevance is measured between the centroid of query term vectors and the centroid of table term vectors. The latter method computes pairwise cosine similarity between table terms ($\vec{t}_j$) and query terms ($\vec{q}_i$) first, and then aggregates those results. Here, query-table relevance is measured using an aggregator function, which can be: (i) maximum of $cosine(\vec{q}_i, \vec{t}_j)$, (ii) sum of $cosine(\vec{q}_i, \vec{t}_j)$ (iii) average of $cosine(\vec{q}_i, \vec{t}_j)$. In this paper, we combine all four measures (i.e., early fusion and late fusion using max, sum, and avg aggregators) to yield the final similarity score. For performance comparison, we employ pre-trained Graph2Vec [11] and Word2Vec embeddings [7].

4 EVALUATION

In this section, we formulate our research questions (Sect. 4.1), discuss our experimental setup (Sect. 4.2), and then present our results and analysis for the three tasks (Sects. 4.3–4.5).

4.1 Research Questions

We address the following research questions:

RQ1 Can Table2Vec improve table population performance against state-of-the-art baselines?

RQ2 Does the training of word embeddings specifically on tables, as opposed to news, affect retrieval performance?

RQ3 Which of the semantic representations (entity vs. word embeddings) performs better in table retrieval?

Table 2: Statistics for Table2Vec embeddings. Neg is short for negative sampling (measured in number of words).

Embedding	Total terms	Unique terms	Neg	Win_size
Table2VecW	200,157,990	1,829,874	25	5
Table2VecH	7,962,443	339,433	25	20
Table2VecE	24,863,683	2,159,467	25	50
Table2VecE*	5,367,837	1,285,708	25	50

4.2 Experimental Setup

For table population, we use Mean Average Precision (MAP) as the main metric and Mean Reciprocal Rank (MRR) as a supplementary metric for performance evaluation. Table retrieval performance is evaluated by Normalized Discounted Cumulative Gain (NDCG) with a cut-off at 10 and 20. To test significance, we use a two-tailed paired t-test and write ∘ to denote not significant, and †/‡ to denote significance at the 0.05 and 0.01 levels, respectively.

We use the Wikipedia Tables corpus [14], which contains 1.6 million high-quality relational tables, both for training the Table2Vec embeddings and for the retrieval experiments. For the word-based embedding, Table2VecW, we filter out empty strings, numbers, HTML tags, and stopwords from the raw text during training to obtain a better representation. For Table2VecH, we employ no normalization for the headings, i.e., "year(s)," "year:," and "year" will be treated as different headings in our experiment. Table 2 shows the statistics of different Table2Vec embeddings. DBpedia is used as our knowledge base, which is consistent with the original experiments in [14, 15]. The test inputs and ground truth assessments are obtained for the three tasks as follows:

- *Row population:* we use the test set from [14]. It contains 1000 relational tables, of which each table has at least six rows and four columns. For evaluation, we take entities from the first i rows ($i \in [1..5]$) as seed entities, and the remaining entities as ground truth. The test set contains 21,502 unique entities.
- *Column population:* we use the test set from [14], consisting of 1000 relational tables. Headings from the first j columns ($j \in [1..3]$) are taken as seed headings, while the rest constitute the ground truth. There are a total of 7,216 unique column headings.
- *Table retrieval:* we use a set of 60 queries (two query subsets, QuerySet 1 and QuerySet 2) and corresponding ground truth relevance labels from [15], a total of 3,120 query-table pairs.

4.3 Row Population

The row population results are listed in Table 3. The top three lines show the results of the baselines from the literature. The bottom three lines are the results of combining the baselines with Table2VecE*. Note that the combination involves a mixture parameter α (cf. Eq. (4)). To understand the potential of using table embeddings, we perform a grid search in steps of 0.1 for the value of α, and report results using the α value that yielded the best MAP score. The best performing α values for BL1, BL2, and BL3 are 0.4, 0.0, and 0.1, respectively. This means that the second baseline does not contribute at all to the combination.

Overall, we find that the combined methods outperform the respective baselines substantially and significantly ($p < 0.01$). BL1 + Table2VecE* yields the best performance in terms of MAP. It is

Table 3: Row population performance. Statistical significance is tested against the respective baseline.

Method	#Seed entities ($\|E\|$)									
	1		2		3		4		5	
	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR
BL1	0.4360	0.5552	0.4706	0.5846	0.4788	0.5856	0.4786	0.5779	0.4711	0.5618
BL2	0.2612	0.4779	0.2778	0.4887	0.2845	0.4811	0.2846	0.4808	0.2817	0.4689
BL3	0.2912	0.5024	0.3024	0.4927	0.3028	0.4815	0.2987	0.4780	0.2910	0.4609
BL1 + Table2VecE*	**0.5581**‡	0.7414‡	**0.6147**‡	0.8141‡	**0.6400**‡	0.8424‡	**0.6524**‡	0.8427‡	**0.6533**‡	0.8372‡
BL2 + Table2VecE*	0.5461‡	0.7710‡	0.6027‡	**0.8317**‡	0.6187‡	0.8440‡	0.6217‡	0.8389‡	0.6223‡	**0.8410**‡
BL3 + Table2VecE*	0.5487‡	**0.7728**‡	0.6049‡	0.8294‡	0.6218‡	**0.8482**‡	0.6249‡	**0.8435**‡	0.6251‡	0.8395‡

Table 4: Column population performance. Statistical significance is tested against the baseline. BL is short for baseline, and TH is short for Table2VecH.

Method	#Seed column labels ($\|L\|$)					
	1		2		3	
	MAP	MRR	MAP	MRR	MAP	MRR
BL	0.2507	0.3753	0.2845	0.4037	0.2852	0.3552
BL + TH	**0.2551**$^{\circ}$	**0.3796**$^{\circ}$	**0.3322**‡	**0.4400**$^{\circ}$	**0.4000**‡	**0.5080**‡

Table 5: Table retrieval performance. Statistical significance is tested against the baseline.

Method	NDCG@10	NDCG@20
Baseline	0.5456	0.6031
Baseline + Word2Vec	0.6006†	0.6588†
Baseline + Graph2Vec	0.5764°	0.6340°
Baseline + Table2VecW	0.6096‡	0.6505†
Baseline + Table2VecE	0.5569°	0.6161°

worth pointing out that the performance of this combined method improves more with more seed entities than the baseline BL1, which reaches its peak already after two seed entities. This indicates the seed entities are better utilized in our embedding-based method.

4.4 Column Population

Table 4 shows column population performance. We find that the combined method involving Table2VecH significantly outperforms the baseline method ($p < 0.01$) in terms of MAP when $|L| > 1$. For $|L| = 3$ it achieves substantial and significant improvements ($p < 0.01$) both in terms of MAP and MRR. Moreover, while the baseline performance does not improve with more seed headings, the combined method can effectively utilize larger input sizes and keeps improving the performance. Combining these findings with the results obtained in Sect. 4.3, we answer RQ1 positively. The interpolation parameter (cf. Eq. (5)) that yielded the best performance for the combined method is $\alpha = 0.01$, which indicates Table2VecH similarity is assigned much higher importance than the baseline.

4.5 Table Retrieval

To answer RQ2 and RQ3, we list the table retrieval results in Table 5. For Graph2Vec and Table2VecE, we achieve improvements over the baseline but these are not statistically significant. Table2VecW and Word2Vec have very comparable performance to each other and they outperform all other methods and significantly improve over the baseline method ($p < 0.01$). The lack of difference between the two indicates that it does not make a difference for the table retrieval task whether word embeddings are trained specifically on tables or not (RQ2). As for the different semantic representations (RQ3), these results show that word embeddings are more beneficial for table retrieval than entity embeddings.

5 CONCLUSION AND FUTURE WORK

In this paper, we have introduced Table2Vec, a neural language model for training word and entity embeddings on various table elements. These embeddings have been utilized in three particular table-related tasks, and have been shown to significantly improve retrieval effectiveness. We have derived these embeddings particularly from a Wikipedia tables corpus, which contains only high-quality relational tables. In the future, we wish to extend our research to other table corpora, as well as to other types of tables.

REFERENCES

[1] Julian Eberius Wolfgang Lehner Robert Wrembel Ahmad Ahmadov, Maik Thiele. 2015. Towards a Hybrid Imputation Approach Using Web Tables. In *Proc. of BDC '15*. 21–30.

[2] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: Entity Linking in Web Tables. In *Proc. of ISWC '15*. 425–441.

[3] Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data Integration for the Relational Web. *Proc. of VLDB Endow.* 2 (2009), 1090–1101.

[4] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding Related Tables. In *Proc. of SIGMOD '12*. 817–828.

[5] Anna Lisa Gentile, Petar Ristoski, Steffen Eckel, Dominique Ritze, and Heiko Paulheim. 2017. Entity Matching on Web Tables: a Table Embeddings approach for Blocking. In *Proc. of EDBT '17*. 510–513.

[6] Majid Ghasemi-Gol and Pedro A. Szekely. 2018. TabVec: Table Vectors for Classification of Web Tables. *CoRR* (2018).

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*. 3111–3119.

[8] David Milne and Ian H. Witten. 2008. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In *Proc. of AAAI '08*.

[9] Kaushik Chakrabarti Mohamed Yakout, Kris Ganjam and Surajit Chaudhuri. 2012. InfoGather: entity augmentation and attribute discovery by holistic matching with Web tables. In *Proc. of SIGMOD '12*. 97–108.

[10] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering Table Queries on the Web Using Column Keywords. *Proc. of VLDB Endow.* 5 (2012), 908–919.

[11] Petar Ristoski and Heiko Paulheim. 2016. RDF2vec: RDF Graph Embeddings for Data Mining. In *Proc. of ISWC '16*. 498–514.

[12] Denilson Barbosa Paolo Merialdo Yoones A. Sekhavat, Francesco di Paolo. 2014. Knowledge Base Augmentation using Tabular Data. In *Proc. of LDOW '14*.

[13] Shuo Zhang. 2018. SmartTable: Equipping Spreadsheets with Intelligent Assistance Functionalities. In *Proc. of SIGIR '18*. 1447–1447.

[14] Shuo Zhang and Krisztian Balog. 2017. EntiTables: Smart Assistance for Entity-Focused Tables. In *Proc. of SIGIR '17*. 255–264.

[15] Shuo Zhang and Krisztian Balog. 2018. Ad Hoc Table Retrieval using Semantic Similarity. In *Proc. of WWW '18*. 1553–1562.

[16] Shuo Zhang and Krisztian Balog. 2018. On-the-fly Table Generation. In *Proc. of SIGIR '18*. 595–604.