ConvTab: A Context-Preserving, Convolutional Model for Ad-Hoc Table Retrieval

Vibhav Agarwal
International Institute of
Information Technology Bangalore
Bangalore, India
vibhav.agarwal@iiitb.ac.in

Akansha Bhardwaj
eXascale Infolab
University of Fribourg
Fribourg, Switzerland
akansha.bhardwaj@unifr.ch

Paolo Rosso
eXascale Infolab
University of Fribourg
Fribourg, Switzerland
paolo.rosso@unifr.ch

Philippe Cudré-Mauroux
eXascale Infolab
University of Fribourg
Fribourg, Switzerland
pcm@unifr.ch

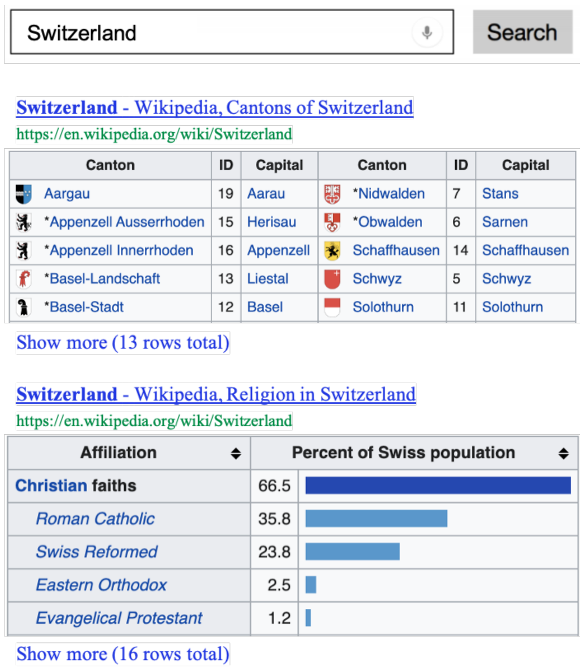
***Abstract*—Ad-hoc table retrieval, also known as table search, is the problem of finding tables relevant to a search query. This search query can be a keyword or a table itself, referred to as keyword-based and table-based search, respectively. With the vast amounts of tabular data available online, it has become essential for users to identify relevant tables that meet their search criteria. In this regard, there has been a wide variety of research on this problem using pure lexical features, semantic representation, embeddings, as well as intrinsic and extrinsic features of the tables. However, one of the significant limitations of most of the existing methods is that they do not keep the table's structure and the globalized context intact when building semantic representations of tabular data. Deriving motivation from this fact, we propose an effective approach based on Convolutional Neural Networks (CNNs) – ConvTab – to train the embeddings of tabular data. Our approach is divided into two phases. First, we leverage the discriminating power of CNNs to train a table classifier. Next, the representations learned from this model are used to generate semantic features for query-table similarity. These query-table similarity features are then used as input to the learning algorithm. We evaluate our approach on the table retrieval task using standard NDCG, MAP, and MRR metrics. Experiments reveal that ConvTab significantly outperforms the state of the art in ad-hoc table retrieval by 16.9% and 8.37% using NDCG at cutoffs 5 and 20, respectively. For reproducibility purposes, we share our model as well as all details of our implementation[1].**

***Index Terms*—Information systems, Learning to rank, Retrieval models, Ranking**

I. Introduction

Tables are a universal method for representing data; they can be used as a visual communication pattern, data arrangement, or organization tools. The Web has a collection of over 150 million tables [1], which, as a whole, represents an invaluable source of semi-structured knowledge. Data from these tables belong to very different categories, including finance, medicine, agriculture, or geography. Web tables have been extensively researched for various tasks like semantic understanding [2], search [1], [3], [4], table mining [5]–[7], table augmentation [8]–[12], etc.

The problem of ad-hoc table retrieval, or table search, is a fundamental discovery task in this context and the focus of our work. Given a query, the task is to retrieve relevant tables [13] and rank them in order of their relevance to the query. It is an essential task on its own and is a core component in many table mining and extraction tasks, like table integration, table completion, etc. Commercial products, such as Microsoft Power Query Table[2], also provide search functionalities and smart assistance features based on table search. Table search may be classified as either keyword-based search or table-based search depending on the type of the input query.



Switzerland - Wikipedia, Cantons of Switzerland
https://en.wikipedia.org/wiki/Switzerland

Canton	ID	Capital	Canton	ID	Capital
Aargau	19	Aarau	*Nidwalden	7	Stans
*Appenzell Ausserrhoden	15	Herisau	*Obwalden	6	Sarnen
*Appenzell Innerrhoden	16	Appenzell	Schaffhausen	14	Schaffhausen
*Basel-Landschaft	13	Liestal	Schwyz	5	Schwyz
*Basel-Stadt	12	Basel	Solothurn	11	Solothurn

Show more (13 rows total)

Switzerland - Wikipedia, Religion in Switzerland
https://en.wikipedia.org/wiki/Switzerland

Affiliation	Percent of Swiss population	
Christian faiths	66.5	
Roman Catholic	35.8	
Swiss Reformed	23.8	
Eastern Orthodox	2.5	
Evangelical Protestant	1.2	

Show more (16 rows total)

Fig. 1: An example of an ad-hoc table retrieval task for query 'Switzerland' that results in two relevant tables arranged according to their relevance score.

Figure 1 shows an example of a table retrieval task where the query 'Switzerland' results in two tables arranged in the order of their relevance score. The first result is about the geographic and administrative regions in Switzerland, while the second one is about its religious demography.

[1]https://github.com/vibhavagarwal5/table2vec-vibhav

[2]https://docs.microsoft.com/en-us/powerquery-m/table-functions

Previous works have considered tables as documents such that traditional document retrieval methods can be directly applied to table retrieval [1], [9], [10]. Supervised learning, based on hand-crafted features from tables, queries, and query-table pairs [13], has resulted in some of the best performing table retrieval systems. Building on this, Zhang and Balog [14] introduced semantic features to embed queries and tables into a semantic space, and then trained a supervised model using both semantic and traditional features. Ghasemi-Gol and Szekely [15] developed table embeddings for table classification, while Gentile et al. [16] trained table embeddings for web table entity matching. Shraga et. al [17] also proposed an approach for table retrieval that uses intrinsic (passage-based) and extrinsic (manifold-based) similarity that results in a better retrieval quality than semantically rich baselines. Hybrid-BERT-Row-Max [18] extends deep contextualised language models like BERT [19] to generate table representations. Additionally, recent approaches have focused on preserving schema information while learning table representations, which has proved to be beneficial [20], [21]. We discuss the related works in detail further in Section II.

Though a significant amount of work has been done in this context, we note that most of the previous table embeddings do not preserve the contextual information present as 2-D matrices in tabular data. Transforming tables into one dimensional textual data potentially results in a significant loss of structural information. We also note that all previous approaches need metadata information on the tables, which is not always available in real-world settings. To fill this gap, we propose a novel convolutional approach for tabular data where we train table embeddings by preserving their structural information. Our semantic features are completely based on the content and do not rely on the metadata of the table.

A. *Our approach and contribution*

Our proposed approach trains a model on tabular data for the table classification task by leveraging the discriminating power of CNNs. Architecturally, our approach draws motivation from the field of computer vision. Broadly, we approach the problem of ad-hoc table retrieval in two steps:

- First, we learn an embedding representation for the entities appearing in the table through an end-to-end learning method (E3LM) as described in Section III. The target task is table classification with 'valid' and 'invalid' categories. 'Valid' tables come from the dataset, while 'invalid' tables are those that are generated synthetically by randomly shuffling the cells. This is further described in Section III-A.
- Next, the features formed by the penultimate layer of the neural net architecture are used to retrieve and rank appropriate tables.

Results indicate that our approach considerably outperforms state-of-the-art (SoTA) approaches for the table retrieval task, by a margin of 16.9% and 8.37% in terms of NDCG scores (at cutoff 5, and 20 respectively) on a large, standard dataset. It is important to mention that our approach can be used for keyword-based as well as table-based search methods. However, as the standard dataset for table retrieval comprised keyword based queries only, we report our results with respect to those queries.

To summarize, our main contributions are as follows:

1) We propose a novel approach to solve the problem of ad-hoc table retrieval by training word embeddings using Convolutional Neural Networks (CNNs) while preserving the context and structure of the various table entries. Compared to all pre-existing methods, our approach does not require the metadata of the tables.
2) For reproducibility purposes, our complete code, run files, and trained models are freely accessible for further research.

The rest of this paper is organized as follows. We discuss related work in Section II. We introduce our approach for table retrieval in Section III, where we describe our training method followed by our method for predicting the rankings of a query-table pair. In Section IV, we introduce our dataset and experimental setup, followed by our results and evaluations structured in the form of research questions that we answer. Finally, we conclude the paper in Section V where we summarize our results, discuss the limitations of our approach as well as potential future avenues for our work.

II. RELATED WORK

A. *Table as a Document*

A number of approaches for ad-hoc table retrieval use standard document retrieval methods for table ranking [1], [9] by considering tables as documents. The most straightforward approach in this context is when a table is represented by a single field containing all the text. The retrieval score is calculated using existing retrieval methods, such as language models or BM25 [22].

A late fusion method [23] has also been proposed for multi-field ranking where, for a given query, a score is calculated independently for each field, and a linear combination of the scores is calculated [10]. The final score is given by:

$$score(Q,T) = \sum_i w_i \times score(Q, f_i) \tag{1}$$

where Q is a given query, T is a table, f_i is the i^{th} field of T, and w_i is the weight associated with f_i.

Some approaches that use supervised ranking methods by generating multiple query, table, and query-table features were proposed in the literature for the task of table retrieval [1], [8], [24]. Zhang and Balog [13] proposed to extend these features with semantic matching between queries and tables using various semantic features such as word embeddings, graph embeddings, bag-of-entities, and bag-of-categories. The DBpedia knowledge base was then used to construct a multi-hot vector representation for both bag of entities and bag of categories. This implies that the dimension of the features is equivalent to the total number of categories, and a value of 1 indicates the presence of a particular feature.

B. Learning Representations for Tabular Data

Recent work has shown that words can be embedded into vectors based on the distributional hypothesis [25]. Entity linking tasks [2], [16] have been solved using such embeddings. A graph generated using the embedding representation of entities can be used to disambiguate entities for the entity linking task. Embeddings have also been used to understand the layout of a table [26].

Deep Neural Networks (DNN)-based architectures have been proposed for table classification [27] using attention mechanisms [28]. Nishida et. al [27] proposed a Recurrent Neural Network (RNN)-based architecture to encode a sequence of tokens for each cell. An attention mechanism is further used to extract the important token from each cell to form an input volume. This constructed volume is then passed through a CNN and then a Fully Connected Network (FCN) to classify tables. After training, each cell token is assigned an embedding.

Ghasemi-Gol and Szekely [15] introduced an unsupervised method to generate a vector representation of a table for the table classification task. Their proposed table embedding is based on cell tokens embedding using four contexts: text within each cell, text in adjacent cells, text in the corresponding attribute or header, and text surrounding the table in the web page. Trabelsi et al. [29] also proposed contextual embeddings generated from the information present in the tables. They learn word embeddings for attribute tokens by enlarging the context to cover the metadata of tables. The additional learned context is then used while ranking queries against tables. Though our method also uses the cell context to learn table embeddings, we use a globalized context. This means that the entire table's content is used by our model in order to generate the embeddings. Also, unlike others, we do not require any metadata, which are often missing online.

TabIESim [17] considers a combination of intrinsic and extrinsic sources that were previously never used for the table retrieval task. Intrinsic table similarity is measured using passage level information, while extrinsic table similarity uses a regularized manifold-based ranking approach. TabIESim combines both similarities using a simple, re-ranking approach. The proposed approach results in a significantly better retrieval quality outperforming semantically rich baselines.

While most works have augmented tables using semantic features like concepts, entities, word and graph embeddings [13], [14], [30], TAPAS [20], and Hybrid-BERT-Row-Max [18] are two approaches that extend deep contextualised language models like BERT [19] to generate table representations. These methods extend BERT's architecture to encode tables as input.

C. Schema Preserving Approaches

The table's schema typically imposes some structure on the table's content. However, the information from the column names does often not provide enough information as it might be short, abbreviated, or hard to interpret. Though the initial motivation behind CNNs [31] was for computer vision related tasks, there have been insights on the adaptation process of a non-image data to an image for CNN architectures [32]–[35]. TAPAS [20], a BERT [19] inspired model encodes the structural information in the positional embeddings, but is designed for the task of semantic parsing. Specific to table retrieval, MTR [21] makes a novel use of Gated Multimodal Units (GMUs) to learn a joint representation of the query and different table modalities. In our work, we propose a CNN-based architecture to train tabular data and preserve the 2D structure of a table. Unlike other approaches, our method uses the content of a table only, and not the metadata. This makes our method more robust and applicable to most real-world settings.

III. ConvTab: A Context-Preserving Model for Ad-Hoc Table Retrieval

In this section, we present our method called ConvTab, a CNN-based architecture for ad-hoc table retrieval.

1) Problem Formalization: Given a keyword query q, ad-hoc table retrieval is the task of returning a ranked list of tables, $(T_1, ...T_n)$ from a collection of tables C. Each table is assigned a relevance score to the query $score(q, T)$. Tables are then sorted in descending order of their scores.

a) Anatomy of a Web Table: Our primary focus is on tables which are embedded in webpages, also known as web tables. Below, we define the basic elements of a web table. The following information is available for each table in our corpus:

- *Table page title*: the title of the webpage that embeds the table.
- *Table caption*: a short textual label that summarizes what the table is about.
- *Table headings*: a list of labels that define what each table row/column contains.
- *Table cell*: a table cell is specified with the row index i and column index j. Table cells are considered as atomic units in a table. It is possible that some of the tables are not populated with a value in each of the corresponding cells.
- *Table row*: a list of table cells that form a row in the table.
- *Table column*: a list of table cells that form a column in the table.
- *Table entities*: tables often mention specific entities, such as persons, organizations, or locations. Table entities are a set consisting of all the entities that are mentioned in the table.

Figure 2 shows our proposed approach architecturally. We perform the task of table retrieval in two phases. First, we train the model on the TabEL [36] dataset comprising of 1.6M tables, along with negative sampling for the table classification task. After completing this training task, the features from the penultimate fully-connected layer are extracted and used for generating semantic features for the query and table. These

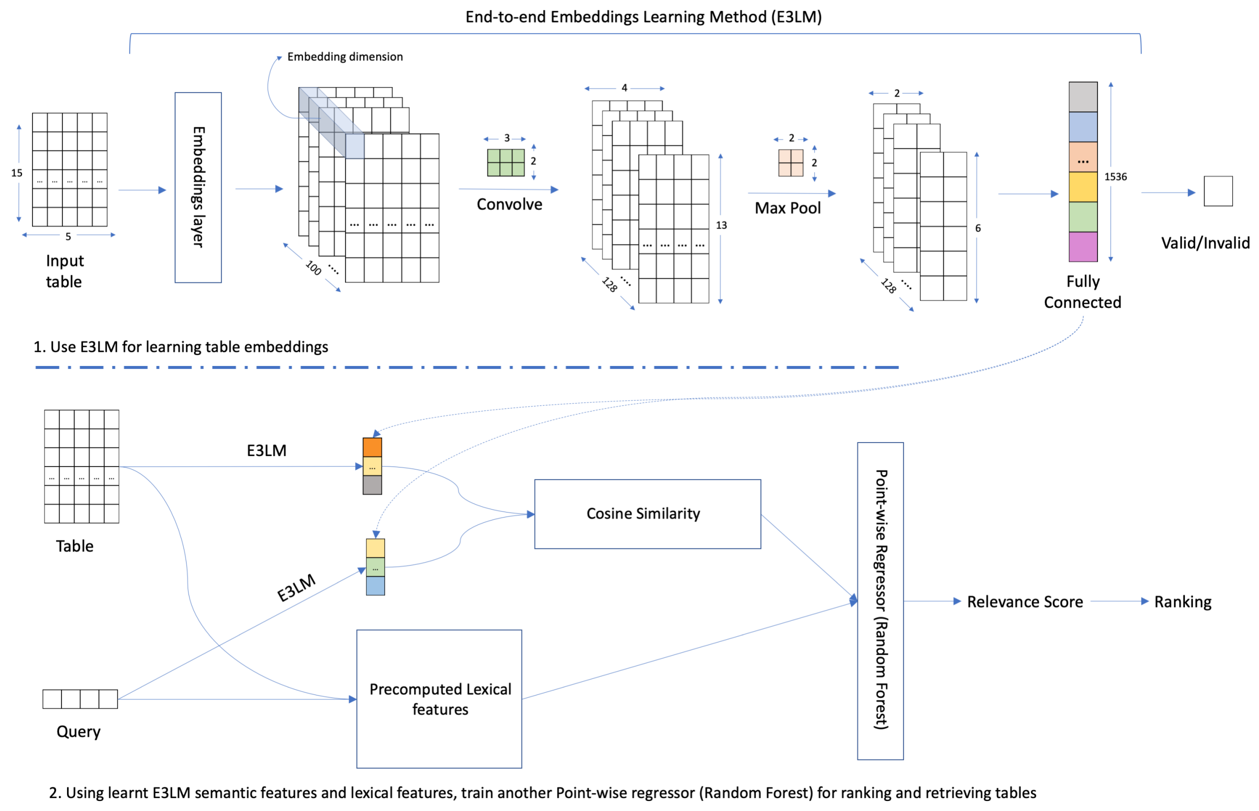

End-to-end Embeddings Learning Method (E3LM)
Embedding dimension
15
5
Input table
Embeddings layer
100
3
2
Convolve
4
13
128
2
2
Max Pool
2
6
128
1536
Fully Connected
Valid/Invalid
1. Use E3LM for learning table embeddings
Table
E3LM
E3LM
Query
Cosine Similarity
Precomputed Lexical features
Point-wise Regressor (Random Forest)
Relevance Score
Ranking
2. Using learnt E3LM semantic features and lexical features, train another Point-wise regressor (Random Forest) for ranking and retrieving tables

Fig. 2: Our proposed approach is divided into two phases. First, we train on the TabEL dataset [36] for table classification. In the second phase, the features from the penultimate layer are extracted and used to generate query, table, and query-table similarity-based features. These features along with lexical features are used as input to the learning algorithm which predicts the ranking of a table.

semantic features, along with lexical features, are used as input to the learning algorithm that ranks the tables for each query. This is further explained in the following sections.

A. *Learning Embeddings*

The textual value of a table cell typically does not contain much information. For example, some text snippet such as "President" may occur in multiple contexts. The embeddings based on cell values only are insufficient, and thus we leverage additional context from the table to generate a more meaningful representation. Our proposed embeddings are contextual and use co-occurrence information present in tabular data. In fact, as the data is arranged in 2-D matrices, we apply a 2-D convolution-based neural network architecture without changing the structure of the tabular data. We treat the tables as images where cells represent pixels, while the dimension of the embeddings represents the channel space of colored images.

a) Convolutional Architecture: The field of deep learning for image classification has considered many architectures. Among them, CNN [31] is most-commonly used to perform image classification. A typical CNN architecture consists of convolutional layers and pooling layers that are followed by fully-connected layers to generate outputs for image classification.

We consider a 2-D convolution for an input matrix X of size I×J, which is convolved with the kernel matrix K_c of size m×n, resulting in a new matrix Y_c, representing the output matrix by the following equation:

$$Y_c[i,j] = \sum_{\alpha=0}^{n} \sum_{\beta=0}^{m} K_c[\alpha,\beta] \cdot X[i+\alpha, j+\beta] \tag{2}$$

Here, the indices i and j belong to the image matrix X while α and β belong to the kernel matrix. The kernel matrix is similar to the filter matrix that is used in computer vision. The resulting output from this convolution is then passed to a maxpooling function:

$$Y_{mp} = \max(Y_c^{\gamma\times\gamma}) \tag{3}$$

which returns the maximum value Y_{mp} from the input patch $Y_c^{\gamma\times\gamma}$, where $Y_c^{\gamma\times\gamma}$ is a sub-matrix of Y_c with dimension $\gamma\times\gamma$. We apply the previous equation consecutively on all patches of the original matrix Y_c. The result is a feature map of lower dimension.

b) Generating negative samples: Our model training task is designed as a table classification task, where we predict the table category to be 'valid' or 'invalid'. As the TabEL [36] dataset is a collection of valid 1.6M tables, we generate artificial negative data using a random sampling of entities that do not occur together in tables. Negative sampling [37] is a technique used to train machine learning models that generally have several orders of magnitudes more negative observations compared to positive ones, but these negative observations are not explicitly provided and instead must be generated somehow.

The TabEL corpus is preprocessed as explained in Section IV-A1. This preprocessing results in a corpus of 2.2 million tables, which serves as the 'valid' category dataset for our training part. For the 'invalid' category, we create synthetic tables by shuffling cells from the entire table corpus and constructing tables such that cells that occur in the same table are not present together in any 'valid' table. For the 'invalid' category, we create synthetic tables by first creating empty tables and then filling them up randomly with cells from the other tables in the corpus. We ensure that no two cells in this 'invalid' table come from the same 'valid' table. The intuition behind this step is that entities in the same table are semantically closer and by this random shuffling, we create tables with semantically distant entities. For example, let us assume we have two tables in our dataset; Table A contains countries and its currencies, and Table B contains medicines and diseases. A potential 'invalid' table in this scenario could be the one where we shuffle the cells across these tables and it contains 'Dollar, Insulin and Common Cold, Russia'. Intuitively, we train the model to distinguish between tables that have similar data and those that do not. We create an equal number of negative tables to maintain class balance. This results in a total corpus of 4.4 million tables, which are fed as input to the end-to-end table (E3LM) classification model.

1) ***Training Network Parameters****:* Figure 2, top, shows the first phase of our approach: the end-to-end learning method, E3LM. First, we initialize the embeddings of all entities present in our model from a Gaussian sphere $\mathcal{N}(0,1)$. The dimension of each embedding vector is set to 100. These embedding dimensions correspond to the image channel space for our input data.

We use I×J tables as an input to our CNN-based model. We use a CNN with a fully connected layer architecture for our E3LM phase. The motivation here is to generate high-level features for the tables, which can be used to compute the semantic similarity. We picked rectangular kernels of shape $\alpha \times \beta$ for our convolution layer, and then do a maxpool with a kernel of size $\gamma \times \gamma$. The intermediate output is flattened and passed through a Fully Connected layer, and a sigmoid activation is used to get an output of size 1. This output is further rounded off in order to get 1 or 0 for our 'valid' or 'invalid' classification, respectively.

We calculate the loss using Binary Cross Entropy [38].

$$L = -\sum_{n=1}^{D} y_n \cdot \log(p(y_n)) + (1-y_n) \cdot \log(1-p(y_n)) \quad (4)$$

where D is the total number of data points, y_n is the original label and $p(y_n)$ is the predicted label of the instance n.

B. Applying Embeddings to Table Retrieval

After this pretraining step, we use the model for our downstream task of table retrieval to generate table-query semantic features. We input the tables and keyword queries to the trained model and extract the penultimate layer features.

We presume that the feature vectors present in the penultimate layer are rich enough for our table-query matching. We extract these features for both the query and table using a forward pass through the network. We treat keyword queries in tabular form in order to pass it through the CNN model to extract the features. Once we have gathered the features for the table and query, we calculate the cosine similarity score between the two as follows:

$$similarity(\vec{T},\vec{Q}) = \frac{\mathbf{T}\cdot\mathbf{Q}}{\|\mathbf{T}\|\|\mathbf{Q}\|} = \frac{\sum_{i=1}^{n} T_i Q_i}{\sqrt{\sum_{i=1}^{n} T_i^2}\sqrt{\sum_{i=1}^{n} Q_i^2}} \quad (5)$$

where $\vec{T}$ is a table feature vector, $\vec{Q}$ is a query feature vector, and *n* is the dimension of the feature vector (1536 in our case). This score serves as the semantic feature for our table query ranking task.

The cosine similarity semantic feature and the precomputed lexical features are used together to train another Random Forest model for ranking our tables for a given input query q. The objective is to classify a given query table pair and predict its relevance score. Subsequently, a score is computed as the weighted sum of the confidence score from the model as follows and is used to rank the tables:

$$score = \sum_{c=0}^{2} c \times p(c) \quad (6)$$

where c is the relevance score class (from 0-2) and $p(c)$ is the probability of that class obtained from Random Forest. This scoring function is the same as that used by [13].

For training the Random Forest, we use five-fold cross-validation. For each fold, score is computed as described in the above equation. Then, all these folds are merged together which is then used to compute NDCG scores.

IV. Experiments & Results

This section first presents our experimental setup, the datasets we used in our empirical evaluation, as well as our baselines. Subsequently, we give a full description of the results we obtain.

A. Setup

In this section, we present the setup of our experiments and comment on the hyperparameters for our models.

Method	NDCG@5	NDCG@10	NDCG@15	NDCG@20	MAP	MRR
LTR [13]	0.5223	0.5422	0.5711	0.5915	0.4112	0.7244
STR [13]	0.5951	0.6293	0.659	0.6825	0.5141	0.7579
T2vW [14]	0.5974	0.6096	0.6312	0.6505	0.4675	0.7806
T2vE [14]	0.5602	0.5569	0.5760	0.6161	0.4176	0.7344
TabIESim [17]	0.6498	0.6479	-	0.6935	0.5124	-
Hybrid-BERT-Row-Max [18]	0.6361	0.6519	0.6558	0.6564	0.6311	0.6673
MTR [21]	0.6631	0.6813	-	0.7370	0.6058	-
ConvTab	**0.7698***	**0.7646***	**0.7768***	**0.7987***	**0.6540***	**0.8347***
ConvTab (with 1-D Conv)	0.4240*	0.4736*	0.5447*	0.6199*	0.4022*	0.5727*
ConvTab (with MLP)	0.2826*	0.3393*	0.3842*	0.4374*	0.2488*	0.4343*

TABLE I: Table retrieval evaluation results using NDCG scores, MAP, and MRR for the proposed approach against SoTA table retrieval methods. The missing entries indicate metrics that were inaccessible due to missing run files.

1) Dataset and Preprocessing: Our training set is the TabEL dataset [36] that contains the full-set of $1,652,771$ high-quality Wikipedia tables. Each table has five indexable fields: table caption, attributes (column headings), rows, the title of the page, and section. In addition, each table contains statistics on the number of columns, number of rows, and set of numerical columns of the table.

Our benchmark set of queries and tables are the same as used by Zhang and Balog [13]. This is a collection of 60 test queries from two independent sources manually ranked against 3120 tables. These 3120 manually annotated table query pairs consist of 18 lexical features like query features (number of query terms, etc.), table features (number of rows, columns, empty cells etc.), and table-query features (query term frequency, language modeling score, etc.). This dataset also contains relevance scores for each table query pair annotated on a scale of 0-2, 0 being non-relevant (i.e., when it is unclear what it is about or when it is about a different topic), 1 being somewhat relevant if some cells or values could be used from this table, and 2 being highly relevant if large blocks or several values could be used from it when creating a new table on the query topic.

We preprocess the tables to remove special symbols, numbers, duplicate/empty rows and columns from the entire TabEL corpus. This is a standard preprocessing step in order to remove noise from the data. In addition, we remove tables of shape 1×1 since they do not have any context. Each cell in a table is considered as a single entity, which is used to construct the vocabulary for the model. We consider all tokens with a frequency of two or more, the rest of the tokens are replaced with an "unknown" ¡UNK¿ token. The CNN architecture requires fixed size input. So, we convert our tables into 15×5 size, which is the mean row and column size of the tables in the corpus. We pad smaller tables with ¡UNK¿ tokens across the rows and columns, and split larger tables into smaller chunks of 15×5. This split increases the total number of tables to 2.2 million in the training corpus. We preprocess the queries in a similar manner as the tables. Further, we study the effect of fixing the table dimensions on ranking results in Section IV-E.

2) Hyperparameter Settings: We set the dimension of word embeddings k, to 100. Our penultimate layer has 1536 rich features, which are used for the final binary classification result of the table being *valid* or *invalid*. Our model is trained for 40 epochs with a batch size of 128. We use Binary Cross Entropy [38] for minimizing the loss function and updating the model weights. The Learning Rate (LR) is set at 6×10^{-6} with a decay set to $0.5 \times LR$ after each epoch. We set our Random Forest to 800 trees, the maximum features for each tree being 4. Our model is implemented using PyTorch[3] and trained on four 12GB Titan X GPUs on a machine with a 64GB RAM and 16 CPU cores.

3) Evaluation metrics: We evaluate the performance of our proposed methods and baselines on the table retrieval task using Normalized Discounted Cumulative Gain (NDCG) [39] at cut-off thresholds 5, 10, 15, and 20. All NDCG results are reported using the TREC evaluation script, trec_eval[4]. We also report Mean Average Precision (MAP) [40], and Mean Reciprocal Rank (MRR) [41] as the evaluation metrics for our retrieval task. We use a paired Student's t-test at the 0.05 level to test significance and denote it using $*$.

B. Baselines

We consider the following baselines from the literature:

Lexical Table Retreival(LTR) [13] method uses standard statistical approaches for generating lexical features from the query, table, and table-query for the table retrieval task. Query features include the number of query terms [42] and the sum of query IDF scores [43]. Table features include the number of rows, columns, NULLs, inlinks, table importance, and pageViews [1], [8]. Query-table features consists of combined features [1], [8], [44] (for instance, the total query term frequency in the table body). All features used by the LTR baseline are described in [13]. The baseline uses a point-wise regression-based ranking method like Random Forest with 1000 trees and maximum 3 features in each tree.

Semantic Table Retrieval(STR) [13] represents queries and tables in the same semantic space and measures the similarity of those semantic representations. STR introduces various semantic features such as word embeddings, graph

[3]https://pytorch.org/
[4]https://github.com/usnistgov/trec_eval

Query	Rel	LTR	STR	HBRM	CONVTAB
Query: stocks					
Stocks for the Long Run / Key Data Findings: annual real returns	2	-	6	**4**	**4**
TOPIX / TOPIX New Index Series	1	9	-	2	7
Hang Seng Index / Selection criteria for the HSI constituent stocks	1	-	-	3	5
Query: ibanez guitars					
Ibanez / Serial numbers	2	**1**	2	4	2
Corey Taylor / Equipment	1	2	3	2	5
Fingerboard / Examples	1	4	5	5	3
Query: board games number of players					
List of Japanese board games	1	13	1	1	1
List of licensed Risk game boards / Risk Legacy	1	-	3	19	4
Query: cereals nutritional value					
Sesame / Sesame seed kernels, toasted	2	**1**	8	2	**1**
Query: irish counties area					
Counties of Ireland / List of counties	2	2	**1**	-	**1**
List of Irish counties by area / See also	2	**1**	2	1	2
List of flags of Ireland / Counties of Ireland Flags	2	-	3	10	3
Provinces of Ireland / Demographics and politics	1	4	4	3	4
Toponymical list of counties of the United Kingdom / Northern . . .	1	-	7	2	5
MÃžscraige / Notes	1	-	6	6	6
Query: external drives capacity					
Comparison of encrypted external drives / Features	2	3	2	-	**1**
Xerox 820 / Disk storage	1	-	8	4	3
Comparison of encrypted external drives / Background information	1	1	3	9	2

TABLE II: Example queries from the query set. 'Rel' denotes the true relevance score of a table. The other columns indicate the rank at which each model retrieves the corresponding table. A lower rank for a table with a true relevance score of 2 is a positive indicator of the retrieval method. For brevity, 'HBRM' denotes the Hybrid-BERT-Row-Max [18] model.

embeddings, bag-of-entities, and bag-of-categories. It also uses pretrained Google News word embeddings[5]. These semantic features, along with lexical features, are used to learn the ranking of the tables. This baseline also uses a random forest as the learning algorithm.

T2vW and T2vE baselines were introduced in Zhang et. al [14] and are table-centric. T2vW and T2vE embeddings are learned using words and entities that appear in the table, respectively, rather than using pretrained Google News word embeddings in STR. These embeddings also provide additional semantic features similar to those introduced in STR.

TabIESim [17] utilizes both intrinsic and extrinsic table similarities for enhanced ad-hoc table retrieval. The intrinsic similarity features are passage-based while the extrinsic similarity features are manifold-based. Both similarities are combined via a cascade re-ranking approach. This approach results in a significantly better table retrieval quality outperforming strong semantic baselines.

Hybrid-BERT-Row-Max [18] uses the BERT language model for the task of ad hoc table retrieval. BERT is used to extract features from the query, the corresponding table context fields, and additional pieces of information from the table (row, column or cells). These features are concatenated into a single feature vector, which is fed into a regression layer to predict the relevance score.

[5]https://code.google.com/archive/p/word2vec/

MTR [21] follows a two-step ranking process. In the first step, an initial pool of candidate tables is retrieved in response to a query by some underlying (basic) retrieval method. In the second step, MTR re-ranks the tables according to their estimated relevance based on several multimodal table properties. MTR uses Recurrent Convolutional Neural Networks (RCNNs) to process both textual (natural language) input and the query.

C. Research Question

We aim at answering the following research questions through our empirical evaluation:

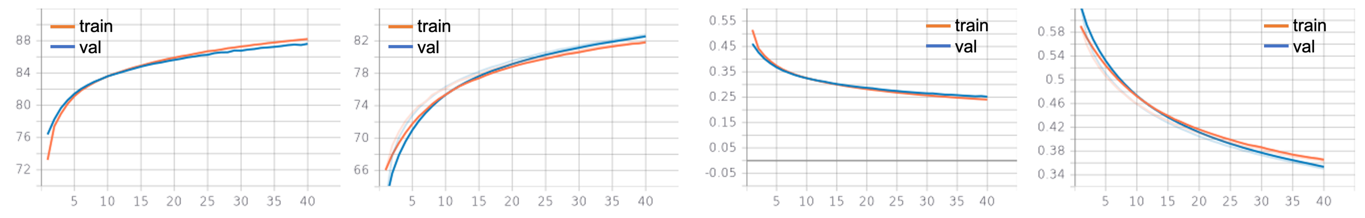
- **Q1**: How effective is our convolution-based architecture at enhancing state-of-the-art models for ad-hoc table retrieval?
- **Q2**: Does fixing the tables in the test data corpus to a specific dimension affect the ranking?
- **Q3**: How effective is it to use a 2-D convolution architecture compared to a 1-D convolution architecture for training the embeddings of the tables?
- **Q4**: What is the ideal model configuration for our convolution-based architecture balancing both effectiveness and simplicity for our task?

D. Results - Q1

Table I shows the results of our model (ConvTab) and the baselines. We observe that our proposed architecture results

Variations of our Model	**NDCG@5**	**NDCG@10**	**NDCG@15**	**NDCG@20**
1×2-DConv + 2×FC	0.7652	0.7633	0.7753	0.7972
1×2-DConv + 3×FC	0.7665	0.7659	0.7805	0.7993
2×2-DConv + 3×FC	0.7670	0.7620	0.7785	0.7985
3×2-DConv + 2×FC	0.7650	0.7630	0.7790	0.7979
1×2-DConv + 1×FC	0.7698	0.7646	0.7768	0.7987

TABLE III: Table retrieval evaluation results using NDCG scores at cutoff of 5,10,15,20 for the proposed approach with different model configurations. We choose the simplest of these configurations for our target task, considering a tradeoff between model complexity and resulting NDCG scores.



(a) (b) (c) (d)

Fig. 3: Accuracy and loss curves obtained during model training for the table classification task (on training and validation data). (a) Accuracy of the model with 2-D convolution. (b) Accuracy of the model with 1-D convolution. (c) Loss of the model with 2-D convolution. (d) Loss of the model with 1-D convolution.

in the best query-table semantic features, outperforming the previous SoTA, MTR, by a margin of 16.9% - 8.37% on NDCG scores at a cutoff range of 5 - 20, respectively. It is interesting to note that the top-two approaches on this task both use a CNN-based model – which indicates that preserving the structure of the table is likely advantageous for learning tabular representations. In contrast to MTR, our method does not need contextual information or metadata from the tables. This makes our method more robust as it can be used in settings where this information is not available. Interestingly, Hybrid-BERT-Row-Max uses BERT model to generate embedding representation of data but yet performs worse than MTR and ConvTab. We believe this is due to the loss of the structural information in tables while feeding flattened tables to the BERT model.

With respect to MAP and MRR, Table I indicates that our approach outperforms Hybrid-BERT-Row-Max in terms of MAP by a margin of 3.62% and T2vW [14] in terms of MRR by a margin of 6.9%. We note that though Hybrid-BERT-Row-Max is a strong baseline, positional embeddings are not robust enough to represent structural information as compared to CNN-based approaches.

To illustrate our results, we report in Table II qualitative results for some of the queries from our test set compared to other baselines whose run files were publicly available.

E. Results - Q2

One of the requirements of the proposed architecture is to fix the dimension of the tables, because of CNN based architecture. In this subsection, we study the effect of fixing the input dimension on the ranking results.

In our case, we have chosen the mean value of the table dimensions in our dataset as the input dimension (15×5), so that most of the tables can be represented effectively. As the size of the tables in the test data can vary, this can potentially mean that larger tables might suffer from significant data loss. To study the impact of this potential data loss, we perform an experiment where we randomly sample multiple 15×5 dimension inputs from each table in the test set.

It is interesting to note that even after this window sampling, the resulting NDCG ranking score remained the same. After careful examination, we found that though the feature representation of the table were different from the ones in the original setting, the similarity score between a table and the corresponding query remained the same as before. As the random forest learning algorithm uses cosine similarity as its semantic feature, the results produced were practically the same.

With this experiment, we observe that fixing the dimension of the table resulted in no data loss, which in turn does not affect the NDCG ranking of the results. This implies, that our proposed method can be used to learn effective table representations for tables of varied dimensions in a real-world setting.

F. Results - Q3

One of the important claims that we make in this paper is the importance of preserving the structure of a table when learning the cell embeddings. To study this point further, we report the difference in training results when using a model for table classification with our best parameter configuration, but with 1-D convolutions.

Table I shows the NDCG results, which clearly indicate that the results with 2-D convolution outperform 1-D convolution by a significant margin. This could be due to the fact that 1-D convolution does not fully capture the table structure and therefore is not able to encode the relationship between column and row entities. Furthermore, Figure 3 shows a comparison of our training accuracy and loss in 1-D convolution compared to the 2-D convolution scenario. We note that while the 2-D convolution-based architecture achieves approximately 88% accuracy, the 1-D convolution-based architecture achieves a training accuracy of about 82% for our classification task.

G. Results - Q4

With the experiments in the previous subsection, we conclude that the features generated through a 2-D convolution architecture are significantly better for our task than those generated through 1-D convolutions. In search of an ideal model, we experimented with several configurations of the 2-D convolution-based model.

Table III shows the different model configurations with 2-D convolutions and the corresponding NDCG scores on our target dataset. We achieved the optimum configuration with a simple $1\times$ 2-D convolution followed by one fully connected layer, considering the trade-off between architectural complexity and resulting performance. We note that more complex models resulted in a very similar accuracy while training the end-to-end embedding learning model, which was followed by similar NDCG scores. We believe that this is probably due to the relative simplicity of the target task. The additional layers usually translate to additional dimensionality, which is not required by the task at hand.

For all the reported experiments so far, we used a random forest classifier for a fair comparison with previous work. As an additional experiment, we replaced the Random Forest for computing the ranking with a Multilayer Perceptron classifier (MLP) [45]. This MLP consisted of three Linear layers with a Tanh activation and a Dropout layer. The final layer output was captured using softmax activation. Table I shows the results with MLP, which were subpar. We believe that due to the lack of complexity in the test data and their small size, MLPs are not a good fit for this particular ranking task. Our benchmark set of 60 queries and 3120 tables also restricted us to only use point-wise ranking methods. In order to use some other ranking method like pair-wise [46], [47] or list-wise [46], [47], the data ideally should have been of the form where two tables are compared, and similarly in case of list-wise, the ideal dataset should have been of the form where a list of tables is ranked for a given query. In our case, since a single best table is given for a particular query, point-wise ranking methods are more appropriate and therefore we used standard classification and regression algorithms.

V. Conclusion & Future Work

In this paper, we proposed a novel approach to solve the task of ad-hoc table retrieval. We generate semantic features for query-table matching in two phases. First, we train an end-to-end machine learning model for the task of table classification on the TabEL corpus. The features from the trained model are subsequently used for generating semantic features for both the query and tables. We performed a number of experiments to validate our approach. In the end, we show that our approach outperforms the state of the art by a significant margin. Moreover, we performed experiments to study the effects of 1-D convolutions compared to 2-D convolutions on our target task, and showed that 2-D convolution architectures performed significantly better in our context. Furthermore, we tried several model configurations with 2-D convolutional architectures and picked a balanced configuration in terms of time, complexity, and resulting ranking.

One frequent critique of CNNs and deep learning architectures is that they lack of interpretation, making it difficult to know what exact features were used for the classification. In future work, we would like to tackle this issue by revisiting the target task as well as our approach from an explainability perspective. We will also extend our current keyword-based search retrieval with a table-based retrieval scenario.

VI. Acknowledgements

This work was supported by the Swiss National Science Foundation under grant number 407540 167320, and by a ThinkSwiss Research Scholarship.

References

[1] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: exploring the power of tables on the web," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 538–549, 2008.

[2] Y. Eslahi, A. Bhardwaj, P. Rosso, K. Stockinger, and P. Cudré-Mauroux, "Annotating web tables through knowledge bases: A context-based approach," in *2020 7th Swiss Conference on Data Science (SDS)*. IEEE, 2020, pp. 29–34.

[3] G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and searching web tables using entities, types and relationships," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1338–1347, 2010.

[4] P. Venetis, A. Y. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, and G. Miao, "Recovering semantics of tables on the web," 2011.

[5] J. Madhavan, L. Afanasiev, L. Antova, and A. Halevy, "Harnessing the deep web: Present and future," *arXiv preprint arXiv:0909.1785*, 2009.

[6] S. Sarawagi and S. Chakrabarti, "Open-domain quantity queries on web tables: annotation, response, and consensus models," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 711–720.

[7] M. Zhang and K. Chakrabarti, "Infogather+ semantic matching and annotation of numeric and time-varying attributes in web tables," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 145–156.

[8] C. S. Bhagavatula, T. Noraset, and D. Downey, "Methods for exploring and mining tables on wikipedia," in *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, 2013, pp. 18–26.

[9] M. J. Cafarella, A. Halevy, and N. Khoussainova, "Data integration for the relational web," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 1090–1101, 2009.

[10] R. Pimplikar and S. Sarawagi, "Answering table queries on the web using column keywords," *arXiv preprint arXiv:1207.0132*, 2012.

[11] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer, "The mannheim search join engine," *Journal of Web Semantics*, vol. 35, pp. 159–166, 2015.

[12] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: entity augmentation and attribute discovery by holistic matching with web tables," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012, pp. 97–108.

[13] S. Zhang and K. Balog, "Ad hoc table retrieval using semantic similarity," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1553–1562.

[14] L. Zhang, S. Zhang, and K. Balog, "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*, 2019, pp. 1029–1032.

[15] M. Ghasemi-Gol and P. Szekely, "Tabvec: Table vectors for classification of web tables," *arXiv preprint arXiv:1802.06290*, 2018.

[16] A. L. Gentile, P. Ristoski, S. Eckel, D. Ritze, and H. Paulheim, "Entity matching on web tables: a table embeddings approach for blocking." in *EDBT*, 2017, pp. 510–513.

[17] R. Shraga, H. Roitman, G. Feigenblat, and M. Canim, "Ad hoc table retrieval using intrinsic and extrinsic similarities," in *Proceedings of The Web Conference 2020*, 2020, pp. 2479–2485.

[18] Z. Chen, M. Trabelsi, J. Heflin, Y. Xu, and B. D. Davison, "Table search using a deep contextualized language model," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 589–598, Jul. 2020, arXiv: 2005.09207. [Online]. Available: http://arxiv.org/abs/2005.09207

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[20] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," *arXiv preprint arXiv:2004.02349*, 2020.

[21] R. Shraga, H. Roitman, G. Feigenblat, and M. Cannim, "Web table retrieval using multimodal deep learning," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Virtual Event China: ACM, Jul. 2020, pp. 1399–1408. [Online]. Available: https://dl.acm.org/doi/10.1145/3397271.3401120

[22] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford *et al.*, "Okapi at trec-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[23] S. Zhang and K. Balog, "Design patterns for fusion-based object retrieval," in *European Conference on Information Retrieval*. Springer, 2017, pp. 684–690.

[24] H. Zamani, B. Mitra, X. Song, N. Craswell, and S. Tiwary, "Neural ranking models with multiple document fields," in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 700–708.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[26] M. Ghasemi-Gol, J. Pujara, and P. Szekely, "Learning cell embeddings for understanding table layouts," *Knowledge and Information Systems*, pp. 1–26, 2020.

[27] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo, "Understanding the semantic structures of tables with a hybrid deep neural network architecture," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[28] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[29] M. Trabelsi, B. D. Davison, and J. Heflin, "Improved table retrieval using multiple context embeddings for attributes," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 1238–1244.

[34] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceed-*

[30] K. Y. Gao and J. Callan, "Scientific table search using keyword queries," *arXiv preprint arXiv:1707.03423*, 2017.

[31] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[32] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, "Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Scientific reports*, vol. 9, no. 1, pp. 1–7, 2019.

[33] P. Rosso, D. Yang, and P. Cudré-Mauroux, "Beyond triplets: Hyper-relational knowledge graph embedding for link prediction," in *Proceedings of The Web Conference (WWW 2020)*, Taipei, Taiwan, 2020. [Online]. Available: https://exascale.info/assets/pdf/rosso2020www.pdf *ings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[35] M. S. Singh, V. Pondenkandath, B. Zhou, P. Lukowicz, and M. Liwickit, "Transforming sensor data to the image domain for deep learning — an application to footstep detection," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2665–2672.

[36] C. S. Bhagavatula, T. Noraset, and D. Downey, "Tabel: entity linking in web tables," in *International Semantic Web Conference*. Springer, 2015, pp. 425–441.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[38] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[39] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[40] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[41] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya, "Structured learning for non-smooth ranking losses," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 88–96.

[42] S. Tyree, K. Q. Weinberger, K. Agrawal, and J. Paykin, "Parallel boosted regression trees for web search ranking," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 387–396.

[43] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "Letor: A benchmark collection for research on learning to rank for information retrieval," *Information Retrieval*, vol. 13, no. 4, pp. 346–374, 2010.

[44] J. Chen, C. Xiong, and J. Callan, "An empirical study of learning to rank for entity search proceedings of the 39th annual international acm sigir conference on research and development in information retrieval,(sigir 2016)," 2016.

[45] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, classifiaction," 1992.

[46] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009. [Online]. Available: http://dx.doi.org/10.1561/1500000016

[47] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," vol. 227, 01 2007, pp. 129–136.