Contextual Fact Ranking and Its Applications in Table Synthesis and Compression

Silu Huang[1], Jialu Liu[2], Flip Korn[2], Xuezhi Wang[2], You Wu[2], Dale Markowitz[2], Cong Yu[2]
[1]University Of Illinois at Urbana-Champaign, shuang86@illinois.edu (work done while visiting Google)
[2]Google Research, {jialu, flip, xuezhiw, wuyou, dalequark, congyu}@google.com

ABSTRACT

Modern search engines increasingly incorporate tabular content, which consists of a set of entities each augmented with a small set of facts. The facts can be obtained from multiple sources: an entity's knowledge base entry, the infobox on its Wikipedia page, or its row within a WebTable. Crucially, the informativeness of a fact depends not only on the entity but also the specific *context* (e.g., the query).

To the best of our knowledge, this paper is the first to study the problem of *contextual fact ranking*: given some entities *and* a context (i.e., succinct natural language description), identify the most informative facts for the entities collectively within the context.

We propose to contextually rank the facts by exploiting deep learning techniques. In particular, we develop *pointwise* and *pairwise* ranking models, using textual and statistical information for the given entities and context derived from their sources. We enhance the models by incorporating entity type information from an IsA (hypernym) database. We demonstrate that our approaches achieve better performance than state-of-the-art baselines in terms of MAP, NDCG, and recall. We further conduct user studies for two specific applications of contextual fact ranking—table synthesis and table compression—and show that our models can identify more informative facts than the baselines.

ACM Reference Format:
Silu Huang[1], Jialu Liu[2], Flip Korn[2], Xuezhi Wang[2], You Wu[2], Dale Markowitz[2], Cong Yu[2]. 2019. Contextual Fact Ranking and Its Applications in Table Synthesis and Compression. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330980

1 INTRODUCTION

Today's search engines employ rich structured data as advanced features such as entity cards (e.g., Google's Knowledge Panel or Bing's Satori). The importance of displaying facts that are meaningful and informative in the context of those features has been demonstrated in [1, 12]. E.g., given the query `[nu couché]` (a famous painting by Amedeo Modigliani), users would generally prefer general facts for the entity, such as `artist-name` and `year-completed`, to idiosyncratic ones such as `venue-of-sale`.

Now consider the following two queries: `[paintings by amedeo modigliani]` and `[most expensive paintings sold]`. The informativeness of associated facts is different in these two cases. For example, `artist-name` is redundant for the former while informative for the latter. Facts like `medium` (e.g., canvas) and `year-completed` are arguably more informative for the former while `price` and `buyer` are informative for the latter. Clearly, the informativeness of a fact depends not only on the entities but also on their context.

In this paper, we study the problem of contextual fact ranking: given a set of entities accompanied by a succinct natural language description (in the form of a title or query), identify the most informative facts from a universe of candidate facts. We focus on the following two use cases.

Table synthesis. Answering list-seeking queries using tables on the Web has become a staple feature for search engines [1]. However, there is a lack of content for torso/tail queries, hence the ability to automatically synthesize tables to expand coverage is desirable. The raw sources for those synthesized tables can come from named lists on the Web [15] or Wikipedia category pages[1]. In both cases, a collection of entities are added to the synthesized table and facts are retrieved from their respective Wikipedia infoboxes. For example, from the Wikipedia category page *Open world video games*[2], a synthesized table can be constructred for all such games, leading to a carousel [12] as depicted in Figure 1. Naturally, the ability to rank facts according to their informativeness to the query context is crucial to the quality of the synthesized table.

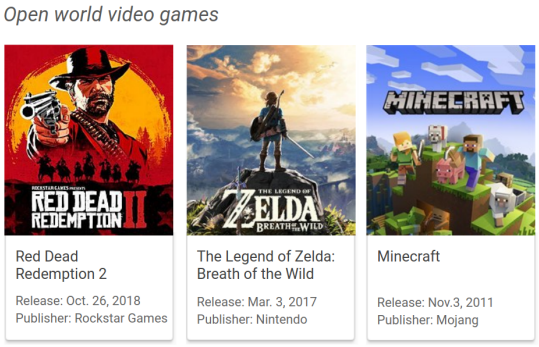

Figure 1: Carousel from a Synthesized Table

Table compression. Even when a table exists to support a table answer, it is often not suitable for presentation as is due to limited visual real estate and needs to be "compressed." For example, given the query `[future tallest buildings]`, Google used to show the table (Figure 2, left) that is generated by projecting the first four columns of the original table (Figure 2, right). Most users, however, would prefer facts such as `height`, `floors`, or `year-of-completion`, which are more informative to query terms `future` and `tallest`. This again motivates our problem of ranking facts based on the context.

While existing work has considered fact ranking for entity summarization and table augmentation (see Section 2 for more details), to the best of our knowledge this is the first paper to study contextual fact ranking given a set of entities *and* a textual context as input. We take a supervised deep learning approach and make

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330980

[1]https://en.wikipedia.org/wiki/Help:Category
[2]https://en.wikipedia.org/wiki/Category:Open_world_video_games

Future Tallest Buildings

Rank	Building	City	Country	Height (m)	Height (ft)	Floors	Year of completion
1	Jeddah Tower	Jeddah	Saudi Arabia	1008 m	3307 ft	167	2020
2	Burj Khalifa	Dubai	UAE	828 m	2,717 ft	162	2010
4	Suzhou Zhongnan Center	Suzhou	China	729 m	2,392 ft	137	?
5	Dubai One Tower	Dubai	UAE	711 m	2,333 ft	161	2021
6	PNB118	Kuala Lumpur	Malaysia	644 m	2,113 ft	118	2021
7	Signature Tower Jakarta	Jakarta	Indonesia	638 m	2,093 ft	113	2021

Figure 2: Table Answer (left), underlying Wikipedia Table (right)

the following contributions. First, we compiled a training corpus at scale by making use of existing relational tables from Wikipedia after significant preprocessing to annotate tables based on page signals. Second, using this training data, we designed a novel multi-tower neural architecture for learning an LSTM-based model for contextual fact ranking. We considered both pointwise and pairwise variants using the learning-to-rank paradigm. Finally, we present a comprehensive set of experiments against multiple non-trivial baselines and demonstrate the superiority of our approach.

2 RELATED WORK

Our study of the contextual fact ranking problem is in many ways inspired by prior research and production work on WebTables [5, 7]. [25] considers the problem of **table synthesis** by stitching together multiple tables of the same schema; this is fundamentally different from the table synthesis application we are interested in. A related research problem on WebTables is **table augmentation**. The overarching idea here is to *extend* a given table by slot-filling the missing fact values, adding more entity rows, or extending to more fact columns [6, 19, 20, 32, 41]. The core technique for fact column extension is to leverage attribute/attribute or entity/attribute correlations from the WebTable corpus [6, 41] or search query sessions [19], and rank the facts based on what's already in the table. The problem is different from contextual fact ranking because no textual context is considered. Nevertheless, the idea of attribute correlation is noteworthy and our experimental results in Section 7 demonstrate the superiority of our deep learning approach over the correlation-based approach. [42] goes beyond simple table content based extension by incoporating user provided instructions such as a query and looks at the problem of, given a user query and a knowledge base, composing a table that can answer the user's query. However, the main motivation here is to *answer* the user query and *not* to suggest or rank facts. For example, given a query `[database company revenue]`, the table answer would contain only two facts, `name` and `revenue`. In contextual fact ranking, however, facts like `market-capitalization` or `headquarter` are also potentially informative, but cannot be retrieved by [42]. Furthermore, it is not clear how [42] can handle queries like `[tallest buildings in the world]`, where there is no explicit notion of "tallest" in the knowledge base.

Most of the research on **table compression** investigates sampling strategies for choosing table rows so as to minimize information-loss metrics (e.g., [9, 38]). One exception is [14], which studies attribute ordering for SQL query results over a table that return a ranked subset of rows. Gao et. al. [16] studies table compression for archival in order to reduce storage, by making use of attribute correlation and arithmetic coding. Those works are only marginally related to our work here.

Learning to rank refers to training a ranking model using machine learning techniques. Li et. al. [24] summarized different approaches of learning to rank into three categories: pointwise [13, 37], pairwise [4, 21], and listwise approach [8, 43]. Specifically, the pointwise approach transforms the ranking task into a classification problem and the loss function is defined on each single object. The pairwise approach transforms the ranking task into a classification on the order of the object pair, and the loss function is defined on each pair. Finally, the listwise approach looks at the entire list all together and the loss function considers the position of all objects per query at a time. Following those directions, we consider both pointwise and pairwise approaches in our design of the deep learning ranking model. Listwise approach is hard to be applied in our case due to the difficulty in obtaining list-like labels.

In the semantic web community, **entity summarization**, namely selecting top-k facts for a given entity, has received some attention. Specifically, SUMMARUM [40] ranks triples based on PageRank scores of the involved entities, LinkSum [39] considers both PageRank and BackLink in their ranking function, and RELIN [11] further extends the random surfer model by taking the informativeness of a fact and the relatedness of each edge into consideration. Furthermore, FACES [17] tries to improve the conprehensiveness by selecting diversified facts via clustering. In addition to single entity, REMES [18] considers the problem of finding top-k facts collectively for a set of entities and formulates it as a joint optimization problem. In this way, different entities can potentially augment with each other by maximizing the relatedness of different entities' facts. **Semantic association ranking** is to rank the semantic relationship for a given entity pair. RankSVM [10] and actively learning [2] has been proposed for personalized ranking and minimizing labels collected from the user. None of those entity summarization and semantic association work, however, incorporate a context as part of the ranking consideration.

Finally, long short-term memory network (**LSTM**), which has seen increased adoption in NLP tasks [27, 33, 34, 36], is a special kind of recurrent neural network that can capture the long-term dependency. It can fall short in encoding the input sentence concisely when the input sequence is long. Attention mechanism jointly learns the embedding and alignments between different modalities. Similar to the visual attention of human, attention mechanism distributes different focus over different input parts and summarizes the input into a more accurate embedding. We exploit LSTM network *without* attention in our ranking task. This is because word sequences in title and facts are short and using memory cells like LSTM can already capture the term dependency. Siamese neural networks have been proposed for different tasks [3, 23, 30]. A siamese neural network consists of twin NNs with exactly the same architecture and weight, each taking an input modality, e.g., sentence or image. The outputs of these two NNs are then fed into a contrastive

loss layering, comparing against each other and calculate the loss. Our pairwise model (Section 6) can be considered as a siamese neural network.

3 PROBLEM FORMULATION

We consider the problem of ranking candidate facts given a descriptive title[3] (e.g., `[future tallest buildings]`) and a collection of entities (e.g., *Jeddah Tower*, *Wuhan Greeland Center*, etc.). We assume a universe of entities where each entity E is associated with a set of facts $\Phi(E) = \{F\}$ and each fact $F = \langle f_1, f_2, \ldots, f_{|F|} \rangle$ is a word sequence denoting a fact name (e.g., `country`, `year-of-completion`, etc.); the associated facts are indexed on E and could come from a knowledge base, Wikipedia infoboxes, WebTables, etc. For the purposes of this paper, we use Wikipedia infoboxes. Let $\mathcal{F}$ denote the union of facts over a set of entities $\mathcal{E}$, that is, $\mathcal{F}_\mathcal{E} = \bigcup_{E \in \mathcal{E}} \Phi(E)$.

Problem (Contextual Fact Ranking). *Given a context $T = \langle t_1, t_2, \ldots, t_{|T|} \rangle$, where t_i are word tokens, and a collection of associated entities, $\mathcal{E}$, rank the candidate facts $\mathcal{F}_\mathcal{E}$ (or simply $\mathcal{F}$ when context is clear).*

In both table synthesis and table compression applications, we use context available on the web page instead of user queries, i.e., page title and table caption. As a natural byproduct of the problem definition above, we can retain only the *top-k* facts, where k is dictated by the application. For instance, if we would like to display them as a table answer in mobile search, k is usually set to 3 in order to fit the table on the phone screen.

Admittedly, when judging the informativeness of a fact for a collection of entities within a context, there is some level of subjectivity. However, there are many cases where much agreement would be expected. As an example, for the context `[paintings by amedeo modigliani]`, the fact `artist` is clearly redundant, hence uninformative, because all painting entities that match the context would have the same value (i.e., *Amedeo Modigliani*). Furthermore, we hypothesize that for a given context, humans are often more capable of judging the relative informativeness of two facts, than judging the absolute informativeness of a single fact, and that aggregation over multiple individuals provides a reliable measure of informativeness. Therefore, we propose to rank facts via supervised learning on a crowdsourced dataset. Unfortunately, there is no large scale labeled dataset of entities with a described context and a "gold standard" of which facts are informative and which are not; hence, we resort to a weak supervision approach based on which attributes are included in Wikipedia relational tables and assuming those that are not included are less informative, as we discuss in more detail in Section 4.

We would like our model to develop an appropriate representation that carries semantics for the context and each fact, inferring only from their respective word sequences but learning across lots of training examples. With that in mind, we develope two deep learning approaches, pointwise (Section 5) and pairwise (Section 6), for the ranking task leveraging the train data generation process we describe in Section 4.

4 TRAINING DATA GENERATION

One natural way to collect the training data is through human labelling. However, human labelling can be costly and developing an effective deep learning model typically requires large amount of training data. Thus, we propose to collect training labels automatically at scale from the open Web. Specifically, we make use of the human-constructed WebTables in Wikipedia. The assumption is that, w.r.t. the table title, facts[4] that are picked by the editors to constitute columns of the table are more salient to the title than facts that are not picked. For instance, Figure 2 (right) is a table from Wikipedia, and we regard facts in the table, e.g., `height` and `year-of-completion`, to be more important than facts that are not in the table, e.g., `management` and `owner`, w.r.t. title `[future tallest buildings]`.

Note that not all columns in the table are facts: the `rank` column, in the above example, does not contain values that are intrinsic to the entities. Thus, the first challenge in this training data generation approach is to identify which columns are entities and facts. For the entities, we adopt methods introduced in [12], which are shown to be very effective. We describe how to identify fact columns next. As defined in Section 3, $\Phi(E)$ is the set of facts stored in each entity E's profile. When generating the training data, we use each entity's Wikipedia infobox as the main profile.[5] However, matching columns in a table and facts from infoboxes of entities in the table is not a trivial problem. We exploit some heuristic methods to achieve the potentially fuzzy alignment. First, for each fact name, we check its synonyms in a dictionary which can be built based on method described in [7]. We say a fact is mapped to a column if their names are synonyms to each other. Second, we check the values for each fact. If a column shares most of its cells' string content (60% in our study) with the fact values in the corresponding entity profile, we consider it a match. Finally, for the remaining columns in the table that do not have any aligned facts in the infobox, we try to search each cell's string content in its associated entity's Wikipedia page, respectively. If most of the cells (60% in our study) in the column can be matched, we also consider the column as a fact and adds the column to the profile. This is reasonable because infobox has limited space to include all the facts. The unmapped columns are ignored.

The next challenge is to ensure the model being aware of the difference among facts recognized from table columns, e.g., `height` should be ranked higher than `country` in Figure 2. The answer is to rely on redundancy in the WebTable corpus. Besides `[future tallest buildings]`, Wikipedia also has `[tallest buildings in the world]`, `[tallest buildings before 1900]`, etc. As they are constructed by different people, their schemas are not always the same but the most cirtical columns likely will be shared.

We use $\langle T, F, I \rangle$ and $\langle T, F_1, F_2, I \rangle$ to denote the individual training example needed by pointwise and pairwise approaches, respectively. Here T and F denote title and fact, respectively, and I is the indicator denotes the labeling. For pointwise, $I = 1$ indicates the fact is informative for the title, and for pairwise, $I = 1$ indicates F_1 is more informative than F_2. In the following, we will describe in details how to generate $\langle T, F, I \rangle$ and $\langle T, F_1, F_2, I \rangle$. First, among all candidate facts $\mathcal{F}$ for T, let $\mathcal{F}_+^T \subseteq \mathcal{F}$ denote the ones mapped to column(s) of the table, and the unmapped ones as $\mathcal{F}_-^T$, i.e., $\mathcal{F}_-^T = \mathcal{F} - \mathcal{F}_+^T$.

Pointwise. Given a title T, we generate a training record $\langle T, F_+, I = 1 \rangle$ for each fact $F_+ \in \mathcal{F}_+^T$ and $\langle T, F_-, I = 0 \rangle$ for each fact $F_- \in \mathcal{F}_-^T$. That is, we consider the facts in the table as informative to the title, while other facts that are not in the table as not informative.

[3] We use the terms *context* and *title* interchangeably.

[4] When referring to WebTables, we use the terms *fact* and *attribute* interchangeably.

[5] Note that access to each entity's Wikipedia infobox is easy because an entity in the table often is linked to its Wikipedia page.

Table 1: Statistics of training data sets

Dataset	#pages	#tables	#examples	#positives	#negatives
Pointwise	45K	82K	1.1M	0.25M	0.86M
Pairwise	45K	82K	3.2M	-	-

Continuing the example in Figure 2, the facts in the table, e.g., `height`, `year-of-completion`, etc., are labelled as positive for the title `[future tallest buildings]`. While facts that are in each entity's profile but not in the table, are labelled as negative.

Pairwise. Given a title T, we generate a training record $\langle T, F_+, F_-, I = 1\rangle$ for each pair $(F_+, F_-) \in \mathcal{F}_+^T \times \mathcal{F}_-^T$. That is, we consider the facts in the table as more informative to the title than those that are not. Continuing the example in Figure 2, the facts in the table, e.g., `height`, `year-of-completion`, etc., are labelled as more informative than those that are not in the Wiki table, with respect to the title `[future tallest buildings]`.

In the pointwise data set, we observed conflicts in labels about whether or not the same fact is informative for similar tables. This is because the pointwise approach assumes a clear distinction between informative and uninformative facts, which is a subjective notion where people may disagree. Nonetheless, aggregated over the many tables in our corpus, we believe the overall training data quality is solid and leads to good performing models as we show in Section 7. This problem is less severe for the pairwise approach since the labels are relative and the model is optimized for ranking accuracy. As a concrete example, consider two tables with similar titles `[tallest buildings in (los angeles|new york city)]`, respectively. Both tables have fact `height` and neither has fact `management`. Only the `los angeles` table has fact `primary-purpose`. The pointwise approach may attribute the inclusion/exclusion of `primary-purpose` to the difference between `los angeles` and `new york city`, hence causing overfitting. The pairwise approach, however, would handle this case nicely by learning `height` > `primary purpose` > `management`.

Finally, while there is less noise in the pairwise training data set, only the pointwise approach can predict whether or not a given fact is informative to a title. And since we have a much larger training set for the pairwise approach, training can be more time-consuming for the pairwise approach than for pointwise. To have a better understanding of the pros and cons of the two approaches, we study both of them and compare their performances empirically.

Table 1 provides statistics of the two training sets. After removing tables with too few columns or rows (≤ 3), and requring a valid table to contain at least one matched fact column as well as a title, we are left with around $82K$ tables from $45K$ Wikipedia pages. From these tables, we obtained $1.1M$ training examples from the pointwise approach and $3.2M$ for pairwise.

5 POINTWISE APPROACH

The pointwise approach considers the ranking task in Problem 3 as a binary classification problem, i.e., classifying whether a fact F is informative with respect to a title T and associated entities $\mathcal{E}$. In this section, we describe the basic model (Section 5.1), followed by the enhanced model (Section 5.2) that incorporates knowledge from the mapping between entities and notable collections.

5.1 Basic Model

Figure 3 depicts our basic dual encoder architecure, similar to [26], with six layers. Note again that the main inputs are the title T and fact F, both are raw text tokenized into a sequence of words. Additional statistical features are extracted from the fact values of entity members. Next, the *Word Embedding Layer* prepares the words from input title and fact as embedding vectors. Those vectors are refined using LSTM in the *Contextual Embedding Layer* and in turn aggregated back into a title vector and a fact vector in the *Summarization Layer*. The *Scoring Layer* measures the semantic similarity between the two vectors, $S_{T,F}$. Finally, the *Output Layer* combines $S_{T,F}$ with statistical features from *Input Layer* to output $g_{T,F}$ as the probability that fact F is informative to title T. We describe each layer in details next.

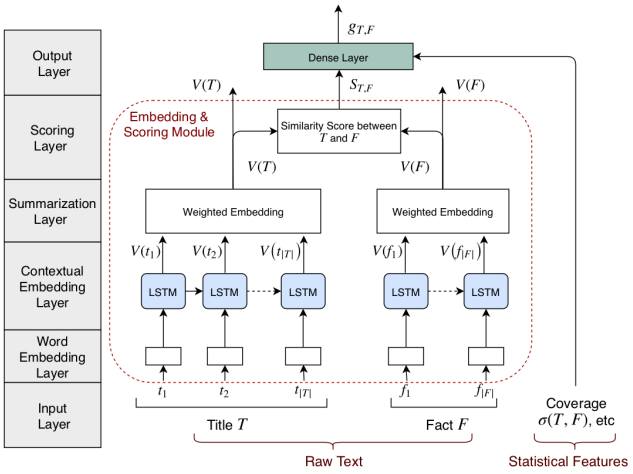

Figure 3: Model for Pointwise Approach

(1) Input Layer. As shown in the bottom part of Figure 3, we use both the textual information and the statistical features extracted from the fact values of entity members as our input. The textual information is represented by two word sequences, i.e., $T = \{t_1, t_2, \cdots, t_{|T|}\}$ and $F = \{f_1, f_2, \cdots, f_{|F|}\}$; by ingesting such textual information into our model, we expect our model to learn the semantic similarity between each title T and fact F. The statistical features include *coverage*—the percentage of entity members with the fact; and *distinct*—a boolean indicating whether this fact always has the same value across all entity members.

These two classes of information are orthorgonal to each other, and can be combined to improve our model's performance. We will describe this in the last layer, i.e., output layer.

(2) Word Embedding Layer. Given a word sequence (a title or a fact), we first map each word to a high-dimensional vector space using fastText [28] pre-trained word embeddings.[6] The reason for using pre-trained word embedding is two-fold. First, using one-hot encoding for each word would result in a vector space of very high dimension, which leads to more trainable parameters in the model and increases the training time complexity. Second, pre-trained word embedding can serve as a good initialization for our model and help improve our model's performance, especially when our training data is not sufficiently large and certain words are not mentioned many times.

(3) Contextual Embedding Layer. In this layer, we try to transform each word's initial pre-trained embedding to a trained embedding that is specific to our ranking task. Specifically, we feed the sequence of initialized word embeddings into a recurrent neural network with LSTM cell, where the output embedding of each word

[6] We tried word2vec [29] and Glove [31] and fastText turned out to be slightly better.

also depends on its preceding words. Note that the LSTM cells in the title and fact are with independent trainable parameters. Let $V(t_i)$ and $V(f_j)$ be the output of each LSTM cell for word $t_i \in T$ and $f_j \in F$ respectively, as shown in Figure 3. Assume the output size of each LSTM cell is d for both title and fact, then we have $V(t_i), V(f_j) \in \mathbb{R}^d$. Intuitively, we want our LSTM cell to map the pre-trained embeddings to a refined embedding space, where semantically related title and fact are placed close to each other. As validated in our experiments, compared to the pre-trained word embedding (input of this layer), the trained embedding (output of this layer) has higher quality and achieves significant improvement for our ranking task. This contextual embedding layer plays an important role in our model.

(4) Summarization Layer. After obtaining the trained embedding for each word in the title and fact, we can now fuse these embeddings into one representative embedding for the title T and fact F, denoted as $V(T)$ and $V(F)$, respectively. As shown in Equation 1, the representative embedding is formulated as a weighted embedding across all words, where w_i encodes the importance of each word and $\sum_i w_i = 1$.

$$V(T) = \sum_i w_i V(t_i)$$
$$V(F) = \sum_j w_j V(f_j) \tag{1}$$

We have tried different strategies in determining the weight w_i, including simple pre-defined strategies and more advanced attention mechanism. For pre-determined weights, we can either use the last word's embedding as our representative embedding, i.e., $V(T) = V(t_{|T|})$, or distribute equal weight to each word, i.e., $V(T) = \frac{\sum_i V(t_i)}{|T|}$. Furthermore, we also tried using the prevailing attention mechanism. Specifically, in order to compute w_i for each word t_i in the title T, we first calculate its similarity score between t_i and each word f_j in the fact, then perform a max pooling over all words f_j and use this similarity score as w_i. Such attention mechanism aims to assign larger weights to more salient words which are closer to the other side. However, since the number of words in each title and fact is quite small (smaller than 15 in most cases), exploiting memory cells like LSTM can already capture the long-term dependency and there is not much gain in using attention mechanism. We choose the pre-determined equal weight strategy due to its efficiency and comparable performance to the others in our experiments.

(5) Scoring Layer. Now that we have the representative embeddings $V(T)$ and $V(F)$, we can then measure the semantic similarity between T and F based on some similarity metric. Here the similarity metric can be as simple as dot product with no learned variable, or similarity metric with lots of variables to be learned, e.g., bilinear function. We observed similar performance across these similarity metric in our experiments. Thus, we opt for the simple dot product as our scoring function, formulated as below in Equation 2.

$$S_{T,F} = V(T) \cdot V(F) \tag{2}$$

(6) Output Layer. In this layer, we try to combine the textual signal ($S_{T,F}$) with other signals extracted from entity members (e.g., coverage). Specifically, we treat the similarity score $S_{T,F}$ as one feature based on the semantic meaning and feed it, together with other statistical features, into a dense layer with two classes as the output, i.e., informative or not. We then perform a softmax over the two output classes and obatin the probability of whether fact F is informative to title T, i.e., $P(I = 1|T, F) = g_{T,F}$.

Loss Function. As a classification task, it is natural to use cross entropy as our loss function as shown in Equation 3, where $\ell_{pt}(T, F, I)$ denotes the cross entropy for each title T and fact F, while L_{pt} represents the overall loss across all title-fact pairs. Specifically, $\ell_{pt}(T, F, I)$ takes the label I and the predicted probability $g_{T,F}$ as the input, and compute the two-class cross entropy.

$$L_{pt} = \sum_T \sum_{F \in \mathcal{F}} \ell_{pt}(T, F, I)$$
$$\ell_{pt}(T, F, I) = -(I \log(g_{T,F}) + (1 - I) \log(1 - g_{T,F})) \tag{3}$$

5.2 Enhanced Model

In this section, we aim to improve our model by incorporating extra information from the Knowledge Base. This is motivated by the observation that some phrases in the title are not very informative even after applying pre-trained embeddings. For instance, it is hard for the model to recognize that *Amedeo Modigliani* is a painter known for graceful portrayal of the human form. To mitigate this problem, we replace (when applicable) recognized entities with the corresponding notable collection in an internal IsA Knowledge Base. In this way, the title `[paintings by Amedeo Modigliani]` can be mapped to `[paintings by painting artist]`. In order to distinguish these two title representations, we call the title before replacement *original title*, denoted as T_o, and the mapped title *collection title*, denoted as T_c. We propose an enhanced model by exploiting both T_o and T_c, as illustrated in Figure 4. Compared to the initial model in Figure 3, the input layer and output layer are the same in our enhanced model. There are three new layers: mapping layer, embedding & scoring layer, and title-weighting layer.

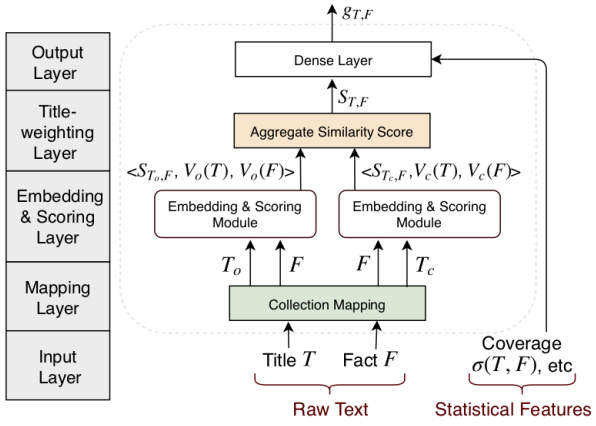

Figure 4: Enhanced Model with Collection Mapping

Mapping layer. Mapping layer generates a collection title T_c by referring to a IsA Knowledge Base. Note that we do not map the fact F, as F is in general interpretable and there is no benefit to substitute the fact name. After the mapping layer, we now have the original title T_o, the collection title T_c, and the fact F.

Embedding & Scoring Layer. In this layer, we try to calculate the semantic similarity between T_o and F and that between T_c and F, respectively. This is exactly the same as what we have done for title T and fact F in Section 5.1. Thus, we apply the embedding & scoring module (the red dotted area in Figure 3) for the two title-fact pairs, $\langle T_o, F \rangle$ and $\langle T_c, F \rangle$. As a result, we get a semantic similarity

score along with the representative embeddings for each pair, i.e., $\langle S_{T_o,F}, V_o(T), V_o(F)\rangle$ and $\langle S_{T_c,F}, V_c(T), V_c(F)\rangle$.

Title-weighting Layer. With two alternative semantic similarity scores $S_{T_o,F}$ and $S_{T_c,F}$, the remaining question is how to combine them as our final similarity score $S_{T,F}$. In some cases, the collection title T_c is generic and can guide the model to pick the right fact, e.g., `painting artist`. However, in some other cases the collection title can be too broad to be useful, e.g., `person`. Thus, we need to integrate these two alternative scores differently based on the context. In particular, we let our model learn the importance for $S_{T_o,F}$ and $S_{T_c,F}$ in different scenarios, denoted as w_o and w_c respectively. The weight is computed by first calculating a score for each title-fact pair (Equation 4), followed by a softmax normalization (Equation 5).

$$\gamma_x = W_x \cdot [V_x(T); S_{T_x,F}] \tag{4}$$

$$w_x = \frac{exp(\gamma_x)}{\sum_{x' \in \{c,o\}} exp(\gamma_{x'})} \tag{5}$$

where $x \in \{c, o\}$; $W_x \in \mathbb{R}^{d+1}$ is a trainable weight vector; [;] is a vector concatenation. Finally, we can represent $S_{T,F}$ as a weighted sum of $S_{T_o,F}$ and $S_{T_c,F}$, formalized in Equation 6.

$$\begin{aligned} S_{T,F} &= w_o S_{T_o,F} + w_c S_{T_c,F} \\ S_{T_o,F} &= V_o(T) \cdot V_o(F) \\ S_{T_c,F} &= V_c(T) \cdot V_c(F) \end{aligned} \tag{6}$$

6 PAIRWISE APPROACH

An alternative approach is to frame the ranking task in Problem 3 as a pairwise classification problem, i.e., classifying the relative ordering of two facts (F_1, F_2) by their informativeness in relation to the title T. Yet we would like to ensure the ranking is transitive, i.e. if $\delta_T(F_1, F_2) > 0$ and $\delta_T(F_2, F_3) > 0$, then $\delta_T(F_1, F_3) > 0$, where $\delta_T(F_1, F_2) > 0$ represents F_1 is more informative than F_2 for title T. Therefore we use Siamese network [23] to learn a informative scoring function $g_{T,F}$ as an intermediate step towards the ranking function $\delta_T(F_1, F_2)$. As shown on Figure 5(a), a Siamese network consists of two identical base neural networks with the same architectures and weights. Each base model is itself a pointwise scoring function $g_{T,F}$ that maps title-fact pairs to ranking scores. As a result, although we train a Siamese network on examples consisting of pairs of facts, we can use just one of these base models at serving time (illustrated in Figure 5(b)), by producing a score $g_{T,F}$ for individual fact and ranking the candidate facts accordingly. Hence we can always obtain consistent pairwise comparisons among all facts $\mathcal{F}$ and thus derive a ranking in descending order of $g_{T,F}$.

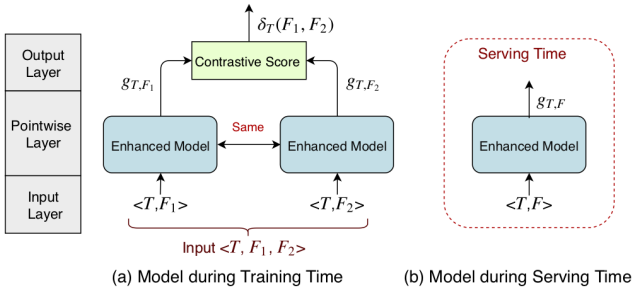

Figure 5: Model for Pairwise Approach

More specifically, for a given training example $\langle T, F_1, F_2, I\rangle$, we feed $\langle T, F_1\rangle$ pair in the left twin base model and $\langle T, F_2\rangle$ pair in the right twin base model. For the base model, we simply use the enhanced model (Figure 4) described in the pointwise approach. Afterwards, we feed the output of each base model into the output layer, where we differentiate between the pointwise score g_{T,F_1} and g_{T,F_2} and output the contrastive score $\delta_T(F_1, F_2)$ as illustrated in Equation 7.

$$\delta_T(F_1, F_2) = g_{T,F_1} - g_{T,F_2} \tag{7}$$

Loss Function. A typical pairwise loss function used in the literature is hinge loss [22]. We formalized the loss function in Equation 8 below, where L_{pr} can be interpreted as the summation of each fact pair's hinge loss across all titles, while $\ell_{pr}(T, F_1, F_2)$ takes $\delta_T(F_1, F_2)$ and the real label I as the input. However, since $\langle T, F_1, F_2, I = 1\rangle$ is equivalent to $\langle T, F_2, F_1, I = 0\rangle$, we can get rid of the redundant fact pairs and only use $\langle T, F_+, F_-, I = 1\rangle$ as our training examples. Thus, I is always 1 and $\ell_{pr}(T, F_+, F_-)$ can be simply formulated as a hinge loss function on $\delta_T(F_+, F_-)$.

$$\begin{aligned} L_{pr} &= \sum_T \sum_{F_+ \in \mathcal{F}_+^T} \sum_{F_- \in \mathcal{F}_-^T} \ell_{pr}(T, F_+, F_-) \\ \ell_{pr}(T, F_+, F_-) &= \max(1 - \delta_T(F_+, F_-), 0) \end{aligned} \tag{8}$$

7 EXPERIMENTS

In this section, we present empirical results in order to validate the two proposed approaches, organized as follows. First, we assess the model performance on a hold-out test set from data generated as described in Section 4. Then, we examine whether the proposed models are effective in the two applications mentioned in Section 1, namely table synthesis and table compression, based on a user study. Finally, a case study is presented to illustrate when our models do or do not work.

7.1 Performance on Test Set

Our first goal is to evaluate whether the proposed approaches are effective in learning a good ranking function $g_{T,F}$ on the training set described in Section 4. To ensure fairness, we created a hold-out set at training to avoid label leaks. Specifically, training and test sets were generated from separate Wikipedia pages[7] so that they are less likely to share the same table schemas.

Recall that at serving time, the prediction score $g_{T,F}$ from pointwise and pairwise approaches are not directly comparable due to their different neural architectures at training time. To address this problem, for each of around 14,000 test examples, we define a total order on the union of all candidate facts $\mathcal{F}$ over the entities, based on each method's respective ranking function, and apply standard IR metrics [35] in comparison to the existing table facts as follows:

- **MAP:** mean average precision over each recall size from 1 . . . $|\mathcal{F}|$
- **Average NDCG:** average of normalized discounted cumulative gain for each $\langle T, \mathcal{F}\rangle$.
- **Average Recall@k:** average of recall at k for each $\langle T, \mathcal{F}\rangle$.

In addition, we compare the pointwise and pairwise approaches with two baselines that can be trained on the same data sets and leave variations of our own model in the later model study:

- *FactCorrelations*: In [7], a greedy algorithm was proposed to leverage fact correlation statistics for table schema auto-completion. Here we start from a full set of candidates and use their heuristic to iteratively prune one fact while trying to maximize schema coherence.

[7] We grouped Wikipedia pages by titles after removing numbers as further restriction.

Figure 6: Peformance comparison on test set

Table 2: Quality contribution (MAP) by components

Model	All	-LSTM	-Statistical Features	-Collection Title
Pointwise	**0.862**	0.605	0.838	0.853
Pairwise	**0.902**	0.632	0.842	0.882

- *TitleFactCorrelations*: Generalizing the idea from *FactCorrelations*, one can also leverage co-occurrences between entities from the table title and facts from respective table schema to condition fact statistics on the given table title.

The results are available in Figure 6. Both pointwise and pairwise approaches significantly outperform the two baselines across all metrics. This is not surprising because the proposed models are supposed to learn deeper patterns underlying the training data while the baselines only exploit shallow correlation statistics. Among the baselines, *TitleFactCorrelations* beats *FactCorrelations* since the latter ignores the title completely.

Between the two proposed approaches, pairwise slightly outperforms pointwise mainly due to the noise during training data generation for the latter discussed in Section 4. We note that the performance gap between the proposed approaches is much smaller than that between baselines, indicating the noise in pointwise training labels does affect model quality, but not significantly. Another interesting observation is that the two approaches have similar average recall@5 but pairwise achieves better average recall@3. This makes us believe that the pairwise approach is stronger at picking the most critical facts among a set of informative fact candidates.

Model Study. Since the overal neural architecture has several components, it is interesting to experiment these model variations after removing certain parts, as a justification of the effectiveness of the combined models. Table 2 lists the MAP metric after removing three major components separately. The highest number from multiple runs is reported. First of all, it is clear that the models integrating all three components achieve the best performance. Next, comparing the remaining columns, removing LSTM (which transforms each word's initial pre-trained embedding to a trained embedding) led to the biggest drop in performance, indicating the importance of learning word embeddings based on our training data. It is also worth noting the similarity of this setting with RankSVM [22] where features fed to SVM objective is static and the only variable is the weight matrix/vector. Realizing the gain brought by training LSTM, we further conduct experiments that allows pre-trained embeddings to be adaptable at the word embedding layer with and without LSTM layer. Unfortunately, we observe lower performance and we conjecture this is due to the lack of large amount of training data. Meanwhile, it is also worth noting that statistical features is quite helpful as it provides complementary structural information which cannot be inferred from textual context. Finally, the last column confirms the effectiveness of the enhanced model proposed in

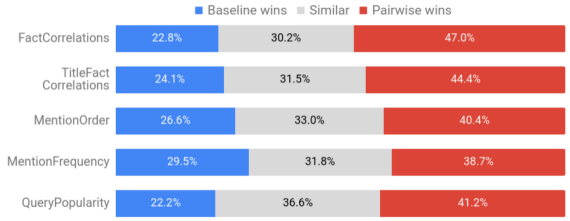

Figure 7: Side-by-side comparison on table synthesis

Section 5.2, which can improve generalizability by utilizing an IsA Knowledge Base. Compared to the other two components, the gain brought by collection title is considerably smaller. The reasons are twofold. Firstly, there can be annotation errors in our IsA Knowledge Base which undermines our collection title quality. Secondly, collection titles benefit more when the original title has long-tail entities, unlike statistical features and LSTM, which are equally applicable to all inputs.

7.2 Performance on Applications

We conducted a user study to evalute our proposed models in the two applications of table synthesis and compression and report the performance here, using a production crowdsourcing platform similar to Mechanical Turk to collect ratings. We focus on the pairwise ranking model here, since it exhibited better performance than the pointwise model on the test set, and compare it against various baselines as described below. (The pointwise model gave similar results but with slightly worse numbers compared to the pairwise model. Detailed results are omitted due to space limit.)

Each task consists of a title and two sets of facts (three facts each) generated by the methods being compared, for the same set of three entities (chosen based on popularity[8]). Each entity was presented as a thumbnail image with an entity name below it; clicking on either of these would open a new tab in the browser to the entity's Wikipedia page in case the rater needed background on the entity for the task. The two sets of facts are presented side-by-side (randomly swapped; each side has a similar look and feel to Figure 1). Each task is rated independently by 3 raters who were asked which side contains a better set of facts. To ensure rater diversity, each rater can rate no more than 6 tasks.

We observe overall 85% of the tasks with at least 2/3 majority votes, with no significant differences across tasks and approaches.

7.2.1 Table Synthesis. We made use of the titled entity lists extracted from Wikipedia category pages such as https://en.wikipedia.org/wiki/Category:Open_world_video_games and selected the most popular categories. We pruned categories according the following criteria (which yielded a total of ~1K categories): (1) Categories containing either too many (>200) or too few (<5) entities were dropped; (2) Categories having simple titles that can be annonated by single entity (e.g., `[video games]`) were dropped; (3) Only entity members belonging to the category's dominant type (e.g., *games* for `[open world video games]`) based on having the same Infobox templates were considered; (4) Only facts common to enough entities (at least 60%) were considered for ranking.

In the previous subsection, we were studying the quality of different approaches and neural structures learning from the same training data set. For the table synthesis task, we compared our pairwise approach with the previous two correlation statistics baselines as well as the following new methods: (1) *MentionOrder*: Facts are

[8]Wikipedia pageview can be obtained from https://tools.wmflabs.org/pageviews.

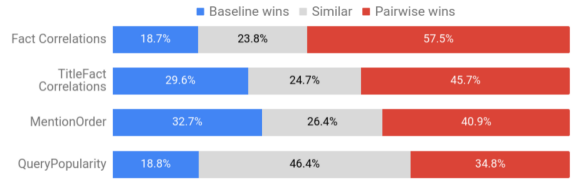

Figure 8: Side-by-side comparison on table compression

ranked based on Infobox template position on the assumption that more important facts are placed higher; (2) *MentionFrequency*: Facts are ranked based on the frequency of mentions among all entity Infoboxes from the category with ties broken using Infobox template position order; (3) *QueryPopularity*: Similar to *TitleFactCorrelations*, we take entity-attribute queries [19] from a search engine query stream and find the most frequent attributes that co-occur with any entity from the title.

Side-by-side results for the various methods compared against pairwise are listed in Figure 7. Assuming scores of raters preferring pairwise approach, baseline and considering a tie are 1, -1 and 0, respectively, we obtained using the bootstrap method the 95% confidence intervals for each of the five baseline methods as [0.184, 0.281], [0.145, 0.241], [0.082, 0.183], [0.04, 0.134] and [0.135, 0.227]—all in favor of the pairwise method. The bars for *FactCorrelations* and *TitleFactCorrelations* are consistent with those reported in the test set evaluation, indicating the effectiveness of our training sets for the purpose of fact ranking in this application. *MentionOrder* and *MentionFrequency*, neither of which consider title, received better ratings than the other baselines. This confirms that position ordering in Wiki Infoboxes does indeed capture importance. However, they are clearly limited since they ignore the title. For example, for `[skyscraper office buildings in chicago]`, `type` is ranked among top 3 by both methods, but it is uninformative as it is redundant w.r.t. the title ("office" building).

7.2.2 Table Compression. We selected 100 Wikipedia tables due to their value for knowledge exploration given a set of news articles published during a 3-month window across topics such as politics, sports, entertainment, technology and economy. Some examples include `[largest trading partners of china]`, `[fifa world cup top goalscorers]`, etc. We conducted similar side-by-side evaluations (between the pairwise model and the baselines) as we did for *table synthesis*; however, we dropped *MentionFrequency* as a baseline since all columns in the table have the same number of mentions among all table rows.

Results are shown in Figure 8 with 95% confidence intervals as [0.324, 0.415], [0.104, 0.206], [0.036, 0.120] and [0.106, 0.195]—again all favoring the pairwise method. One notable difference from Figure 7 is that *MentionOrder* had the second best win-loss ratio. This shows the default order of columns in the table in many cases can be directly applied for compression, just like what search engines are currently doing. However, in some cases like Figure 2 where informative columns can also be put on the right, deeper understanding of the table semantics is necessary. This aside, the numbers look very similar to those in table synthesis, indicating that the two applications are closely related. It is interesting to see that *QueryPopularity* has many tasks rated as "about the same", but on the other tasks, raters agree that pairwise performs much better. This means *QueryPopularity* can work reasonably well in some cases but poorly in others. We checked its loss cases, such as `[champion hurdle winners]`, and found that this is generally due to the low coverage of queries on long-tail entities.

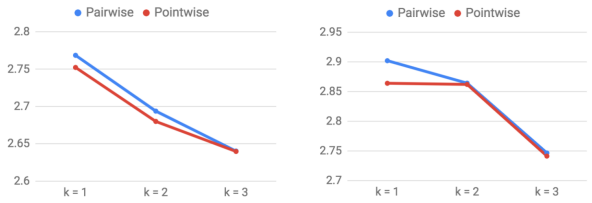

Figure 9: Average informative score (left: table synthesis, right: table compression)

7.2.3 Pairwise vs. Pointwise. A separate side-by-side experiment was conducted to compare pointwise and pairwise models, in which we saw slightly more wins by the pairwise model. We observed that often the same facts were chosen but in different orders. So we asked raters to assign an informative level to each fact using one of the four ratings: Crticial(4), Informative(3), Uninformative(2) and Unrelated(1). Average scores over top-k positions are shown in Figure 9 with standard deviation ommited for clarity. For table sythesis tasks (Fig. 9-left), slightly higher scores are observed for the pairwise at the top two positions. The two models behave similarly when we extend to top three positions, implying the facts retrieved by the two methods are close but ranked differently. For table compression tasks (Fig. 9-right), we observed higher ratings than in table synthesis, where facts were from entities' Wikipedia Infoboxes, because table columns are created to be informative even without ranking. Specifically, pairwise did a better job at the top position. Pointwise attained even higher score at second position compared to its first position. Based on this, we conclude that pointwise is not as good as pairwise in terms of recognizing the most informative facts although the overall ratings of the two are very close when $k = 2$ and $k = 3$.

7.3 Case Study

Table 3 shows several representative examples that illustrate where the pairwise model succeeds and fails. We use "loss" (second column) as the performance indicator, which is computed as the percentage of ratings for the same title for which facts from any one of the baselines were preferred over that from pairwise. The example titles were then grouped row-wise into four categories based on our understanding of factors affecting their performance.

For each of the four titles in the first group, there exist many examples (count in the fourth column) in the training data sharing the same title pattern (the third column). The model was capable of generalizing across these patterns to rank candidate facts similarly. Conversely, the learning-based approaches did not work well on titles in the second group, due to lack of similar titles in the training data. Incompleteness and noise in training data generation has also had negative impact on our model. For example, for the title `[nba championship head coaches]`, raters considered `coaching career` as an informative fact. However, it did not appear in Infoboxes or Wikiepdia page content, hence not included in the training corpus. As another example, in `[musicians from ohio]`, the fact `genre` was not identified by the pairwise model despite the existence of a `[list of musicians from chicago]` table with a related `known for` fact, as we failed to align the two facts in training data generation. Lastly, our model is not good at cases when the title requires background or context to interpret its semantics, like `[doctor who doctors]` where it is necessary to understand that the second mention of doctor in the title refers to the doctor role in the TV series *Doctor Who* instead of physician.

Table 3: Grouped titles with similar model performance.

Title	Loss (%)	Title Pattern	# Similar Titles in Training	Why Pairwise Model Succeeds (or Fails)
songs written by taylor swift	0	songs written by ...	51	Exists plenty of similar training examples so that model can predict well
mass shootings in the united states	0	mass shootings in ...	10	
hurricanes in bermuda	0	hurricanes in ...	50	
fifa world cup top goalscorers	0	... goalscorers	12	
stock exchanges in india	0.5	stock exchanges in ...	0	Lack of training data
french noble families	0.42	... noble families	0	
nba championship head coaches	0.5	... head coaches	76	Noisy training data
musicians from ohio	0.44	musicians from ...	5	
missions to mars	0.44	-	-	Understanding title semantics is hard
doctor who doctors	0.44	-	-	

8 CONCLUSIONS & FUTURE WORK

We studied the problem of contextual fact ranking—given a set of entities and a contextual description (e.g., a title), rank the candidate facts based on their informativeness to the context. We built deep learning models to accomplish this ranking task and employed weak supervision by making use of the relational tables in Wikipedia. Specifically, we developed pointwise and pairwise models with multi-tower neural architecture, and fuse textual and statistical features across the entities, the context, and the fact. We demonstrate that our approaches outperform other state-of-art algorithms, measured by the standard information retrieval metrics for ranking, including MAP, average NDCG, and average recall. We further show, via ratings collected from a production crowd evaluation platform, that our proposed models did indeed provide more informative facts in table synthesis and compression applications compared to baseline approaches.

Beyond enhancing search experience, we are actively exploring other applications that can benefit from contextual fact ranking such as programmatic Wikipedia content generation and enrichment of news content via structured data. We are also looking for ways to improve model quality, such as expanding the training data generation beyond Wikipedia tables, exploiting real user queries, identifying semantic entities from the string representations of titles and facts, leveraging correlation statistics, etc., all of which are beyond the scope of this paper, but are interesting as future work.

REFERENCES

[1] Sreeram Balakrishnan, Alon Y. Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying WebTables in Practice. In *CIDR*.

[2] Federico Bianchi, Matteo Palmonari, Marco Cremaschi, and Elisabetta Fersini. 2017. Actively learning to rank semantic associations for personalized contextual exploration of knowledge graphs. In *ESWC*. Springer, 120–135.

[3] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and Roopak Shah. 1994. Signature verification using a "siamese" time delay neural network. In *NIPS*. 737–744.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to rank using gradient descent. In *ICML*. 89–96.

[5] Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, C. Yu, D. Z. Wang, and E. Wu. 2018. Ten years of WebTables. *VLDB* 11, 12 (2018), 2140–2149.

[6] Michael J Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data integration for the relational web. *VLDB* 2, 1 (2009), 1090–1101.

[7] Michael J. Cafarella, Alon Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. 2008. WebTables: exploring the power of tables on the web. *PVLDB* 1, 1 (2008), 538–549.

[8] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*. 129–136.

[9] Varun Chandola and Vipin Kumar. 2007. Summarization–compressing data into an informative representation. *KAIS* 12, 3 (2007), 355–378.

[10] Na Chen and Viktor K Prasanna. 2012. Learning to rank complex semantic relationships. *IJSWIS* 8, 4 (2012), 1–19.

[11] Gong Cheng, Thanh Tran, and Yuzhong Qu. 2011. Relin: relatedness and informativeness-based centrality for entity summarization. In *ISWC*. 114–129.

[12] Fernando Chirigati, Jialu Liu, Flip Korn, You Will Wu, Cong Yu, and Hao Zhang. 2016. Knowledge exploration using tables on the web. *VLDB* 10, 3 (2016), 193–204.

[13] Wei Chu and S Sathiya Keerthi. 2005. New approaches to support vector ordinal regression. In *ICML*. 145–152.

[14] Gautam Das, Vagelis Hristidis, Nishant Kapoor, and S. Sudarshan. 2006. Ordering the attributes of query results. In *SIGMOD*. 395–406.

[15] Hazem Elmeleegy, Jayant Madhavan, and Alon Y. Halevy. 2011. Harvesting relational tables from lists on the web. *VLDB J.* 20, 2 (2011), 209–226.

[16] Yihan Gao and Aditya Parameswaran. 2016. Squish: Near-optimal compression for archival of relational datasets. In *SIGKDD*. 1575–1584.

[17] K. Gunaratna, K. Thirunarayan, and A. Sheth. 2015. FACES: Diversity-Aware Entity Summarization Using Incremental Hierarchical Conceptual Clustering.. In *AAAI*. 116–122.

[18] Kalpa Gunaratna, Amir Hossein Yazdavar, K. Thirunarayan, A. Sheth, and G. Cheng. [n. d.]. Relatedness-based multi-entity summarization. In *IJCAI*.

[19] Rahul Gupta, Alon Halevy, Xuezhi Wang, Steven Euijong Whang, and Fei Wu. 2014. Biperpedia: An ontology for search applications. *VLDB* 7, 7 (2014), 505–516.

[20] Rahul Gupta and Sunita Sarawagi. 2009. Answering Table Augmentation Queries from Unstructured Lists on the Web. *PVLDB* 2, 1 (2009), 289–300.

[21] Ralf Herbrich. 2000. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers* (2000), 115–132.

[22] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*. 133–142.

[23] Gregory Koch, R. Zemel, and R. Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, Vol. 2.

[24] Hang Li. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems* 94, 10 (2011), 1854–1862.

[25] Xiao Ling, Alon Y. Halevy, Fei Wu, and Cong Yu. 2013. Synthesizing Union Tables from the Web. In *IJCAI 2013*. 2677–2683.

[26] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL*. 285–294.

[27] Minh-Thang Luong, Hieu Pham, and C. Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv:1508.04025* (2015).

[28] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, C. Puhrsch, and A. Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *LREC*.

[29] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[30] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI*, Vol. 16. 2786–2792.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.

[32] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering Table Queries on the Web using Column Keywords. *PVLDB* 5, 10 (2012), 908–919.

[33] Tim Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv:1509.06664* (2015).

[34] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv:1509.00685* (2015).

[35] G. Salton and M. J McGill. 1986. *Introduction to Modern Information Retrieval*.

[36] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv:1611.01603* (2016).

[37] Amnon Shashua and Anat Levin. 2003. Ranking with large margin principle: Two approaches. In *NIPS*. 961–968.

[38] N.T. Tam, N.Q.V. Hung, M. Weidlich, and K. Aberer. 2015. Result selection and summarization for Web Table search. In *ICDE*. 231–242.

[39] Andreas Thalhammer, Nelia Lasierra, and Achim Rettinger. 2016. Linksum: using link analysis to summarize entity data. In *ICWE*. 244–261.

[40] Andreas Thalhammer and Achim Rettinger. 2014. Browsing dbpedia entities with summaries. In *ESWC*. 511–515.

[41] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. 2012. Infogather: entity augmentation and attribute discovery by holistic matching with webtables. In *SIGMOD*. 97–108.

[42] M. Yang, B. Ding, S. Chaudhuri, and K. Chakrabarti. 2014. Finding patterns in a knowledge base using keywords to compose table answers. *VLDB* 7, 14 (2014), 1809–1820.

[43] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A support vector method for optimizing average precision. In *SIGIR*. 271–278.