

Wiki2Row: The In's and Out's of row suggestion with a Large Scale Knowledge Base

Alperen Karaoglu^{*†}
Carina Negreanu
alperenkaraoglu@gmail.com
Microsoft Research Cambridge
t-caneg@microsoft.com

Dany Fabian
Microsoft Research Cambridge
daniel.fabian@microsoft.com

Shuang Chen[‡]
Harbin Institute of Technology
schen@ir.hit.edu.cn

Andrew Gordon
Microsoft Research Cambridge
adg@microsoft.com

Jack Williams
Microsoft Research Cambridge
t-jowil@microsoft.com

Chin-Yew Lin
Microsoft Research Asia
cyl@microsoft.com

ABSTRACT

Row suggestion, a generalization of set expansion, is the task of augmenting a given table of text and numbers with additional, relevant rows. A viable approach is to generate trustworthy suggestions by grounding candidates in a verifiable source and in our work we focus on knowledge bases, in particular on Wikidata. Our pipeline begins by linking existing rows and columns to entities and properties. The primary focus of this work is to improve candidate generation and ranking without requiring in-domain training or fine-tuning. Our novel contributions are to account for semantic information by using BigGraph embeddings and GPT-3 free text generation, and tabular information by differentiating between in-table properties (explicit in given columns) and out-of-table properties (implicit from the knowledge base). Measured on the WikiTables benchmark, our solution exceeds or achieves comparable performance to previous state-of-the-art systems that use small, carefully curated knowledge bases (such as DBpedia). We extend our algorithm to present the first approach to bias-aware row suggestion when table completion is not achievable, that is, when we cannot define a complete set of entities. We suggest quantitative measures to evaluate performance on this task.

KEYWORDS

knowledge base linking, natural language applications, bias and fairness, scalability, semantic knowledge, tabular data

ACM Reference Format:

Alperen Karaoglu, Carina Negreanu, Shuang Chen, Jack Williams, Dany Fabian, Andrew Gordon, and Chin-Yew Lin. 2018. Wiki2Row: The In's and

^{*}Both authors contributed equally to this research.

[†]Work conducted during AI Residency at Microsoft Research Cambridge

[‡]Work conducted during internship at Microsoft Research Asia

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/1122445.1122456>

Out's of row suggestion with a Large Scale Knowledge Base. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/1122445.1122456>

1 INTRODUCTION

Tables pervade our day-to-day, whether that is in a business, academic, or personal context. The ability to suggest new rows for a table can complete a set by example, or provide new and relevant insights. A suggestion can be linked to an external knowledge base, providing provenance for the data and instilling confidence. In this work we demonstrate state of the art performance for the task of row suggestion, using a large scale knowledge base with a loose type ontology.

Row Suggestion via a Knowledge Base. Our goal is to suggest extra rows that are based on trustworthy information. To do so we link every suggestion to an entity in a knowledge base (KB). Row suggestion using a knowledge base divides into two phases: *candidate generation* and *candidate ranking*. Candidate generation searches the knowledge base for entities that are relevant to the existing *seed rows* in the table; candidate ranking orders the generated candidates by a metric of suitability as a suggestion. This approach is adopted by prior work, including the leading system EntiTables [38, 39].

To maximise the breadth and freshness of row suggestions, a system should be able to exploit the largest and fast growing knowledge base Wikidata [32]. Existing systems such as EntiTables have only been proven against smaller knowledge bases such as DBpedia [1], and key characteristics of Wikidata mean that techniques suitable for DBpedia are not suitable for Wikidata. For example, EntiTables uses as entity types the hand-curated Wikipedia categories imported into DBpedia. In contrast to DBpedia, the ontology of types and categories in Wikidata is known to be messy and loose. As Krötzsch says [18]: “Good ontologies are hard work and don’t just ‘emerge.’” Unfortunately, any candidate generation mechanism for Wikidata that relies on types or categories will surface a large number of candidates that cannot be ranked efficiently.

We develop a new algorithm for candidate generation suitable for large scale knowledge bases such as Wikidata. We start from the BigGraph embeddings [20] of the seeds and create a candidate set from their nearest neighbors. Entity embeddings allow us to

find relevant candidates without relying on a well defined type ontology, and without surfacing vast numbers of candidates. Previous approaches exploit external data sources to obtain candidates, such as tables found on the web [38, 39]. We use free text generation via GPT-3 [6] to improve the diversity and coherence of candidate sets. Generative language models like GPT-3, which are trained with almost the entirety of web data, capture a rich knowledge of co-occurrence information that users tend to use in practice. Creative output comes at the cost of reliability so in our work we ground the generations by translating the strings to knowledge base entities, ensuring every suggestion is linked.

In order to find a suitable ranking function we start by creating a set of features based on tabular structure and KB semantics. Unlike previous work we make a distinction between in-table and out-of-table properties shared between seeds and candidates. Although we have a manageable set of candidates there is still a significant class imbalance. We re-phrase the problem as outlier detection where the good suggestions are outliers. We use the output of the detector as the ranking function.

Mitigating Bias in Row Suggestion. Some tables describe fixed sets that can be completed by row suggestion, such as *U.S. states*, while other tables describe large or innumerable sets, such as *CEOs*. As a consequence, completing a table using row suggestion might not be achievable, or it might not be desirable as it would be overwhelming for the user [7]. In the case where we return an incomplete set we need to be mindful regarding hidden biases and in this work we propose an enhanced ranking function that minimizes representational harms, in particular stereotyping. This is particularly important as users tend to over rely on embedded AI systems with initial adoption [11]. In the NLP literature there are several approaches for “debiasing” [3] and in this work we introduce a new alternative.

We propose the concept of *diversity preservation*: the idea that the distribution of a property of concern in the underlying population is preserved in the completed table. For instance, when enhancing a table about CEOs we may know that in Wikidata roughly 15% of relevant entities are women. If our system only suggests male candidates we know we need to correct our prediction and re-rank our candidates. Although we are aware that Knowledge Bases are inherently biased, we believe that maintaining a minimum diversity relative to a verifiable source and informing the user of our bias measure can increase trust and awareness.

Contributions.

- (1) We propose a new, scalable, algorithm for candidate row generation. Starting from seed entities in a table we use BigGraph embeddings to create a pool of candidates that we further diversify with candidates generated from free text generation (GPT-3).
- (2) We suggest a novel ranking function based on an outlier detector trained to account for semantic similarity and tabular structure.
- (3) We introduce the idea of diversity preservation to mitigate bias during row suggestion, suggest a quantitative metric for binary properties, and measure performance of our algorithms on a small test set.

- (4) We enhance current property linking algorithms by catering for missing properties in the KB.

Relative to prior art [38, 39] our system achieves and even exceeds performance on standard metrics under a more general setup: linking and generation in Wikidata versus DBpedia, free text generation versus Webtable alignment, and using only limited metadata (seed rows) versus rich metadata (seed rows, headers, captions, and titles)

2 RELATED WORK

For relevant work we include both row suggestion and set expansion literature as the tasks are highly related.

Regarding candidate provenance previous work has either taken a similar approach to ours and source from a Knowledge Base such as DBpedia or Freebase ([26], [27], [42], [51], [39]), or use free text ([28], [30], [25]) or use structured text such as web tables ([33], [8], [39]) or web queries ([13], [14], [34]).

In order to tackle ambiguity and semantic drift several research contributions make use of similarity between words/concepts by employing lexical or syntactic reasoning, embeddings or corpus co-occurrences ([14], [27], [30], [37], [42], [28], [38]) as well as improving the seed quality [25]. To our knowledge we are the first to use an entity embedding space, BigGraph [20], to account for relations between entities for ranking and generation.

Previous work uses various model classes for ranking, for example Machine Learning approaches ([35], [8], [19], [43], [36]), probabilistic approaches ([13], [33], [34], [30], [31], [39], [10]), graph theory approaches ([44]), and human-in-the-loop approaches ([24], [9]). Our pipeline uses Machine Learning components as we include free text generation and use the output of a unsupervised outlier detector as a ranking function. Unlike previous work we do not train or fine-tune the language model or the embeddings on tabular data. Due to the lack of a large corpus of varied data (the datasets are primarily synthetic or carefully curated) we are wary of generalization.

For relevant work regarding mitigating bias we refer to the literature presented in [3]. Previous approaches primarily debias using data manipulation (for example by augmenting data [46] or word embeddings [47]) or model adjustment [45]. Our work is most related to approaches that constrain predictions, for example [45]. We do not modify the optimization function, but re-rank candidates so that the enhanced table is not significantly more biased than the KB.

3 TABLE INTERPRETATION

A key attribute of Wiki2Row is that row suggestions are linked to data within the knowledge base. This attribute comes under the umbrella of table interpretation that can be further split in three sub-tasks as proposed in [2]: *column type identification*, *entity linking* and *property linking*. This task has been well studied and a thorough survey [40] has been recently published. In our work, table interpretation is a secondary contribution that supports row suggestion. We rely on previous methods for entity linking, in line with the work presented in [41], and improve the algorithm for property linking as this task is fundamental for our row suggestion algorithm. Section 4.1 describes our property linking algorithm.

We now formally define the task of table interpretation, starting by describing the tabular data and knowledge base (KB), and then defining the task settings.

Let T range over relational tables of the form $\{t_{1,1}, \dots, t_{m,n}\}$, where t ranges over text values. A table is a matrix of text cells of m rows and n columns. Write $T_{[i,*]}$ for row i in table T ; write $T_{[*],j}$ for column j in table T .

To make our approach generally applicable, we do not assume availability of metadata beyond the table contents, such as column headers or titles.

The knowledge base (KB) studied in our work follows RDF (Resource Description Framework) standard which consists of a terminological component (TBox) and an assertion component (ABox). Let $\mathcal{E} = \{e_1, \dots, e_{|\mathcal{E}|}\}$ be the set of all entities in the KB. The TBox defines the schema structure of the KB including type ontology $\mathcal{T}$ and properties set $\mathcal{P}$. The type ontology $\mathcal{T} = \{\tau_1, \dots, \tau_{|\mathcal{T}|}\} \subseteq \mathcal{E}$ consists of type definitions and a type hierarchy constructed with the *subclass of* relation. An entity $e \in \mathcal{E}$ can belong to one or more types of the type ontology. Specifically, we write $\text{type}(e) = \tau$ to denote an entity e belongs to a type $\tau \in \mathcal{T}$. The properties set $\mathcal{P} = \{p_1, \dots, p_{|\mathcal{P}|}\}$ defines the set of possible properties used to describe key attributes of an entity. The ABox is the instantiation of KB which is composed of a set of RDF triples $\langle s, p, o \rangle$, where s denotes a subject (an entity $e \in \mathcal{E}$), $p \in \mathcal{P}$ is a property (also known as predicate or relation) and o denotes an object (an entity e , or a data value, e.g. number, time, string etc.). We write $p(e)$ for *property lookup* that returns o when $\langle e, p, o \rangle$ exists in the KB, and $\perp$ otherwise. We implicitly assume that $p(e)$ maps to zero or one target. Although tailored for our target KB, Wikidata¹, the same notation applies to other knowledge bases like DBpedia and Freebase, etc.

Definition 1. Given table T and entity set $\mathcal{E}$, *table entity linking* aims to link entity mention m_k in a specific cell $t_{i,j}$ of table T to its referent entity in $\mathcal{E}$, or predict there is no corresponding entity in the KB, denoted $\perp$.

Definition 2. Given table T and property set $\mathcal{P}$, *property linking* is the task of associating a pair of columns, indexed by s and o , with a property $p \in \mathcal{P}$ such that property p relates $T_{[*],s}$ and $T_{[*],o}$ componentwise.

4 ALGORITHMS OF WIKI2ROW

In this section we describe the novel algorithms of our system Wiki2Row. Our contributions focus on two primary tasks: *property linking* (Section 4.1), defined earlier; and *row suggestion* (Section 4.2), which we now define.

Definition 3. Let $E = (e_{1,1}, \dots, e_{n,1})$ be the list of entities corresponding to the left most table column. *Row suggestion* generates a ranked list of entities E_{new} to be added to E . *Table completion* generates a ranked list of entities to complete the set of entities in E such that $E \cup E_{new} = E_{max}$, the maximum representative set (i.e. no further entities share the set membership).

4.1 Property linking

Our main contribution for the KB linking model regards property linking rather than entity linking. Given a set of candidate entities for the main column we try to find the most likely mapping

¹<https://www.wikidata.org/>

between relational columns and Wikidata properties. We take a similar approach to [41] with a couple of modifications - we take a different approach to linking numerical columns and we include a module to cater for missing properties.

Numeric Properties. For numeric properties we introduce the concept of *characteristic ranges*. Write $C_{p;\tau}$ for the characteristic range of property p for type τ , defined as:

$$C_{p;\tau} = \text{range}(\text{characteristic}(\mathcal{E}_{p;\tau})) \\ \text{where } \mathcal{E}_{p;\tau} = \{p(e) | \forall e \in \mathcal{E}. \text{type}(e) = \tau, p(e) \neq \perp\} \\ \text{and } p \text{ is a numeric property.}$$

The *characteristic* function removes the outliers from the set via Isolation Forest Algorithm [21]. The *range* function returns upper and lower bounds for a set of numbers.

We now define a property scoring function for a numeric cell value $t_{i,j}$ in table T , given linked main column $\mathcal{L}$ which is a column vector of entities. First, write $\mathbb{1}$ for the *indicator* function that maps true to 1 and false to 0. Write $\text{score}_{ij}(p)$ for the *exact* property score and write $\text{score}^*_{ij}(p)$ for the *approximate* property score, defined as:

$$\text{score}_{ij}(p) = \mathbb{1}(\text{conv}(p(\mathcal{L}_i) = t_{i,j})) \\ \text{score}^*_{ij}(p) = \mathbb{1}(\exists \tau. \text{conv}(t_{i,j}) \in C_{p;\tau} \wedge \text{type}(\mathcal{L}_i) = \tau)$$

As a first pass we say that a cell value (t_{ij}) is consistent with the property value of the linked entity in the same row ($\mathcal{L}_i$) if their values are equal within unit conversion (conv). If this approach does not return sufficient coverage or there is a tie we compute a secondary score that approximately matches using a characteristic range, also within unit conversion.

String Properties. Our second contribution is to introduce an algorithm that improves property linking when the property values are strings. We say that a cell value (t_{ij}) is consistent with the property value of the linked entity in the same row ($\mathcal{L}_i$) if their values are equal within fuzzy matching (similarly to [41]). If this approach does not return sufficient coverage or there is a tie and $p(\mathcal{L}_i) = \perp$, we estimate how likely it is that the property is missing. We do this by looking at the properties of the n nearest neighbors in BigGraph that share the type of e ,

$$\text{score}(p, e) = \sum_n \text{sim}(e_i, e) (\mathbb{1}(p(e_i) \neq \perp) - \mathbb{1}(p(e_i) = \perp)) \\ \text{sim}(e, e_i) = 1 - \frac{L_2(e, e_i)}{\max(\{L_2(e, e_j) | j < n\})}, \\ \text{where } \bar{f}(x) = \min(\max(0, f(x)), 1).$$

We choose BigGraph embeddings as they are highly representative of the relations within Wikidata. BigGraph assigns a feature vector (embedding) for each entity, trying to place adjacent entities close to each other in the vector space, while pushing unconnected entities apart. Therefore, entities that have a similar distribution of neighbors will end up being nearby.

We now define a property scoring function for a string cell value $t_{i,j}$ in table T , given linked main column $\mathcal{L}$. Write $\text{score}_{ij}(p)$ for the *exact* property score and write $\text{score}^*_{ij}(p)$ for the *approximate*

property score, defined as:

$$\begin{aligned} \text{score}_{ij}(p) &= \mathbb{1}(p(\mathcal{L}_i) \approx t_{i,j}) \\ \text{score}^*_{ij}(p) &= \arg \max_{p \in \text{pred}} \max_{e \in \mathcal{E}_{1i}} \text{score}(p, e) \end{aligned}$$

Complete Property Linking. We now define the complete property linking algorithm. Let T be an (m, n) -table with linked main column $\mathcal{L}$, and let j be the index of the column to be linked; function $\text{LINKPROPERTY}(T, \mathcal{L}, j)$ returns the linked property for column j or fails, and is defined in Algorithm 1.

Algorithm 1 Property linking for column index j

```

function LINKPROPERTY( $T, \mathcal{L}, j$ )
   $\mathcal{P}_e \leftarrow \{p \in \mathcal{P} \mid \exists e \in \mathcal{L}. p(e) \neq \perp\}$ 
   $s \leftarrow \{p \mapsto \sum_{i \leq m} \text{score}_{ij}(p) \mid p \in \mathcal{P}_e\}$ 
   $s_{\max} \leftarrow \{p \mapsto n \in s \mid n = \max(\text{image}(s))\}$ 
   $s^{\sim} \leftarrow \{p \mapsto n + n' \mid \forall p \mapsto n \in s_{\max}.$ 
     $n' = \sum_{i \leq m} \text{if } \text{score}_{ij}(p) = 0 \text{ then } \text{score}^*_{ij}(p) \text{ else } 0$ 
   $\}$ 
   $s^{\sim}_{\max} \leftarrow \{p \mapsto n \in s^{\sim} \mid n = \max(\text{image}(s^{\sim}))\}$ 
  if  $s_{\max} = \{p \mapsto n\} \wedge n \geq \text{threshold}$  then
    return  $p$ 
  end if
  if  $s^{\sim}_{\max} = \{p \mapsto n\} \wedge n \geq \text{threshold}$  then
    return  $p$ 
  else
    return fail "Cannot Resolve Confidently"
  end if
end function

```

4.2 Row suggestion

Algorithm 2 Row suggestion for table T with seeds $\mathcal{L}$

```

function SUGGEST( $T, \mathcal{L}$ )
   $\mathcal{T}_{\mathcal{L}} \leftarrow \text{commonTypes}(\mathcal{L})$ 
   $\mathcal{P}_{\mathcal{L}} \leftarrow \{\text{LINKPROPERTY}(T, \mathcal{L}, j) \mid j \in \text{cols}(T)\}$ 
   $\mathcal{E}_T \leftarrow \{e \in \mathcal{E} \mid \text{type}(e) \in \mathcal{T}_{\mathcal{L}} \wedge \forall p \in \mathcal{P}_{\mathcal{L}}. p(e) \neq \perp\}$ 
  // 1. Select top candidates from knowledge base.
   $C \leftarrow \text{top}_k(\text{sort}_{\text{dist}}(\{(e, \arg \min_{0 < i \leq |\mathcal{L}|} \text{dist}(\mathcal{L}_i, e)) \mid e \in \mathcal{E}_T\}))$ 
  // 2. Extend candidates using language model.
   $T' \leftarrow \text{toTable}(\text{GPT-3}(\text{toText}(T)))$ 
   $C \leftarrow C \cup \{\text{link}(t') \mid t' \in T'_{[*,1]}\}$ 
  // 3. Rank the combined candidates.
  return  $\text{rank}(C)$ 
end function

```

Algorithm 2 presents the high-level algorithm for row suggestion, which we now describe in detail. Row suggestion takes as input a table T of seeds, and the linked main column $\mathcal{L}$. We split the algorithm for row suggestion into three sub-tasks: candidate generation from the knowledge base, candidate generation using free text generation, and candidate ranking.

Candidate generation is particularly important as ranking all candidates in Wikidata has a high computational cost and at the

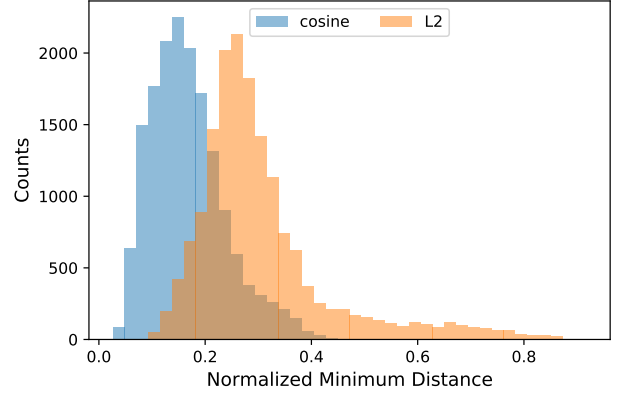


Figure 1: The characteristic distribution of ground truth row suggestions in BigGraph under scaled cosine similarity and L2 distance as generated on a subset of the validation data.

current rate of growth it cannot scale even with the most efficient ranking algorithms. We propose a combination of two approaches that complement each other to create a diverse candidate set.

Generation from KB. First we source knowledge base candidates from Wikidata using the BigGraph embedding space. The initial step is to prune the knowledge base using basic predicates, reducing the amount of computation required within the high dimension BigGraph embedding space. We retain only the entities that share an immediate type with a seed entity, and that also have every linked property from the table—we name this set $\mathcal{E}_T$. When we have a small number of seeds (our current setup), we search $\mathcal{E}_T$ for the nearest neighbours of each seed, selecting the closest k candidates using rescaled cosine distance (written dist). We can extend this approach for very large tables when we might want to keep extra seeds by checking if we can find clusters in the seed space. By using BigGraph embeddings we did not have to train a new set of embeddings on relevant tabular data to build a space that is representative for our setup (as proposed in [38]) because the embedding space is already constructed based on the relations between entities.

A key assumption in our work is that using nearest neighbors in BigGraph is meaningful for generation via similarity. In our work we explored various distances and we briefly discuss two options: L2 distance and cosine distance. In Figure 1 we can see the typical distribution of minimum L2-distances (normalised by the maximum observed distance) and minimum rescaled cosine distance between a seed and a target entity. Unlike other static embedding spaces candidates are not at the closest distance because when embedding entities (and not words) we end up clustering close variations of entities together. As Figure 1 shows, cosine distance is better than the L2 distance at grouping targets closer to the origin as in the case of standard embedding spaces. Recall that in order to keep the set manageable we require that a viable candidate has at least one type in common with the seeds and has in-table properties among its properties. Requiring full property overlap is too restrictive for

our choice of KB considering the property sparseness in Wikidata. In some cases (for example when the type is human) we need to further limit the number of generations so we constrain by demanding only in-table property overlap.

Generation from GPT-3. The second approach for candidate generation is to enhance the candidate pool with predictions obtained from GPT-3 free text generation. Input table T is converted to text and used to sample GPT-3, from which textual rows are generated and converted to table T' . For each text cell in the first column of T' we attempt to link using Elastic Search, and the best entity results are added to candidate set C . Unlike previous approaches we do not fine-tune or train on tabular data [8] and we do not require further context [19] as we can use the seed table as a prompt. We believe this is particularly important considering how homogeneous the current tabular datasets are.

Candidate Ranking. After we create a pool of candidates C , we construct a novel ranking score that takes into account the following features.

- Distance to closest seed.
- Ratio of properties not in the table in common with seeds.
- Minimum normalised Levenshtein distance between candidate entity label and seed entity labels.
- Minimum string embedding (in FastText space [4]) cosine distance between the candidate entity label and seed entity labels.
- Ratio of types shared between seeds and candidate.
- Ratio of seeds that have the candidate as a neighbour.
- Candidate generator (BigGraph, GPT-3, or both).

We rank by using the output of an outlier detector that predicts whether a candidate should be part of the suggestion set. We propose pursuing this class of algorithms as despite our best efforts class imbalance (correct versus incorrect suggestions) is significant. We find that with a few features we can have reasonable performance and efficiency by considering semantic, tabular and entity features. To account for semantic information we use Levenshtein distance and cosine distance in a static embedding space. We choose FastText as it has better performance for out of vocabulary and rare words. To better link the knowledge base characteristics to the information presented in the table we create various features between the seeds and candidates and prioritize in-table properties (as candidate filters).

We can classify outlier detectors in three classes, supervised, unsupervised and semi-supervised, and we use the toolbox proposed in [50] for detection algorithms. In this work we show best performance for each class of systems. For semi-supervised and supervised systems we first use a kNN algorithm on the training data to assign table-clusters based on our feature space (and then map the test tables to a given cluster). As a supervised system we use XGBOD [48], an ensemble system that combines unsupervised outlier mining algorithms to extract representations to improve the feature space on top of which runs a supervised classifier. For each cluster we train a VAE [16] on non-outliers (incorrect predictions) and find outliers using the reconstruction error (the hypothesis being that the higher the reconstruction error, the most likely a

candidate is to be a viable suggestions). Finally we can split unsupervised methods in several subclasses, linear models (e.g. OCSVM [29]), proximity-based (e.g. LOF[5]), probabilistic (e.g. ABOD[17]) and neural networks (e.g. MO-GAAL [22]), and considering the features we have created proximity-based and neural network models are most suitable.

5 MITIGATING BIAS IN ROW SUGGESTION

For a given binary property of entities, a table with row suggestions *preserves diversity* if the distribution of the property in the whole table equals or exceeds that in the underlying population.

Quantifying Diversity Preservation. We quantify the probability of diversity preservation by a simple application of standard Bayesian statistics. Consider a binary property p and let d be the probability of $p(e) = 1$ for any entity e in the underlying population. We consider the binary values $p(e)$ for each e in the completed table to be a sequence of Bernoulli trials with an uncertain probability parameter μ . For the row suggestions to reflect the underlying distribution d , we wish that $\mu \geq d$. We cannot know μ exactly from a few samples in the table, but we can calculate the probability that $\mu \geq d$. We say that a row suggestion *preserves diversity (of property p) with probability P* to mean that P is the probability that $\mu \geq d$.

By classical probability [15], the posterior distribution of the unknown parameter μ is $\mu \sim \text{Beta}(c + 1, n - c + 1)$, where c is the count of entities e with $p(e) = 1$ and n is the number of rows in the completed table.

For example, if $d = 15\%$ and the count of $p(e) = 1$ is exactly $c = 15$ for a completed table of 100 rows, then the probability that $\mu \geq 15\%$ is 55%. If $c = 14$, the probability that $\mu \geq 15\%$ is 44.1%. If $c = 5$, the probability that $\mu \geq 15\%$ reduces to 0.1%.

In the limit, as n tends to infinity, if the count $c = d \times n$, the probability P (that $\mu \geq d$) tends to 50%. If $d < 50\%$ then P tends to 50% from above. Hence, if $d < 50\%$, whether a row suggestion *preserves diversity of with probability 50%* is equivalent to whether $\frac{c}{n} \geq d$. Still, the advantage of defining the probability P over the simple binary test $\frac{c}{n} \geq d$ is that P amounts to a numeric score for how well diversity is preserved, and we can systematically choose an acceptance threshold other than 50%. (These calculations are based on standard properties of the *Beta* distribution.)

Choosing the number d and the acceptable probability of diversity preservation is a matter of policy. We can set d based on statistics from the knowledge base or from some other source. If our policy is for diversity preservation to be 40% probable or more, the examples $c = 15$ and $c = 14$ above are acceptable, while $c = 5$ is not.

Algorithm for Diversity Preservation. We start by identifying a finer grained type for humans, for example humans who are CEOs, and we compute the distributions over properties in Wikidata associated with systemic bias in society. Relevant properties include: sex or gender, religion, convicted of, ethnic group, unmarried partner, blood type, social classification, sexual orientation, lifestyle, and medical condition. Given a seed table, we suggest extra rows using the algorithm of Section 4 (or indeed some other algorithm). For each property p that we wish to protect, we compute the probability P that the enhanced table (seeds and suggestions) preserves

diversity with respect to p . If P is below an acceptable threshold (such as 40% mentioned above) we reject the row suggestions, and need to re-rank the suggestions by prioritizing the top candidates e with $p(e) = 1$ so as to exceed the acceptable threshold. The choice of properties to protect and the acceptable thresholds are questions of policy, outside the scope of this paper, but determined by the particular application of row suggestion.

6 EXPERIMENTAL EVALUATION

Datasets. In this work we use three datasets: the full SemTab Round 2 data with targets for Property linking (12174 tables and 43753 targets), the test and validation datasets proposed in [39] (each have 1000 tables) and a toy subset of the SemTab data that is suitable for our Bias Mitigation setup. For table completion we pruned the test data in [39] to 932 tables as the entities are either not present (or of poor quality) in Wikidata, or not well defined in BigGraph (we could not find neighbours). For bias mitigation we collected tables where humans share the same career at different granularity levels (for example politician, personality or athlete from Soviet Union). As there is no ground truth we wanted to keep the size of the dataset manageable for manual checking and annotations and limited to 35 tables with 7.86 rows and 4.78 columns on average.

The SemTab dataset is synthetically generated via SPARQL queries on Wikidata. Various data manipulations on original entity title, e.g. using aliases (Michael Jordan may be also named Air Jordan), injecting spelling errors by randomly inserting, deleting or replacing some characters etc, are introduced to make the data more challenging. In contrast, the datasets in [39] are from WikiTables which are real tables curated by humans. To our knowledge there are currently no large scale, high quality annotated real tables suitable for this problem.

Dataset augmentation. In order to improve our experimental results we have to augment the data in a couple of ways. As the WikiTables dataset was curated in 2014 as part of TabEL [2], the set of ground truth suggestions is incomplete, for example the table about US presidents does not include Donald Trump. Unfortunately extending the ground truth tables can be noisy so we choose a more conservative approach: we create a set of rough table extensions and we check if any of the extra entities are among the highly ranked candidates in our model. If they are we remove them when reporting results. This problem affects at least 58 out of 937 tables.

A second issue with the curated test set is that the main assumption in the task definition does not necessarily hold for this dataset. A significant number of tables in the test set are not strictly relational (as defined in [40]). A common pattern (between 8.7%-19.8%) can be seen in Figure 2. From the task definition (Definition 3) there is an implicit assumption that the first column is the main column, but in this case the relational columns do not describe the sports event (they are properties of the athlete). We can argue that tables like this are actually two tables joined on a common column - in this case the Player. With our property linking algorithm we can identify the join column in the table as defined by maximum number of properties identified. We then restrict the table to first column and join column to run our algorithm to find suggestions for the first (main) column. Conservatively we estimate that at

Year	Player	Team represented	Position played
1994	Emmitt Smith	Dallas Cowboys	Running back
1995	Barry Sanders	Detroit Lions	Running back
1996	Brett Favre	Green Bay Packers	Quarterback

Figure 2: Example of non-relational table in WikiTables.

least 83 tables are joined tables and not strictly relational. As far as evaluation is concerned, we could avoid this debate by reporting full row predictions. Unfortunately as prior art does not currently support full row suggestions we leave it for future work to make sure we report fairly by extending the baselines.

Experimental setup. For rapid development we created a pipeline to download Wikidata dumps and load them into a Spark distributed cluster in DataBricks that accepts between 2-11 workers (Standard DS4 v2 with 28GB Memory, 8 Cores, 1.5 DBU). The full pipeline can efficiently run in Databricks and the runtime upperbound for the full test set is 14 minutes for entity-property linking, 47 minutes for candidate generation, 11 minutes for feature generation and 6 minutes for ranking. We believe the runtime can be significantly improved as we have not focused on optimizing the pipeline. For reproducibility we use the pre-trained BigGraph embeddings for Wikidata² (2019 version - which includes 55 million Q-number entities) and the 300 dimensional, 1 million FastText Embeddings³ trained with subword information on Wikipedia 2017, UMBC web-base corpus and statmt.org news dataset.

Particulars to reproduce Candidate Generation Experiments. In order to generate GPT-3 candidates we call the API 10 times on average (as we are limited by the number of tokens) and use temperature 0.8 (we trade stability for diversity). We generate 100 raw candidates per table out of which, on average, only 56 can be linked to a Wikidata entity and generate 386 candidates. Thus we generate significantly less candidates than standard approaches and just 20 times more than the average table size (21 rows).

For BigGraph generation when using the type constraint we reduce the search space by a factor of 50 at the cost of limiting maximum average recall to 93.6%. Introducing the in-table property constraint reduces the space further by a factor of 2 without affecting maximum average recall.

Particulars to reproduce Ranking Experiments. In our work we have experimented with all the available detectors in [50]. In general we found that ensembles give a performance boost of 5%-7% and in particular XGBOD [48] and LSCP [49] perform best. We set the contamination level between 1%-6% and as we use the output just for ranking (and not classifying), we are looking for methods that are tolerant to changes in the contamination hyperparameter. For proximity based methods we are also interested in the number of neighbours hyperparameter and for all the methods we have tried the results did not change drastically by changing this hyperparameter.

²https://torchbiggraph.readthedocs.io/en/latest/pretrained_embeddings.html

³<https://fasttext.cc/docs/en/english-vectors.html>

For unsupervised methods, MO-GAAL[22] outperformed other neural network approaches, but we found that it was not particularly stable for our set of features (level of contamination and number of subgenerators had a fair impact). Furthermore training and inference is significantly more expensive than other methods without a substantial improvement (as can be seen in Table 4). Proximity based outlier detectors proved to be very effective for the set of features we created and HBOS [12] had a slightly better performance than other approaches.

For supervised and semi-supervised methods we use a subset of the validation set proposed in [39] that we can confidently link to Wikidata (912/1000 tables).

Property linking results. We start by presenting an ablation study to show our system's Property Linking performance. We chose SemTab data as we can report results that are externally verified as part of this year's competition. As the competition is ongoing we are not aware of the architectures used by the other leading models, *MTab4Wikidata* and *TeamDagobah*, thus we are just reporting relative performance. Our method achieves comparable performance with the top1 team (*MTab4Wikidata*) and we notice that our missing properties technique gives us a boost of 3.2% as can be seen in Table 1. For this dataset the gain is significant as basic methods can achieve very high scores - our simple baseline where we do direct matching achieves 95%. On this dataset simple numerical linking was sufficient and we had no gain from using our more advanced approach. We report results only for Round 2 as we could not run an ablation study on other rounds (the number of submissions is relevant for the challenge). As the data is generated with the same processing we do not think there will be a significant change.

Property linking		
	F1	PRECISION
MTab4Wikidata	99.7	99.7
TeamDagobah	99.2	99.4
Direct-match	95.0	95.0
Wiki2Row	96.1	97.7
Wiki2Row + missing property	99.3	99.4

Table 1: Property linking results on SemTab2020 Round 2 data.

Candidate generation results. An important component of our work is candidate generation. Unlike previous approaches we generate candidates from Wikidata and unfortunately the prior generation algorithms are not suitable for this KB. Categories play an important role in previous approaches [39], but for the vast majority of Wikidata entities categories are not provided and thus we cannot fairly compare approaches by a direct implementation of prior art.

We provide a first comparison in Table 2 by looking at the performance of our approach versus the current state of the art in candidate generation. The number of candidates quoted is the average total number of generations per table, for Entitables = EntitableKB + WebTables (1505) and for Wiki2Row = BigGraph + GPT3 (386), where in a given budget we prioritise WebTables/GPT3 over KB

Average Recall per number of candidates generated				
METHOD	ANCHOR	1000	10000	100000
EntiTables KB	DBpedia	63.8	78.9	88.2
EntiTables	+ tables	78.4	93.9	96.6
BigGraph type	Wikidata	63.5/71.2	80.6/87.3	89.5/93
+ GPT-3	+ free text	78.1/82.9	88.4/92.9	94.8/96.1
+ in-table prop	Wikidata	64.5/71.9	81.2/87.6	89.8/93.2
+ GPT-3	+ free text	80.4/85.1	90.2/94	95.7/96.2

Table 2: Average recall for generations from top 3 seeds. Results from testing with 932 tables from Wikitables [2], as used to benchmark Entitables [39], augmented for use with current Wikidata as described in Section 6. The top two rows of results describe variations of EntiTables; the bottom four rows of results describe variations of Wiki2Row. For Wiki2Row, we provide the results for generation when the distance is L2 (left side) or cosine (right side). High recall on 1000 candidates is perhaps most important: the Wiki2Row recall of 85.1% is best, and means that on average about 16 rows are included in the candidate set, as the test tables have on average 18.4 rows (apart from seeds) to be predicted. The Wiki2Row advantage diminishes on 10,000 and 100,000 candidates, but the additional orders of magnitude (10X and 100X) of computation yield only a few extra rows in absolute terms.

Stability study		
	3 SEEDS	4 SEEDS
EntiTable KB	73.1 +/- 0.1	74.9 +/- 0.08
EntiTables	93.6 +/- 0.08	94.2 +/- 0.07
BigGraph	75.9 +/- 0.09	77.6 +/- 0.08
GPT-3	44.8 +/- 2.6	46.3 +/- 3.2
BigGraph + GPT-3	93.8 +/- 0.6	94.1 +/- 0.8

Table 3: Average recall performance with error-bars by choosing every 3-seed/4-seed combination in top 5 rows. The bottom three rows describe variations of Wiki2Row. We limit to an average of 5000/6000 candidates per table to recover the EntiTables generations from Table2Vec [38]. Results from testing on same 932 tables as described in Table 2.

generation. For the baseline we reimplemented the work presented in [39] and we can recover their performance (the results from [38] can be seen in Table 3). For our work we provide an ablation study to show the effect of each refinement. When the number of candidates is relatively small we significantly improve performance, but lose our advantage as we increase number of candidates. We hypothesize this is a consequence of the fact that Wikidata is significantly larger than DBpedia. We observe similar results for 4 or 5 seeds.

Previous approaches only report results by generating suggestions when they consider the top-k rows as seeds. We argue that unless we treat rows as an ordered collection the generation should be order agnostic. We report average recall for 3 or 4 seeds in Table 3 for our best performing model where we prune by type and in-table properties and generate 5000 candidates.

Mean average precision for table completion					
METHOD	TYPE	WIKI2ROW		ENTITABLES	
		3	4	3	4
Table2Vec	Supervised	53.4	53.6	59.9	60.2
XGBOD	Supervised	68.2	68.4	59.8	60.4
VAE	Semi-supervised	60.7	61.2	58.6	58.4
MO-GAAL	Unsupervised	61.8	61.6	59.7	59.6
LSCP + HBOS	Unsupervised	60.2	60.6	54.2	54.9

Table 4: Mean average precision (MAP) for table completion by taking the top 3/4 rows as seeds. We report performance on the whole ground truth set and inherit a handicap from candidate generation (between 6% - 7%). Results from testing on same 932 tables as described in Table 2.

We have included only 3 and 4 seeds as previous models have peak performance in this case and GPT-3’s prompt requires a fair amount of information. As we can see the error margin is relatively small for BigGraph and prior art, but significant for GPT-3. In our study we observed that GPT-3 performs very well in a scenario such as Figure 2 when there is an apparent order. Our best hypothesis is that when the seeds have a certain pattern GPT-3 confidently reproduces the pattern, which is consistent with previous studies [6].

Thus we have shown that despite generating from a larger, less well structured KB we manage to achieve and even exceed the candidate coverage from prior art on DBpedia.

Ranking results. For the task of table completion we test the capabilities of the ranking function by using the candidates previously generated. When comparing the generation results between the two methods we find that in the case where we restrict candidate numbers to 5000 we have only 1121 candidates in common (on average). As we want to test the ranking function we run our outlier detection algorithms over both candidate sets in Table 4. For the Table2Vec ranking function [38] we directly quote the authors’ results and our own implementation of their method for our candidate set. We do not compare against [8] as they are only considering the case where we have 0 or 1 seeds which is not applicable to our scenario.

As Table 4 shows, XGBOD has the best performance on this test set, but non-supervised methods also perform competitively. This result is particularly important as it indicates that our set of features is well constructed. Unfortunately the good results do not necessarily translate for the candidate set generated in EntiTables. We hypothesize this is caused by the way we constructed our feature space to take advantage of BigGraph’s structure which does not translate as well for DBpedia entities mapped in the Wikidata space. Similarly Table2Vec’s method did not work as well for our candidates as the suggestions generated by BigGraph are semantically closer to the ground truth.

Although scoring worst, LSCP+HBOS can be a good choice considering it is significantly more efficient than other approaches and it is fully unsupervised. Thus we hope it would generalize well and be easily supported for online pipelines.

Mitigating Bias. We focus on 35 tables of humans, and consider preservation of diversity with respect to the fraction of women.

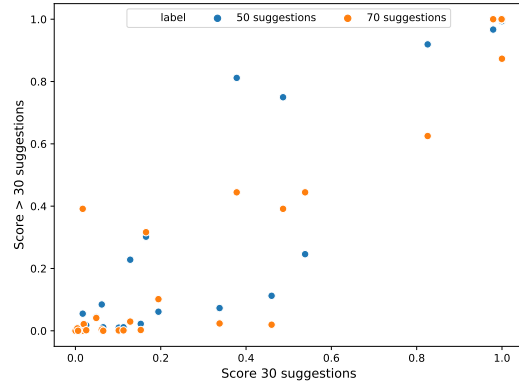


Figure 3: Bias scores as a function of number of suggestions. We observe that the level of bias does not improve with more suggestions as we do not observe many samples above the diagonal. Results from subset of 35 tables from the test set described in Table 1.

We considered several occupational categories for humans, each of which has a different fraction of women in our underlying knowledge base. Some of these fractions are: singer (45.8%), musician (14.7%), politician (12.3%), writer (26.9%), personality (23%).

For each of the 35 tables, we took the diversity d to be preserved as the fraction of women in the occupation of the table, suggested a set of rows, and computed the probability P that the completed table preserved diversity. We computed this score P for different numbers of suggested rows: 30, 50, and 70 suggestions.

Figure 3 plots each score for 50 and 70 suggestions, versus the score for the baseline of 30 suggestions. The plot shows we obtain baseline scores across the whole unit interval, with a concentration below 20%. We do not see a particular improvement with 50 or 70 suggestions, unfortunately, as one would hope the probability would rise as more observations are available. A simple explanation is that diversity preservation becomes stricter with the number of samples as the uncertainty around parameter μ decreases. Thus having a re-ranking strategy is important especially in large sets.

Figure 4 shows the results for the most common categories in the dataset. There appear to be significant differences in the ability of some parts of our system to preserve diversity, depending on the occupational category. For politicians we perform particularly poorly and this is consistent with previous bias experiments for language models where certain jobs (such as engineer [23]) are significantly under-represented. Other categories, such as singer, have a good performance despite the high fraction of women (45.8%) in the Wikidata population, but it might be an artifact of the small test set. We leave as future work to identify which parts of our system or indeed the knowledge base can explain the differences in scores between occupational categories.

To summarize, diversity preservation is important for row suggestion, and we have preliminary results. Our test set is small and we cannot yet draw meaningful conclusions from the data. Still, we present results to exemplify the type of analysis we could conduct

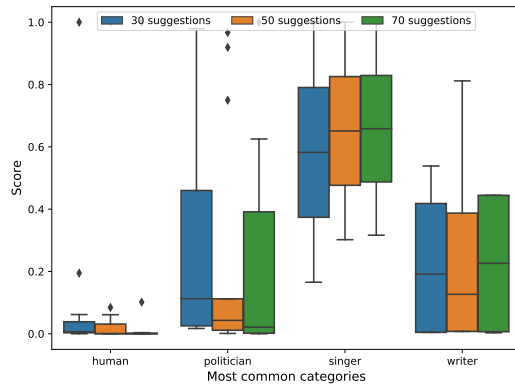


Figure 4: Bias scores for the most common categories in the dataset, when suggesting 30, 50, or 70 suggestions per table. The Score axis is the probability that a suggestion preserves diversity (with respect to women), that is, the probability that $\mu \geq d$ where d is fraction of women in the underlying Wikidata population and μ is the inferred parameter for women in the table completed by the generated rows. The fraction d is 23% for human, 12.3% for politician, 45.8% for singer, and 26.9% for writer. Results from subset of 35 tables from the test set described in Table 1.

at scale. It is important too to preserve additional factors, including for example race.

7 CONCLUSIONS AND FUTURE WORK

The focus of our work is to present a new solution for row suggestion that improves on previous state-of-the-art approaches. During our research we became aware that systems currently only consider small, well-curated KBs and thus we decided to build a system that is tailored for Wikidata, the largest and fast growing knowledge base (at the time of writing having reached one hundred thousand entities).

Although our results are encouraging we can further improve on a few fronts. For instance, an approach such as the one presented in [8] uses a deep learning model for contextualised embeddings. In our future work we would like to marry such an approach (a language model such as GPT-3) with our knowledge base model. This way we would not have to engineer a feature set and hopefully it would generalize better cross KBs. This approach would also let us improve and integrate our entity-property linking system.

In order to proceed we require a better dataset and we will focus on collecting and annotating user data. This way we can be confident that any deep learning approach (that is for example trained on WikiTables and tested on the new dataset) generalises well, and that the results presented in this work are representative.

REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The semantic web*. Springer, 722–735.

[2] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: Entity Linking in Web Tables. In *The Semantic Web - ISWC 2015*.

[3] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (Technology) is Power: A Critical Survey of “Bias” in NLP. *arXiv preprint arXiv:2005.14050* (2020).

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* (2017).

[5] M. Breunig, H. Kriegel, R. Ng, and J. Sander. 2000. LOF: identifying density-based local outliers. In *SIGMOD '00*.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[7] Jeroen De Knijf, Anthony Liekens, and Bart Goethals. 2011. “Tell Me More”: Finding Related Items from User Provided Feedback. In *International Conference on Discovery Science*.

[8] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *arXiv:2006.14806*

[9] Julian Eberius, Maik Thiele, Katrin Braunschweig, and Wolfgang Lehner. 2015. Top-k Entity Augmentation Using Consistent Set Covering. Association for Computing Machinery.

[10] Zoubin Ghahramani and Katherine A Heller. 2006. Bayesian sets. In *Advances in neural information processing systems*.

[11] Ella Glikson and Anita Williams Woolley. 2020. Human trust in Artificial Intelligence: Review of empirical research. *Academy of Management Annals* ja (2020).

[12] Markus Goldstein and A. Dengel. 2012. Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm.

[13] Rahul Gupta and Sunita Sarawagi. 2009. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment* (2009).

[14] Yeye He and Dong Xin. 2011. Seisa: set expansion by iterative similarity aggregation. In *Proceedings of the 20th international conference on World wide web*.

[15] E. T. Jaynes. 2003. *Probability Theory: The Logic of Science*. Cambridge University Press.

[16] Diederik P. Kingma and M. Welling. 2014. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114 (2014).

[17] H. Kriegel, M. Schubert, and A. Zimek. 2008. Angle-based outlier detection in high-dimensional data. In *KDD*.

[18] Markus Krötzsch. 2018. Ontological modelling in Wikidata. Talk at *Workshop on Ontology Design and Patterns 2018*, held at *ISWC 2018*. Available at https://iccl.inf.tu-dresden.de/w/images/e/ed/Ontology_modelling_Wikidata_Markus_Kroetzsch_WOP2018.pdf.

[19] Guy Kushilevitz, Shaul Markovitch, and Yoav Goldberg. 2020. A Two-Stage Masked LM Method for Term Set Expansion. *arXiv preprint arXiv:2005.01063* (2020).

[20] A. Lerer, Ledell Yu Wu, Jiajun Shen, Timothée Lacroix, Luca Wehrstedt, A. Bose, and Alexander Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. *ArXiv* (2019).

[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-Based Anomaly Detection. (2012).

[22] Yezheng Liu, Zhe Li, Chong Zhou, Yuanchun Jiang, Jianshan Sun, M. Wang, and X. He. 2020. Generative Adversarial Active Learning for Unsupervised Outlier Detection. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[23] Moin Nadeem, Anna Bethke, and Siva Reddy. 2020. StereoSet: Measuring stereotypical bias in pretrained language models.

[24] F. Qi, X. Wu, and N. Wang. 2017. Building Top-k Consistent Results for Web Table Augmentation. In *2017 14th Web Information Systems and Applications Conference (WISA)*.

[25] Zhenyu Qi, Kang Liu, and Jun Zhao. 2012. Choosing Better Seeds for Entity Set Expansion by Leveraging Wikipedia Semantic Knowledge. In *Pattern Recognition - Chinese Conference, CCPR 2012, Beijing, China, September 24-26, 2012. Proceedings*.

[26] Dominique Ritze, Oliver Lehmberg, and C. Bizer. 2015. Matching HTML Tables to DBpedia. In *WIMS '15*.

[27] Xin Rong, Zhe Chen, Qiaozhu Mei, and Eytan Adar. 2016. Egocet: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In *Proceedings of the Ninth ACM international conference on Web search and data mining*.

[28] Luis Sarmiento, Valentin Jijkuon, Maarten De Rijke, and Eugenio Oliveira. 2007. “More like these” growing entity classes from seeds. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*.

[29] B. Schölkopf, John C. Platt, J. Shawe-Taylor, Alex Smola, and R. Williamson. 2001. Estimating the Support of a High-Dimensional Distribution. *Neural Computation* (2001).

[30] Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

- [31] Isabel Valera and Zoubin Ghahramani. 2014. General Table Completion using a Bayesian Nonparametric Model. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*.
- [32] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.
- [33] Chi Wang, Kaushik Chakrabarti, Yeye He, Kris Ganjam, Zhimin Chen, and Philip A Bernstein. 2015. Concept expansion using web tables. In *Proceedings of the 24th International Conference on World Wide Web*.
- [34] Zhijun Xiao, Cuiping Li, and Hong Chen. 2020. PatternRank+NN: A Ranking Framework Bringing User Behaviors into Entity Set Expansion from Web Search Queries. *ACM Trans. Web* (2020).
- [35] Lingyong Yan, Xianpei Han, Ben He, and Le Sun. 2020. End-to-End Bootstrapping Neural Network for Entity Set Expansion.. In *AAAI*.
- [36] Lingyong Yan, Xianpei Han, Le Sun, and Ben He. 2019. Learning to Bootstrap for Entity Set Expansion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
- [37] Puxuan Yu, Zhiqi Huang, Razieh Rahimi, and James Allan. 2019. Corpus-based Set Expansion with Lexical Features and Distributed Representations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [38] Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2Vec: Neural Word and Entity Embeddings for Table Population and Retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [39] Shuo Zhang and Krisztian Balog. 2017. Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [40] Shuo Zhang and Krisztian Balog. 2020. Web Table Extraction, Retrieval, and Augmentation: A Survey. *ACM Trans. Intell. Syst. Technol.* (2020).
- [41] Shuo Zhang, Edgar Meij, Krisztian Balog, and Ridho Reinanda. 2020. Novel Entity Discovery from Web Tables. In *Proceedings of The Web Conference 2020*.
- [42] Xiangling Zhang, Yueguo Chen, Jun Chen, Xiaoyong Du, Ke Wang, and Ji-Rong Wen. 2017. Entity Set Expansion via Knowledge Graphs. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*.
- [43] Yunyi Zhang, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2020. Empower Entity Set Expansion via Language Model Probing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*.
- [44] Zhenzhong Zhang, Le Sun, and Xianpei Han. 2016. A Joint Model for Entity Set Expansion and Attribute Extraction from Web Search Queries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*.
- [45] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457* (2017).
- [46] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. *arXiv preprint arXiv:1804.06876* (2018).
- [47] Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning gender-neutral word embeddings. *arXiv preprint arXiv:1809.01496* (2018).
- [48] Yue Zhao and M. K. Hryniewicki. 2018. XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning. *2018 International Joint Conference on Neural Networks (IJCNN)* (2018).
- [49] Yue Zhao, M. K. Hryniewicki, Zain Nasrullah, and Z. Li. 2019. LSCP: Locally Selective Combination in Parallel Outlier Ensembles. In *SDM*.
- [50] Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research* (2019).
- [51] Yuyan Zheng, Chuan Shi, Xiaohuan Cao, Xiaoli Li, and Bin Wu. 2017. Entity Set Expansion with Meta Path in Knowledge Graph. In *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I*.