MAGIC: Mining an Augmented Graph using INK, starting from a CSV

Bram Steenwinckel[1][0000−0002−3488−2334], Filip De Turck[1][0000−0003−4824−1199], and Femke Ongenae[1][0000−0003−2529−5477]

IDLab, Ghent University-imec, Technologiepark 126, 9050, Gent, Belgium
bram.steenwinckel@ugent.be

Abstract. A large portion of structured data does not yet reap the benefits of the Semantic Web. Therefore, The "Tabular Data to Knowledge Graph Matching" competition at ISWC tries to bridge this gap by evaluating and promoting the creation of such semantic annotations tools. Besides annotating data semantically, the system should also be able to further augment the datasets based on the provided annotations. In this paper, we propose a system that is capable of both annotating and augmenting a dataset by using the interpretable embedding technique INK. The "Tabular Data to Knowledge Graph Matching" competition was used to evaluate the proposed annotation capabilities of our proposed system.

Keywords: Tabular Data · Semantic Annotation · Node Embedding · Data Augmentation · Entity Recognition · Type Recognition · Property Recognition.

1 Introduction & Challenge Description

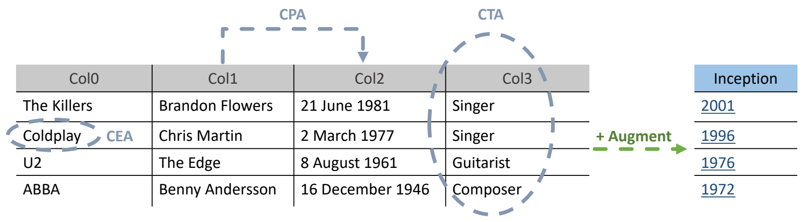
A large portion of existing or newly produced structured data is not semantically annotated. Solutions exist to enrich raw structured data semantically based on the textual descriptions within these structured files [15,12,7,1,4,3].

The "Tabular Data to Knowledge Graph Matching" competition that has been hosted for several years at the International Semantic Web Conference (ISWC) [9,8] stimulates the creation and optimization of these semantic annotation systems. These systems try to extract semantic annotations from a given Knowledge Graph (KG): a collection of interlinked descriptions of entities, both human and machine-readable. DBpedia [2] and Wikidata [17] are two such KGs.

Given a Comma-Separated Values (CSV) file, most of the time, three different challenges have to be tackled in the "Tabular Data to Knowledge Graph Matching" competition, as visualized in Figure 1: (i) the automated assignment of a

DBpedia or Wikidata column type (Column-Type Annotation (CTA)), (ii) a DBpedia or Wikidata entity has to be assigned to the different cells (Cell-Entity Annotation (CEA)), and (iii) relations between different columns have to be inferred, when possible (Columns-Property Annotation (CPA)). In our example Figure 1, the CEA task would have to assign the *dbr:Coldplay* or WIKI:Q45188 to text description of COLDPLAY in Col0. For the CTA task, Col3 must be assigned to *dbo:PersonFunction* or *wiki:Q66715801.* The relation between Col1 and Col2 for the CPA task must be defined as *dbp:birthDate* or *wiki:P569* in a semantic annotation system. The competition consists of different rounds, with structured files from different domains. No ground truth labels are provided upfront, which means only unsupervised learning methods can be used.

Most of the existing solutions are based on external lookup methods and infer the column and property types afterwards by either clustering the entities based on an uninterpretable embedded vector or by using entity-specific scoring procedures [12,7]. These annotators are denoted as semantic annotation platforms and provide a link to existing KGs but are currently not able to augment an annotated, structured file with new information available in those KGs. When for example, semantic annotations are provided for all entities within our example in Figure 1, the current techniques are not able to augment this table with additional linked data. Here, as an example, the inception dates for each entity in Col0 could be added to further enrich the current structured file. Semantic data augmentation tools, which can add new information based on tabular data, exist today [6]. Semantic data augmentation tools which can both define semantic annotations and also provide and new information based on semantic annotations are of high interest nowadays as more and more machine learning (ML) techniques try to use a graphical representation as input and tasks such as node classification are becoming popular [14]. The current nature of the existing semantic annotators, such as the uninterpretable embedding characteristics makes them not directly suitable for such an augmentation preprocessing step.



Col0	Col1	Col2	Col3
The Killers	Brandon Flowers	21 June 1981	Singer
Coldplay	Chris Martin	2 March 1977	Singer
U2	The Edge	8 August 1961	Guitarist
ABBA	Benny Andersson	16 December 1946	Composer

Inception
2001
1996
1976
1972

Fig. 1. The three different subchallenges of the competition are explained in a simple example. In the CEA task, a system tries to define each individual cell entity. Besides these 3 sub-challenges, a more general data augmentation task can also be interesting to perform.

Therefore, in this paper, we describe MAGIC: a data mining tool to augment a structured file with data residing in a KG. In order to mine this augmented data, MAGIC will make use of INK [16], a fully interpretable embedding technique capturing all information from the neighbourhood of a node within our graph. Based on that interpretable embedding, the MAGIC platform can easily perform the CEA, CTA and CPA tasks all at once compared to its competitors and can provide additional linked data defined in those interpretable embeddings. By performing all tasks at once and storing the obtained INK embedding, the number of external calls to entity lookup services can be reduced. The fact that INK can also query KGs stored locally reduces the calls needed to external SPARQL endpoints even further.

We have organised the rest of our paper as follows: In Section 2, we give a general introduction to our interpretable embedding technique INK. Next, Section 3 describes how INK is being used in our semantic annotation tool MAGIC. Implementation details are provided in Section 4. Section 5 shows the results obtained during the "Tabular Data to Knowledge Graph Matching" competition. To illustrate the additional benefits of both the interpretable embedding technique INK in combination with the MAGIC platform, we demonstrate how an existing structured file can be augmented with additional information within a KG in Section 6. Finally, Section 7 concludes this paper and shows some additional future research directions.

2 INK: Instance Neighbouring by using Knowledge

Various techniques exist to transform information residing in KGs to a more appropriate format for an ML model, such as RDF2Vec [13]. INK is such a technique that builds node embeddings by transforming the neighbourhood of the node within a KG into an interpretable structured format. An example of the INK node extraction approach is provided in Figure 2. Here, the goal is to build an INK embedding for the Coldplay node. INK will first query the neighbourhood until a predefined depth. If we define the depth parameter K to be one, only the nodes within the direct neighbours (visualized in grey) will be visited. INK iteratively extracts neighbourhood information and store these neighbourhoods efficiently (visualized in Figure 2 on the right). The predicates in the neighbourhood of depth one are concatenated with their corresponding objects as values (concatenation here performed using the special character §). To add the neighbourhoods of depths > 1 (shown by the orange and green coloured edges within our example KG), INK concatenates all the relations on a path from the root node to the object node together, without providing detailed information about all intermediate nodes on that path. This intermediate information is still available due to the extraction at the lower neighbourhood's depths.
All this extracted information is stored in a dictionary value with as key the root node. When extractions are provided for multiple of these root nodes, a two-dimensional matrix or data frame can be constructed defining which of these extracted (relation, object) pairs occur in the root nodes. Accompanied with descriptive labels for each of such a root node, this two-dimensional rep-

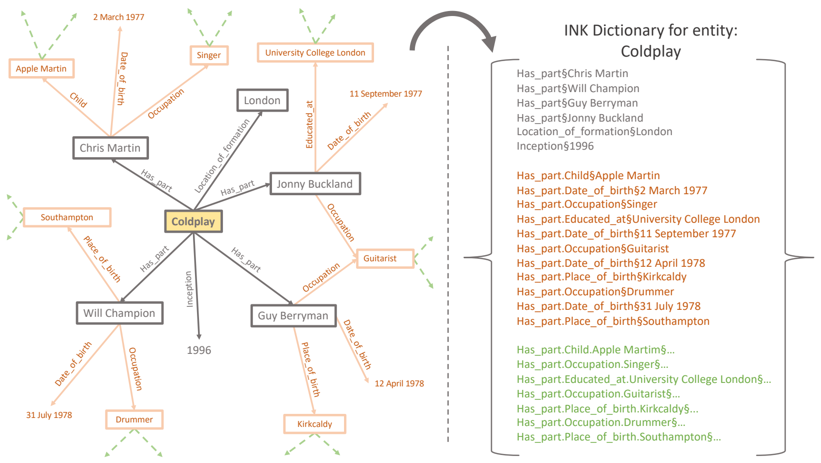
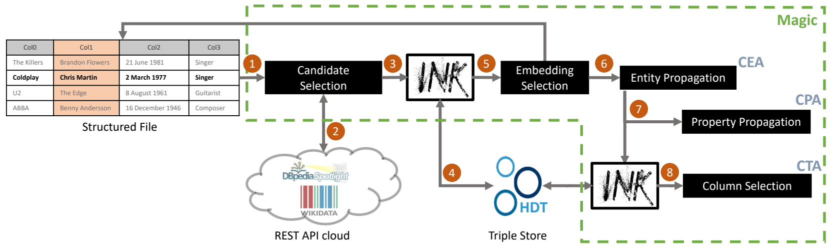

2 March 1977
Apple Martin
Date_of_birth
Singer
University College London
Child
Occupation
London
11 September 1977
Educated_at
Date_of_birth
Chris Martin
Location_of_formation
Has_part
Has_part
Jonny Buckland
Southampton
Coldplay
Occupation
Place_of_birth
Has_part
Has_part
Guitarist
Inception
Occupation
Will Champion
Guy Berryman
Date_of_birth
1996
Date_of_birth
Occupation
Place_of_birth
12 April 1978
31 July 1978
Drummer
Kirkcaldy
INK Dictionary for entity:
Coldplay
Has_part§Chris Martin
Has_part§Will Champion
Has_part§Guy Berryman
Has_part§Jonny Buckland
Location_of_formation§London
Inception§1996
Has_part.Child§Apple Martin
Has_part.Date_of_birth§2 March 1977
Has_part.Occupation§Singer
Has_part.Educated_at§University College London
Has_part.Date_of_birth§11 September 1977
Has_part.Occupation§Guitarist
Has_part.Date_of_birth§12 April 1978
Has_part.Place_of_birth§Kirkcaldy
Has_part.Occupation§Drummer
Has_part.Date_of_birth§31 July 1978
Has_part.Place_of_birth§Southampton
Has_part.Child.Apple Martim§...
Has_part.Occupation.Singer§...
Has_part.Educated_at.University College London§...
Has_part.Occupation.Guitarist§...
Has_part.Place_of_birth.Kirkcaldy§...
Has_part.Occupation.Drummer§...
Has_part.Place_of_birth.Southampton§...

Fig. 2. Example of the INK dictionary representation (right), extracted from several neighbourhoods of the Coldplay node within a larger KG (left).

resentation can be used to perform some more general machine learning tasks, such as node classification. More information about INK and the performance of this technique on several node classification tasks can be found in previous work [16].

3 System Description

The interpretable embedding approach as discussed in Section 2 is used in a system called MAGIC to annotate structured files. An overview of the MAGIC system is provided in Figure 3. In general, three modules can be derived: A pre-processing module to select the appropriate annotation candidates, a processing module based on INK to get the interpretable embedding and select the best matches and at last, a post-processing module to offload the obtained information to the corresponding competition tasks. The rest of the section describes, in detail, the eight different steps to process the structured input file.



Col0	Col1	Col2	Col3
The Killers	Brandon Flowers	21 June 1981	Singer
Coldplay	**Chris Martin**	**2 March 1977**	**Singer**
U2	The Edge	8 August 1961	Guitarist
ABBA	Benny Andersson	16 December 1946	Composer

Fig. 3. Overall MAGIC architecture to define semantic annotations.

3.1 Defining the Major Column in a Structured File

In most cases, one single column holds multiple relationships to other columns within a table, for which annotation should be provided. This column can be seen as the major column. In our example of Figure 1, Col1 is our major column as both the information of Col0, Col2 and Col3 are derived from the information in this column. In many cases, this major column is known upfront and can be provided as additional input. During the “Tabular Data to Knowledge Graph Matching” competition, the major column was derived from the CPA target file when available. If such a target file was not provided, the MAGIC approach was rerun for every available column.

3.2 External Entity Lookup

Each cell description of only the major column is then provided to the more general MAGIC annotator. A pool of possible annotation candidates is generated from these cell descriptions. For the competition tasks which requires DBpedia annotations, the DBpedia Spotlight [11] service was used to generate possible candidates. When the annotations had to be linked to Wikidata entries, requests were sent to the Wikidata API[1] to search for entities using labels and aliases (wbsearchentities).

3.3 Candidate Selection

The external entity lookup services are not restricted, which means all possible candidates matching the provided cell description are returned. Based on multiple predefined characteristics, candidates can be prefiltered. The candidate selection step reduces, in this perspective, the number of matches obtained from the external API’s. In this first version of the MAGIC system, we neglected any intelligent filtering of candidates.

3.4 Generating embeddings with HDT backend

For the selected candidate annotations, an INK embedding of depth 2 was generated as discussed in Section 2. To limit the number of external SPARQL requests, INK extracted the neighbourhood information for these embeddings from a local compact data structure called (Header, Dictionary, Triples) or HDT [5]. HDT is a binary serialization format for RDF that keeps big datasets compressed to save space while maintaining search and browse operations without prior decompression. This makes it an ideal format for storing and sharing RDF datasets on the Web. An HDT-encoded dataset is composed of three logical components (Header, Dictionary, and Triples), carefully designed to address RDF peculiarities.

[1] https://www.wikidata.org/w/api.php

- Header: The Header holds metadata describing an HDT semantic dataset using plain RDF. It acts as an entry point and shows the key properties of the content even before retrieving the whole dataset.
- Dictionary: The Dictionary is a catalogue comprising all the different terms used in the dataset, such as URIs, literals and blank nodes. A unique identifier (ID) is assigned to each term, enabling triples to be represented as tuples of three IDs, which reference their respective subject/predicate/object term from the dictionary.
- Triples: As stated before, the RDF triples can now be seen as tuples of three IDs. Therefore, the Triples section models the graph of relationships among the dataset terms. By understanding the typical properties of RDF graphs, we can come up with more efficient ways of representing this information, both to reduce the overall size, but also to provide efficient search operations.

Popular and well-known libraries (like RDFLib [10]) provide additional mechanisms on top of this HDT data structure to translate general SPARQL queries to the underlying data sources. HDT datasets can be generated with a script but both Wikidata and DBpedia HDT versions were already made available by the HDT community. Those were used during this competition[2].

3.5 Selecting the best candidate embedding

At this point, interpretable embeddings of depth 2 were generated for multiple candidates, originating from a single cell within the major column of our structured file. Within these interpretable embeddings, a next function will search for matching information residing in the other cells on the same row from which the candidate embeddings were generated. In our example, the text description of Chris Martin can return two Wikidata entities: one is the Coldplay singer (Q712860), the other candidate is an American football player (Q519982). The INK embedding generated for the Coldplay singer will contain the information residing in the other cells of that same row (the is_part, birth date and occupation relation). As an embedding of depth 2 is provided, the labels of these associated nodes are incorporated in the embedding. The best candidate or embedding can be easily selected by counting the number of times additional information residing in other cells from the same row, is also available in the generated embedding of the candidate. In our example, the Chris Martin Coldplay singer embedding will have a matching count of 3, while the other Chris Martin has a zero match count. Therefore, the Chris Martin Coldplay singer annotation is selected as the best candidate in this situation.

3.6 Filling Additional Cells based on Selected Candidates

Selecting the best candidate according to the embedding also provides all information to the cells within the same row of the original structured file. Instead of

[2] `https://www.rdfhdt.org/datasets/`

performing an entity lookup for those cells, the annotations are directly derived from the major column embeddings. This eventually reduces both the costs of performed SPARQL lookups and external API calls. The annotations are all stored within a dictionary and output for the CEA task.

3.7 Defining Relationships Between Cells

Similarly, the embedding also provides the information going from the major column cell to the neighbouring cells within the same row. This information is directly derived from the provided embedding by selecting the object or data property relationships from the INK embedding. The combination of a relationship between two cells is kept within a dictionary. After the procedure to provide annotations for all cells is finished, the relationships between all those two cells are counted and the relationship with the maximal count is returned for the CPA task.

3.8 Defining column types based on additional type embeddings

The previous process is iterated for each cell within our major column. After all those annotations are provided, the column type can be derived from all cells containing annotated values. For all cells, an INK embedding of depth 1 is generated and the rdf:type for DBpedia and Wikidata P31 relationships are kept to determine the column type annotations. Again, the annotation with the highest count value is kept and returned for the CTA task.

4 Implementation

Both MAGIC[3] and INK[4] are implemented in Python and are made available on Github for future research. INK's implementation details can be found in the original paper. The MAGIC code is designed to perform evaluations for both DBpedia and Wikidata tasks but can be adapted to any other task.

- The code to select the major column is currently left outside MAGIC. This makes it possible for users to provide either this major column by itself or to write specific code to determine this column within a structured file.
- The code to search for entity candidates is abstracted. This ensures that future approaches can integrate other entity search API's (such as the DBpedia lookup service) without redesigning the internal MAGIC code. Also, additional preprocessing steps, such as translations, spell checks, etc. can be added to this component in the future.
- INK abstracts which data source it uses to generate the embeddings. In this version of the system, the embeddings were generated using an HDT backend and an HDT INK connector was made based on RDFlib to perform

[3] `https://github.com/IBCNServices/MAGIC`
[4] `https://github.com/IBCNServices/INK`

this task. In future projects, other already existing connectors (such as the INK Stardog[5] connector to connect to a triple store) can be used or created when needed.

5 Evaluation Results

The “Tabular Data to Knowledge Graph Matching” competition of 2021 consisted of three rounds in which multiple structured CSV files had to be annotated for either one, two or all CEA, CPA and CTA tasks. The different rounds consisted of multiple datasets from various domains where either DBpedia or Wikidata annotations were requested.
In total, four different metrics were used to evaluate the system. On the one hand, we had the F_1-score, which is a harmonic mean of the precision and recall of our system:

$$\begin{aligned} \text{precision} &= \frac{\#\text{correct annotations}}{\#\text{annotations made}} \\ \text{recall} &= \frac{\#\text{correct annotations}}{|\text{target cells}|} \\ F_1 &= \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{1}$$

During the second round, new metrics were used for the CTA challenge. Perfect annotation were encouraged, and at same time one of its ancestors (okay annotation) were also evaluated. Thus we calculate Approximate Precision (APrecision), Approximate Recall (ARecall), and Approximate F1 Score (AF1).

$$\begin{aligned} \text{APrecision} &= \frac{\sum_{a \in all} \text{ annotations } g(a)}{\text{all annotations } \#} \\ \text{ARecall} &= \frac{\sum_{\text{col } \in \text{ all target columns}} (\text{max\_annotation\_score(col)})}{\text{all target columns } \#} \\ AF1 &= \frac{2\times \text{ APrecision } \times \text{ ARecall}}{\text{APrecision } + \text{ ARecall}} \end{aligned} \tag{2}$$

with # denotes the number, g(a) returns the full score 1.0 if a is a perfect annotation, returns $0.8^{d(a)}$ if a is an ancestor of the perfect annotation and its depth to the perfect annotation d(a) is not larger than 5, returns $0.7^{d(a)}$ if a is a descendent of the perfect annotation and its depth to the perfect annotation d(a) is not larger than 3 and returns 0 otherwise. $max\_annotation\_score(col)$ returns g(a) if col has an annotation a, and 0 if col has no annotation.
All evaluations were performed on the same 32 core Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz cluster node with 125 gigabyte RAM. The Wikdata HDT datasource of 3 march 2020 was used for all Wikidata related tasks. The October 2016 English DBpedia HDT datasource was used for all DBpedia related tasks. The results of our approach are summarized in Table 1 respectively.

[5] https://www.stardog.com

	CEA		CTA		CPA	
	F1	Precision	F1	Precision	F1	Precision
Round 1						
DBpedia tables	0.184	0.506	0.159	0.628	/	/
Wikidata tables	DNE	DNE	DNE	DNE	/	/
Round 2						
BioTable Wikidata	0.837	0.838	0.916	0.916	0.838	0.888
HardTable Wikidata	0.836	0.947	0.757	0.681	0.865	0.954
Round 3						
BioDivTab Wikidata	0.100	0.253	0.142	0.192	/	/
HardTableR3 Wikidata	0.641	0.721	0.687	0.687	0.788	0.936
GitTables	/	/	DNE	DNE	/	/

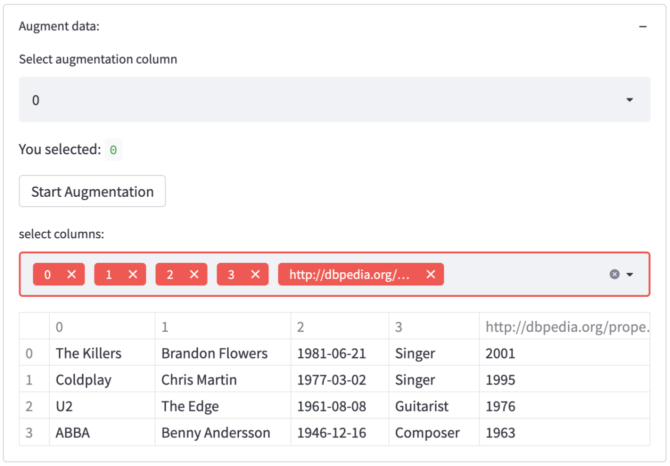
Table 1. Results obtained by the magic annotation system within the "Tabular Data to Knowledge Graph Matching" competition. / indicate that no evaluation mechanism was provided to evaluate these results. DNE (Did Not Execute) represents the tasks for which MAGIC did not provide any useful results.

The MAGIC system has competitive results when a clear CPA task has been defined. When such a task and corresponding links between the columns within our structured files is not defined, the system has more difficulties selecting the correct entity descriptions. For some datasets, our system extracted too many candidate embeddings to evaluate. This resulted in memory issues and loss of information within the corresponding tasks. The evaluations of these datasets (in particular, the Wikidata tables and GitTables) are therefore not provided in this table.

6 Data Augmentation

Besides the provided annotations for the CEA, CPA and CTA task within the "Tabular Data to Knowledge Graph Matching" competition, the MAGIC framework holds an additional advantage compared to its competitors. The INK embeddings generated in the magic system are interpretable and used to match the nearby cells within the same row. Additionally, information originating provided from a new relationship, which holds for all cells within a column can be added when it is available from the interpretable embedding. When e.g., we annotated all cells within our structured file in Figure 1. Combining all INK embeddings of Col0 will reveal additional information regarding the Bands listed in this file. One such additional relationship could be the Inception year, which is information that can easily be added as a new column to our original dataset. MAGIC can automate this process, revealing new possibilities to extend the original dataset with new information.

To make these benefits even more tangible, an additional GUI application has been developed that displays both the annotation and augmentation parts. An



	0	1	2	3	http://dbpedia.org/prope.
0	The Killers	Brandon Flowers	1981-06-21	Singer	2001
1	Coldplay	Chris Martin	1977-03-02	Singer	1995
2	U2	The Edge	1961-08-08	Guitarist	1976
3	ABBA	Benny Andersson	1946-12-16	Composer	1963

Fig. 4. MAGIC GUI application to augment annotated, structured files.

example of the GUI is visualized in Figure 4. A video about this GUI application, using the basic example of Figure 1, is also made availabe[6]

7 Conclusion and Future Work

In this paper, a system to annotate and augment a structured file with semantic knowledge is being proposed. This system shows the benefits of combining the interpretable embedding technique INK with a semantic annotation tool. Future work can now focus on both the preprocessing and post-processing functionalities to improve the generated annotations. Currently, matches are provided on exact string comparisons, without taking any malformed or misspelt text into account. The generated embeddings are also not used to detect wrongly annotated cells. Simple outlier detection or clustering tools based on the generated embedding of a single column can already help to filter those wrong annotations. At last, a thorough evaluation is needed of how this system can help to augment existing datasets and how this augmented data can help in more broad ML tasks.

8 Acknowledgements

Bram Steenwinckel (1SA0219N) is funded by a strategic base research grant of the Fund for Scientific Research Flanders (FWO).

[6] `https://www.youtube.com/watch?v=ZhTKxcTBZNE`

References

1. Nora Abdelmageed and Sirko Schindler. Jentab: Matching tabular data to knowledge graphs. In *SemTab@ ISWC*, pages 40–49, 2020.
2. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.
3. Rabia Azzi and Gayo Diallo. Amalgam: making tabular dataset explicit with knowledge graph. In *SemTab@ ISWC*, pages 9–16, 2020.
4. Marco Cremaschi, Roberto Avogadro, Andrea Barazzetti, and David Chieregato. Mantistable se: an efficient approach for the semantic table interpretation. In *SemTab@ ISWC*, pages 75–85, 2020.
5. Javier D Fernández, Miguel A Martínez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. Binary rdf representation for publication and exchange (hdt). *Journal of Web Semantics*, 19:22–41, 2013.
6. Sainyam Galhotra, Udayan Khurana, Oktie Hassanzadeh, Kavitha Srinivas, Horst Samulowitz, and Miao Qi. Automated feature enhancement for predictive modeling using external knowledge. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 1094–1097. IEEE, 2019.
7. Viet-Phi Huynh, Jixiong Liu, Yoan Chabot, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. Dagobah: Enhanced scoring algorithms for scalable annotations of tabular data. In *SemTab@ ISWC*, pages 27–39, 2020.
8. Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, and Kavitha Srinivas. Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In *European Semantic Web Conference*, pages 514–530. Springer, 2020.
9. Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Kavitha Srinivas, and Vincenzo Cutrona. Results of semtab 2020. In *CEUR Workshop Proceedings*, volume 2775, pages 1–8, 2020.
10. D Krech. Rdflib: A python library for working with rdf. *Online https://github.com/RDFLib/rdflib*, 2006.
11. Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011.
12. Phuc Nguyen, Ikuya Yamada, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. Mtab4wikidata at semtab 2020: Tabular data annotation with wikidata. In *SemTab@ ISWC*, pages 86–95, 2020.
13. Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer, 2016.
14. Bram Steenwinckel, Gilles Vandewiele, Pieter Bonte, Michael Weyns, Heiko Paulheim, Petar Ristoski, Filip De Turck, and Femke Ongenae. Walk extraction strategies for node embeddings with rdf2vec in knowledge graphs. In *International Conference on Database and Expert Systems Applications*, pages 70–80. Springer, 2021.
15. Bram Steenwinckel, Gilles Vandewiele, Filip De Turck, and Femke Ongenae. Csv2kg: Transforming tabular data into semantic knowledge. *SemTab, ISWC Challenge*, 2019.
16. Bram Steenwinckel, Gilles Vandewiele, Michael Weyns, Terencio Agozzino, Filip De Turck, and Femke Ongenae. Ink: knowledge graph embeddings for node classification. *Data Mining and Knowledge Discovery*, page in production, 2021.
17. Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.