TorchicTab: Semantic Table Annotation with Wikidata and Language Models

Ioannis Dasoulas[1,2,3,†], Duo Yang[1,2,3,†], Xuemin Duan[1,2,3] and Anastasia Dimou[1,2,3,*]

[1]KU Leuven, Department of Computer Science, B-2860, Sint-Katelijne-Waver, Belgium
[2]Flanders Make@KULeuven, B-3000 Leuven, Belgium
[3]Leuven.AI – KU Leuven institute for AI, B-3000 Leuven, Belgium

Abstract

An abundance of tabular data exists and is used by a wide range of applications. However, a big portion of these data lack the semantic information necessary for users and machines to properly understand them. This lack of table semantic understanding impedes their usage in data analytics pipelines. Solutions to semantically interpret tables exist but they are focused on specific annotation tasks and types of tables, and rely on large knowledge bases, making it difficult to re-use in real-world settings. Thus, more robust systems that produce more precise annotations and adapt to different table types are needed. The Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) was introduced in an effort to benchmark semantic table interpretation systems, by evaluating them over diverse datasets and tasks. In this paper, we introduce TorchicTab, a versatile semantic table interpretation system able to annotate tables with varied structures by using either an external knowledge graph, such as Wikidata, or annotated tables with pre-defined terms for training. We evaluate our proposed system according to the different annotation tasks of the SemTab challenge. The results show that our system can produce accurate annotations for different tasks across varied datasets.

Keywords

Semantic Annotation, Knowledge Graph, Wikidata, Entity Linking, Property Linking, Language Model

1. Introduction

Semantic Table Interpretation (STI) aims to understand and semantically annotate tables, leveraging external knowledge bases as reference [1, 2, 3]. To extract meaningful insights from tables and unleash their full potential, understanding their semantic structure and underlying meaning is needed. The Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)[1] was introduced to benchmark systems, providing different tasks and datasets

SemTab'23: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching 2023, co-located with the 22nd International Semantic Web Conference (ISWC), November 6-10, 2023, Athens, Greece
*Corresponding author.
†These authors contributed equally.
ioannis.dasoulas@kuleuven.be (I. Dasoulas); duo.yang@kuleuven.be (D. Yang); xuemin.duan@kuleuven.be (X. Duan); anastasia.dimou@kuleuven.be (A. Dimou)
http://orcid.org/0000-0002-8803-1244 (I. Dasoulas); http://orcid.org/0009-0008-5942-3397 (D. Yang); http://orcid.org/0000-0002-3256-8341 (X. Duan); https://orcid.org/0000-0003-2138-7972 (A. Dimou)
0000-0002-8803-1244 (I. Dasoulas); 0009-0008-5942-3397 (D. Yang); 0000-0002-3256-8341 (X. Duan); 0000-0003-2138-7972 (A. Dimou)

[1]https://www.cs.ox.ac.uk/isg/challenges/sem-tab/

[4, 5, 6, 7]. Systems leverage different methods to generate annotations by analyzing large knowledge bases [4, 5, 6, 7, 8] or employing classification over training examples [9, 10].

In past editions of the SemTab challenge, heuristic systems, whose scoring is based on majority voting, were the most accurate in annotating synthetic, real-world, and noisy tables for all tasks. However, as this year's newly introduced datasets reveal, the systems should be able to adapt to different table structures, missing content or limited knowledge bases. Existing systems assume that each table contains a unique subject column and all other columns refer to it. However, in real-world datasets, a subject column may be missing or some columns may provide information for a non-subject column. In addition, previous approaches rely on pre-existing RDF graphs, assuming that all relevant information is already present in another RDF graph. However, in some cases, such a knowledge base may not be available, but instead annotated table only.

We propose *TorchicTab*, a versatile annotation system, consisting of two complementary sub-systems, '*TorchicTab-Heuristic*' and '*TorchicTab-Classification*'. Each subsystem targets different scenarios of the semantic table interpretation problem and provides adequate solutions. '*TorchicTab-Heuristic*' requires access to RDF graphs to: (i) leverage heuristic data mining to link tables with existing RDF graphs, and annotate tables by inferring the subject column from other elements in the table; and (ii) predict properties and qualifiers from large RDF graphs, such as Wikidata, for n-ary relations expressed by three table columns, allowing it to capture complex relationships between entities, going beyond one-to-one mappings. '*TorchicTab-Classification*' does not require access to RDF graphs, but a sufficiently large number of annotated tables for training to: (i) leverage the pre-trained language model DODUO [10] that learns from provided table annotations, developing a rich understanding of the table content; (ii) adopt a sub-table sampling strategy and incorporate rich context to address the challenges from extra large tables.

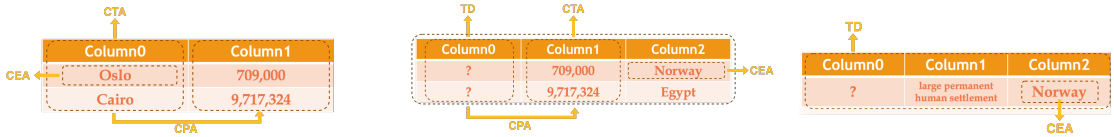
TorchicTab provides high-quality annotations both by leveraging heuristic methods to link table concepts with Wikidata entities and properties, as well as by applying classification methods to annotated tables for predicting columns and predicates for unseen tables. Experiments carried out in the context of SemTab 2023 challenge show promising results for all tasks, ranking TorchicTab amongst the top systems for all tasks it competed. While most systems focus on specific tasks, TorchicTab provides accurate annotations for all tasks, showcasing its adaptability to different scenarios, datasets and available knowledge bases. We plan to continue developing TorchicTab, providing more fine-grained solutions to improve its accuracy and efficiency.

The remainder of this paper is organized as follows: Section 2 shortly describes the SemTab competition and Section 3 provides an overview of related annotation systems. In Section 4, TorchicTab is explained, with details regarding its architecture and workflow. Section 5 demonstrates our experimental settings and results. Conclusions are presented in Section 6.

2. SemTab Challenge

The Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)[1] benchmarks table annotation systems over varied datasets and annotation tasks [4, 5, 6, 7]. In its fifth edition (SemTab 2023), it includes two rounds, each having a variety of tables to be annotated with concepts from Wikidata [11], DBPedia [12] or Schema.org [13].

The challenge consists of four annotation tasks: (1) The **Cell Entity Annotation** (CEA)



(a) Complete table (b) 'Horizontal' table (c) 'Entity' table

Figure 1: SemTab tasks

associates a table cell with an entity, given the input table (Figure 1a); (2) the **Column Type Annotation** (CTA) assigns a semantic type to a column; (3) the **Column Property Annotation** (CPA) discovers a semantic relation contained in the RDF graph that best represents the relation between two columns; and (4) the **Topic Detection** (TD) is newly introduced in this edition. It identifies the topic of a table that lacks a subject column and assigns a class (Figure 1b & 1c).

Tasks can also be classified depending on whether pre-existing RDF graphs are directly involved: Some tasks target an RDF graph, which the annotation system uses as a knowledge base to identify similarities between the entities and relations observed within the table and the ones in the RDF graph; Other tasks only have a pre-defined set of terms from popular RDF graphs, accompanied by numerous tables annotated with classes and properties contained in the terms set, which are formulated as multi-class classification problems.

Datasets of varying difficulty are provided to be annotated using Wikidata entities and properties. (1) The **Wikidata tables** provided in Round 1 (Figure 1a) contain synthetic information present in the Wikidata knowledge graph, covering a wide range of domains. Each table contains numerous columns, with the first being the subject column, and the rest providing context regarding the subject column. (2) The **Wikidata Column-Qualifier Tables**[2] are Wikidata tables with columns providing additional context to the main Wikidata properties, containing n-ary relations expressed by three table columns. (3) The **SOTAB Tables** refer to the WDC Schema.org Table Annotation Benchmark[3], generated by extracting Schema.org data from the Common Crawl. The data is grouped into separate tables for each class/host combination with the label spaces being constructed from two main sources, Schema.org and DBpedia. Tables lacking a subject column are introduced in the tFood[4] [14] dataset, which is derived from the food domain. (4) The **tFood Entity Tables** with a single missing entity (Figure 1c) which consist of a single row, with the first element being the subject element, but its content is missing. The rest of the row elements refer to the subject element, with the second containing a type description, e.g., in Figure 1c, the hidden element is the word 'Oslo', Norway's capital, represented by the entity *P*585 in Wikidata. Amongst other categories, *P*585 is an instance of the entity *P*515 which represents the concept 'city' and is described by the text excerpt: 'large permanent human settlement' in Wikidata. (5) The **tFood Horizontal Relational Tables** contain values related to multiple missing entities associated by the same topic, e.g., in Figure 1b, the subject column is missing but can be derived from the table's context.

[2]https://github.com/bennokr/semtab2023-CQA/
[3]http://webdatacommons.org/structureddata/sotab/
[4]https://zenodo.org/record/7828163

3. Related Work

Most semantic table interpretation systems rely on large knowledge bases to establish links between tables and related annotations or use annotated tables as training data to develop models generating annotations for unseen tables. Systems that reference large RDF graphs are mostly based on *external lookup methods* to retrieve candidates for values, headers and relationships within the table [15, 16, 17]. *Internal lookup methods* are also adopted by some systems, by indexing RDF graphs to analytics engines (e.g., Elasticsearch[5]), to improve entity lookup results and avoid using inconsistent online API services [18, 19, 20, 21] which often restrict usage, result set sizes and query response time. To deal with noisy table cells, combinations of multiple lookup strategies, e.g., fuzzy search [22, 15], and spelling correction [16, 23] are leveraged.

To rank the selected candidates, solutions consider heuristic *score-based* (DAGOBAH 2022 [15], KGCODE-Tab [16], MTab 2021 [23], LinkingPark [24]) or *graph-based* (IDLab [25], MantisTable [26]) ranking algorithms, comparing the context of the input tables with the sub-graphs of the candidate entities; *embedding techniques* capturing all information in the neighborhood of nodes within the graph and entities clustering based on their representations (DAGOBAH 2019 [19], Magic [27]); and *probabilistic graphical models* to rank the candidates (MTab 2019 [28], Mulwad et al. [29]). Existing solutions lack versatility and adaptability to real-world annotation problems. They heavily rely on large RDF graphs for annotation, assuming that all information is available and a subject column exists in each table, with all other columns referring to it.

In contrast to table annotation methods that rely on existing knowledge bases, tabular data annotation can be approached as a multi-class classification task given sufficient training annotation data. The WDC Schema.org Table Annotation Benchmark (SOTAB) [30] provides an evaluation framework for table annotation systems, and shows the difficulty of the benchmark using supervised classifications, which can be broadly categorized into two main approaches: *feature engineering-based* and *deep learning-based* [31]. The former extracts features (statistical information [32], textual similarities [33], etc.) from table rows and columns. These features are then used in conjunction with machine learning models (Random Forest [34], Logistic Regression [34], and K-Nearest Neighbor [32], etc.) to perform classification. However, feature engineering requires domain knowledge and manual effort to select and create relevant features, which can be time-consuming and may limit the model's ability to capture high-level abstractions.

Deep learning models learn complex representations automatically [35] and demonstrated potential in annotating of tabular data. TURL [9] pioneers the application of pre-trained language models to extract the structure information in relational Wikipedia tables. Although it exhibited promising results in various tasks including CTA and CPA, it heavily relies on table metadata (e.g., headers, caption, etc.), which are not available in SOTAB datasets. DODUO [10] was proposed as a generic framework also with a pre-trained language model. It only relies on cell values to make the prediction and demonstrated outstanding performance with multi-task learning in comparison to DOSOLO (i.e. without multi-task learning). To be specific, DODUO fine-tunes BERT [36] with a task combination (CTA & CPA) using table serialization. In our work, we further develop it with a sub-table sampling strategy for input data efficiency towards large tables and incorporate richer table context for table column and relation classification.

[5] https://www.elastic.co

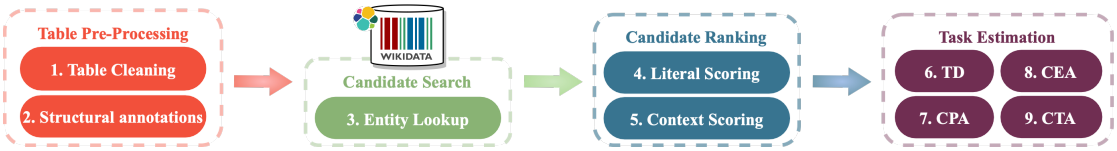


Figure 2: TorchicTab semantic table interpretation with Wikidata, given an input table.

4. TorchicTab Approach

The architecture of *TorchicTab* consists of 2 subsystems: (1) *TorchicTab-Heuristic* (Section 4.1) to annotate datasets with Wikidata as reference knowledge base, and (2) *TorchiTab-Classification* (Section 4.2) to annotate column type and properties within a given pre-defined set of terms.

4.1. TorchicTab-Heuristic

TorchicTab-Heuristic annotates complete tables (Section 4.1.1), tables lacking subject column (Section 4.1.2), or containing column-qualifiers (Section 4.1.3).

4.1.1. Semantic Annotation with RDF Graph Analysis

TorchicTab-Heuristic semantically annotates tables with entities and relations from Wikidata (Figure 2). It employs a combination of candidate lookup methods to maximize candidate coverage and heuristic scoring algorithms to assign various scores to rank the retrieved candidates in four steps: (1) table pre-processing, (2) candidate search, (3) ranking, and (4) task estimation.

Table Pre-Processing Firstly, all input tables go through a **pre-processing** step: Non-cell-values and HTML tags are removed, and incorrect encodings are fixed using the ftfy[6] tool. Afterwards, the input tables go through a **structural annotations** step to identify columns whose values can be represented as named entities in the reference RDF graph (NE-columns) or as literal values (L-columns). This way, candidates are retrieved only for cells that are expected to be represented as entities in the RDF graph. We employ a combination of REGEX pattern recognition and pre-trained SpaCy[7] Natural Language Processing (NLP) models to classify cells as named entities (NE) or literals (L). The column label is decided from the majority voting of the column cells' labels, thus splitting the table to NE-columns and L-columns.

Candidate search The candidate lookup step assigns candidate entities to all NE-column cells. We employ a combination of lookup strategies mainly targeting an Elasticsearch[5] index due to its robustness, ease of use and fast response times compared to online APIs which frequently exhibit inconsistencies and lack advanced search capabilities. We populated our Wikidata index with entity names, retrieved from Wikidata entity labels, and aliases from all Wikidata entities, referring to nicknames, short names, alternative names etc. (e.g., P1448 ⟶ official name, P1813 ⟶ short name, P2561 ⟶ name).

[6]https://ftfy.readthedocs.io/en/latest/
[7]https://spacy.io

The Elasticsearch index identifies candidates, supported by searches using the Wikidata public API[8], if the preliminary results are not sufficient. Given a target cell contained in a NE-column, a set of relevant candidate entities is retrieved from Wikidata, based on the entities' names or aliases' similarity to the cell. We employ 4 lookup strategies, in sequence, to maximize our system candidate coverage: (1) The **Complete Cell Strategy** uses the complete cell to query the Elasticsearch index; (2) the **Fuzzy Strategy** leverages Elasticsearch fuzzy queries for noisy cells and cells with spelling mistakes to retrieve candidates similar to the cell, but different by a small number of characters; (3) the **Cell Token Strategy** removes stop-words from the cell, splits it into tokens and queries the index for each token separately (e.g., the cell 'The Batman' represents the entity *Q*2695156 with label 'Batman'); and (4) the **Cell Token Combinations Strategy** removes stop-words from the cell and splits it into tokens. The index is queried for all possible combinations of tokens (e.g., the cell 'Messi Lionel' represents the entity *Q*615 with label 'Lionel Messi').

Our lookup step combines the robustness and searching capabilities of the Elasticsearch index with the relevance ranking algorithms of the Wikidata API to maximize the candidate coverage. We use the Levenshtein ratio [37] and BM25 metric [38] to keep the 20 best candidates after empirically observing that in most cases the most adequate entity was amongst the 20 top candidates. If there are too many or not many candidates, the Wikidata API is also used to provide additional candidates. For example, if the word 'Spain' is the cell value, the lookup will retrieve more than 20 candidates. However, in most tables, the entity that refers to the word 'Spain' is expected to be the country named Spain (*Q*29:Spain). To increase our coverage in these cases, we also leverage the relevance ranking algorithms that the Wikidata API provides.

Candidate ranking The candidates are ranked based on *string similarity*, comparing their labels to the table's cells, and *context similarity*, comparing their sub-graphs within the RDF graph to the cell's context within the table. Each candidate is assigned a similarity score to estimate the most probable annotations for entities, column types and relations. The candidates' scores and properties are used to calculate the most suitable relations between columns, via majority voting (CPA). The outputs of the CPA and candidate ranking are used to select the most suitable candidate for each NE-column cell (CEA). The outputs of CEA are then used for each NE-column to rank candidate types that could represent them and select the best (CTA).

A confidence score is assigned to each candidate which are ranked based on the literal similarity with the cell and semantic contextual similarity with the table's context. The **literal score** combines the Levenshtein ratio between the candidate entity and the cell with the BM25 retrieval metric of the candidate, calculated by the term frequency (TF) and inverse document frequency (ITF). For entities represented by multiple names and aliases, the highest Levenshtein ratio between their representations and the cell is considered.

For the **semantic contextual similarity**, we retrieve the candidate's sub-graph (i.e. all Wikidata triples where the candidate entity appears as subject) using the Wikidata API. We iteratively compare the candidate's neighborhood with the context around the corresponding cell (i.e. cells in the same row with the examined cell). To achieve that, we identify the Wikidata data types of the objects in the candidate's sub-graph (Wikidata ID String, Coordinate, Quantity,

[8]https://www.wikidata.org/wiki/Wikidata:REST_API

Time, Multilingual text), before applying similarity metrics to compare them to the table's context. We use similarity metrics to identify if certain objects of triples whose subjects are candidate entities appear in the input tables. For string, multilingual text and time datatypes and Wikidata IDs, we rely on Levenshtein ratio to measure the similarity with the context of the cell, after extracting all their labels and aliases. For amounts or coordinates, we use a simple number similarity function, comparing how close the objects are numerically with numerical values in the row of the examined cell. For each neighboring cell in the row of the cell we examine, we use these metrics to investigate their similarity with the candidate's sub-graph and keep the highest similarity score. The context score of the candidate is calculated by averaging over the similarity scores obtained for the row. The candidate's total score is calculated by multiplying the literal and context score. We used an amplification factor of 4 for the literal score, after empirically observing it yields better candidate rankings.

CPA The properties in the candidates' sub-graphs are ranked to select the most suitable among the subject column and each other column. We use majority voting to find the most fitting relationship between two columns. We combine the frequency that candidate properties appear and the context score of the candidate subjects or objects associated with this property. If a property exists in the RDF graph associating a candidate entity of a cell from the subject column with a candidate entity of a cell from an (*NE-column*) or a value similar to an (*L-column*) cell, and the cells are in the same row, the property is a *candidate property*. For a *candidate property*, we define property frequency as the number of triples in the RDF graph that contain a candidate entity of the subject column cell as their subject, the *candidate property* as their property, and a candidate entity of an NE-column cell or a value similar to the cell of an L-column as their object, with the corresponding cells being in the same row. The candidate property with the highest property frequency is selected. If multiple properties are tied, the one whose associated candidates have a higher combined context score is chosen.

CEA The candidate entities for each NE-column cell are ranked to distinguish the most suitable and are compared based on the total ranking score they accumulated in the candidate ranking step. The results of the CPA task are used to raise the score of candidates associated with the selected properties. When analyzing the subject column or a non-subject NE-column, the candidate entities that appear in the RDF graph as subjects or objects, respectively, in triples whose predicate is one of the properties selected in the previous step, get a bonus score. For each cell, the entity with the highest combination of total and CPA bonus score is chosen.

CTA The entity that best describes the CEA outputs of a NE-column is selected as the column's entity type. For each NE-column, we use its CEA outputs and employ majority voting to find the most suitable type. We identify candidate types in the RDF graph from entities connected with the CEA entities found with the $P31$ ('instance of') relationship. We include super-classes of these candidate types, linked with $P279$ ('subclass of') relationship, and their ancestors, for a broader range of candidate types, in case majority voting does not yield results by only investigating direct candidates. We define as *type frequency* the number of CEA output entities connected to the candidate type directly ($P31$), or indirectly (super-classes of the directly connected entities) and as *type distance* the number of entities in the shortest path between a CEA output entity and a candidate type. We find the candidate type with the highest type frequency and lowest type distance that represents most cell entities, in order for the column type to represent the semantic meaning of the column as specifically as possible. We want to

Figure 3: TorchicTab semantic table interpretation for Column-Qualifier Annotation.

have a low type distance in order for the column type to represent the semantic meaning of the column as specifically as possible. If multiple type candidates share the best score, we select amongst them using type frequency alone. If there is still no winner, we isolate the top scoring type candidates from the last step and select the one with the lowest type distance, and, if there are still ties, we investigate if there is a hierarchy in the RDF graph between the top candidate types ($P279$) and select the most specific type, according to their hierarchy. If they are not connected, we randomly select one of the top scoring candidate types as our CTA output.

4.1.2. Semantic Annotation for Subject-less Tables

To analyze and annotate the tFood[4] [14] that lack a subject column, the workflow we follow is similar to the one depicted in Figure 2, adjusting the candidate selection step to account for the lack of a subject column, which does not allow the retrieval of candidate entities. We extend our Elasticsearch, adding an index containing all Wikidata triples, as we can no longer rely on extracting candidates using only entities' names and aliases.

For tFood Entity Tables, after pre-processing and identifying the structural annotations, we proceed to retrieve the NE-columns' entity candidates. For the subject column, we use the type description to retrieve topic candidates, which are instances ($P31$) or sub-classes ($P279$) of classes described by the provided description. We rank the retrieved topic candidates using the context score module, comparing them with the row's content and the retrieved entity candidates for NE-column cells. The labels of the topic candidates are not considered during the ranking process since there is no indication about them in the table. The topic candidate with the highest context score is selected as the Topic Detection (TD) output for the system. For the CEA task, we scan the RDF graph for triples containing the TD output as subject. For each NE-column, the candidate that appears as object in one of the triples is selected.

4.1.3. Column-Qualifier Annotation

The Column-Qualifier Annotation predicts properties and qualifiers from Wikidata in a table[9] sourced for n-ary relations given by a set of property labels and qualifier labels. The workflow (Figure 3) mainly involves the Subject Calculation (steps 2 and 3)and the Property and Qualifier Calculation (steps 4, 5, 6, and 7).

Subject Calculation The calculation of the cell entities in the subject column is achieved by leveraging the Wikidata item link provided by the identified Wikipedia page. Specifically,

[9]https://github.com/bennokr/semtab2023-CQA/

we first employ the relevance ranking algorithms that the query action of the Wikipedia API[10] provides to get *n* (*n* defaults to 3) page title candidates based on a given cell value in the subject column (e.g., Silver Logie Award for Most Popular TV Presenter ⟶ ['Logie Award for Most Popular Presenter', 'Logie Awards', 'Logie Awards of 2022']).

Subsequently, we crawl the entire Wikipedia page contents of page title candidates using Beautiful Soup[11]. We determine the optimal page title by calculating the overlap rate between the content of each candidate page and the corresponding object cells and qualifier cells associated with the subject cell, where the candidate with the highest overlap rate is considered the best match. From the optimal Wikipedia page, we can obtain and access its QID through the 'Wikidata item' link in the 'Tools' side panel. Thus, we identify the subject cell entity from the hyperlink of the HTML element which has a title attribute containing the string 'Structured data on this page hosted by Wikidata' in the optimal page (e.g., Logie Award for Most Popular Presenter ⟶ https://www.wikidata.org/wiki/Special:EntityPage/Q6667534 ⟶ Q6667534).

Property and Qualifier Calculation We use Wikidata's SPARQL endpoint[12] to obtain candidate labels and then determine the optimal property and qualifier labels. We initiate a query to retrieve the properties and qualifiers associated with the identified subject QID, focusing on those that are present in the provided labels list as candidates, along with their corresponding object and qualifier values. We sort the property candidates by calculating the text similarity score between the value of each candidate's object and the value of the object cells in the table using the *token_set_ratio* function from FuzzyWuzzy[13] library (e.g., Q6667534's property P1346 (winner) appeared in the given labels list and obtains the highest score).

Then we employ the same text similarity method with extra weights obtained from the previous step to calculate the optimal qualifier and treat the corresponding property associated with the optimal qualifier as the optimal property. Specifically, we introduce a weighting factor to sum the obtained scores of property candidates and the text similarity score of the corresponding qualifier as the final score (e.g., Q6667534's property P1346 (winner) and qualifier P1686 (for work) obtain the final highest score). This weighting ensures that the final decision regarding the qualifier considers both the relevance and importance of the associated property. Given the potential variability in values within the subject column of the table, we adopt a grouping approach (Figure 3, step 1) where each unique value and its associated object and qualifier cells are treated as input for the Subject Calculation and Property and Qualifier Calculation to obtain the optimal property(s) and optimal qualifier(s). We use majority voting (Figure 3, step 8) to decide the final outcome.

4.2. TorchicTab-Classification

The CTA and CPA tasks for the SOTAB tables involve annotating table columns and relationships between the main column and other columns. *TorchicTab-Classification*, considers pre-defined terms from vocabularies like `Schema.org` [13] and `DBpedia`[12], unlike the previous subsystem

[10]http://en.wikipedia.org/w/api.php
[11]http://www.crummy.com/software/BeautifulSoup/
[12]https://query.wikidata.org/sparql
[13]https://pypi.org/project/fuzzywuzzy/

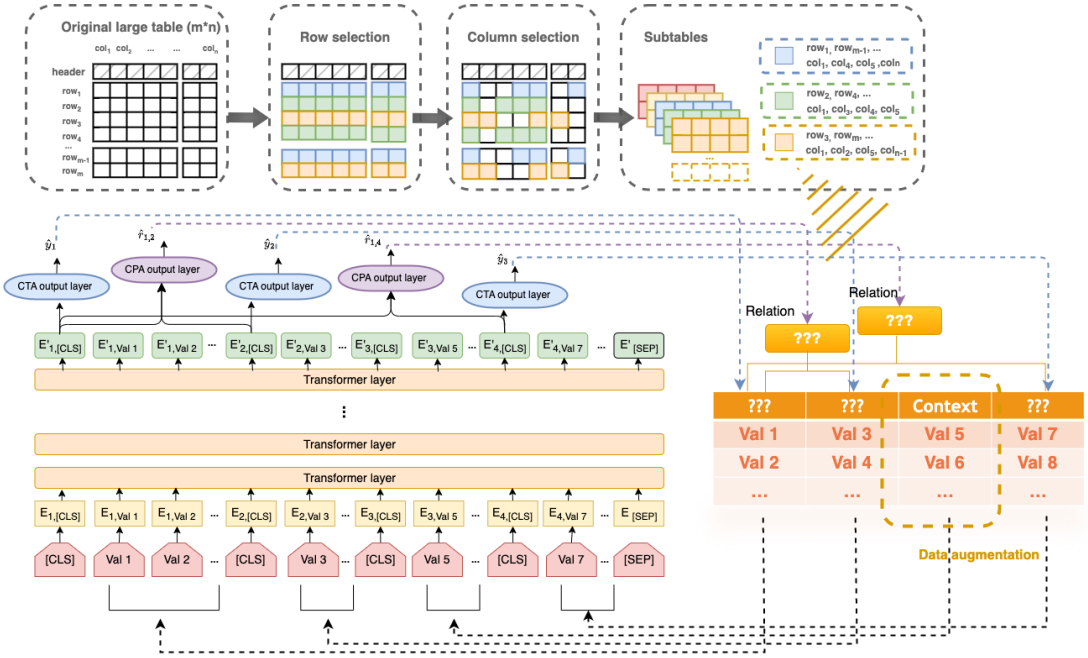


Figure 4: Sub-table sampling strategy (reworked figure from the DODUO architecture [10])

which focused on entity discovery in the RDF graph. We formulate both as multi-class classifications where each column or column pair can be annotated with only one label and apply DODUO [10], a multi-task learning framework based on pre-trained language models.

Given the complexity of SOTAB tables, we needed to balance performance and efficiency when applying DODUO to deal with these large tables containing numerous rows and columns. Due to the limitations imposed by the maximum length of token inputs (typically 512 tokens) in most language models, it becomes challenging to analyze the entire table within the DODUO architecture. Besides that, DODUO only models the information contained in labeled columns, disregarding the extensive context from the unlabeled columns, and it only captures information within those columns for upcoming predictions. For example, in certain cases, only one or two columns serve as labels or predicted targets, while many more unlabeled columns are ignored.

Sub-table Sampling Strategy We followed a sub-table sampling strategy, shown in Figure 3, to split the large tables and incorporate richer context within the original DODUO to tackle the challenges. We perform several steps before and after applying DODUO, including:
(1) Row selection A specific number of rows (e.g., 40 out of 56 rows of a table) was randomly selected; these rows are randomly divided into smaller equal sub-tables (e.g., 8 sub-tables with 5 rows each); each sub-table serves as a single unit as training sample after further processing.
(2) Column selection Each sub-table will be reduced to a maximum of 10 columns; the selection of these columns is random (e.g., 10 out of 14 columns). While some columns may be shared among sub-tables from the same large table, they may also have different columns.
(3) Token construction For each sub-table, we have 50 tabular cells to represent as an individual

Table 1
Semantic Annotation with KG Analysis Performance

Task	val			test		
	P	**R**	**F1**	**P**	**R**	**F1**
R1-WD-CEA	88.74	89.32	89.03	83.90	82.10	83.00
R1-WD-CTA	78.34	93.46	85.23	74.90	89.90	81.70
R1-WD-CPA	94.55	94.55	94.55	93.40	93.40	93.40
R1-TFOOD-ENT-CEA	97.44	87.50	92.20	-	-	-
R1-TFOOD-ENT-TD	75.20	74.67	74.94	-	-	-
R2-CQA	-	-	-	82.20	-	-

sample for training and predicting, thus the maximum tabular cell length is 10 sub-word level tokens to fulfill the limitation by the maximum length of token inputs (512) in language models.
4) Majority voting Once the column's type or relation predictions within sub-tables were done by DODUO, we employed a majority voting strategy to consolidate the final predicted outcome of the column or column pair, which involved aggregating the individual predictions based on sub-tables (8) and selecting the most frequent predicted type or relation as the final result.

Language detection was used to identify each table's language based on the 1st row, as 10% to 20% of the tables in SOTAB are not in English but rather in German, French, etc. This adds additional complexity to the annotation task and emphasizes the importance of applying a multilingual language model. We chose the pre-trained language model based on the top 104 languages with the largest Wikipedia using a masked language modeling (MLM) objective.

5. Experiments and Results

We present our results and experimental settings for Round 1 (R1) and 2 (R2) of SemTab 2023, elucidating the capabilities of TorchicTab. Our results include the f1-score (F1), precision (P) and recall (R) for the validation and test sets provided by the competition in each round.

TorchicTab-Heuristic *TorhicTab-Heuristic* was used for R1's Wikidata tables (R1-WD), tFood entity tables (R1-TFOOD-ENT), and R2's Column-Qualifier (R1-CQA) annotation. The experiments were run on Ubuntu 18.04 LTS server with a 12-threaded Intel(R) Xeon(R) CPU E5-2620. Multi-processing was used to accelerate the annotation by splitting the datasets into batches.

Our system efficiently annotates the CPA task of the R1-WD datasets but its score drops for CEA and CTA (Table 1). It produces high-quality annotations for the CEA task of the subject-less R1-TFOOD-ENT datasets. The TD results are lower because different entities may share the context presented in the table, making it difficult to select a single topic, given some context about it. The less context is available, the more probable is that multiple topics can be described by it. The CQA results also demonstrate the capability of our system to annotate qualifiers.

TorchicTab-Classification To determine the optimal thresholds for language model input, we evaluated the coverage percentages by examining different thresholds for the number of columns and rows, and cells' length across the entire dataset (Table 2). Furthermore, we

Table 2
SOTAB Dataset Statistics: **Lbl** refers to the label space in classification; **lng** to the percentage of non-Enligsh tables; **Row/Column Cov.** to the coverage percentage, e.g., 58.6% of tables within R1-SCH CTA dataset contain no more than 40 rows; **Length Cov.** to the coverage percentage of cell's length.

SOTAB	Task	Lbl	Tables	Lng	Row Cov.			Column Cov.			Length Cov.		
					40	80	120	5	10	15	5	10	15
R1-SCH	CTA	40	56,547	15.2	58.6	73.1	78.9	42.2	78.8	92.5	31.5	67.7	80.0
	CPA	50	53,722	15.5	54.5	70.2	77.3	26.0	76.0	92.50	40.6	67.2	78.7
R2-SCH	CTA	80	117,846	13.3	56.4	72.0	78.8	32.4	69.3	98.0	44.2	68.6	78.3
	CPA	105	101,394	16.9	52.1	67.9	75.5	23.5	69.4	90.3	49.6	72.4	84.3
R2-DBP	CTA	46	86,973	13.1	57.2	72.8	79.6	35.9	71.3	84.5	45.2	69.8	78.3
	CPA	49	64,052	18.1	53.2	68.9	76.1	27.5	74.9	94.3	46.5	70.9	80.4

examined the proportion of non-Enlgish tables through language detection across the dataset to determine if it is necessary to employ a multilingual version of language model.

Considering the input data efficiency and limitation of token inputs (typically 512 tokens) in language models, we decide how to efficiently fill in the 512 token inputs without high cost in computation from this analysis. We found that 73% of the tables in R1-SOTAB-SCH-CTA have a total of 80 rows or fewer, 79% have no more than 10 columns, and 68% of the cells have no more than 10 sub-word level tokens (Table 2). Based on these, we performed sub-table construction for large tables. The sampling settings within each table were configured to maximally 40 rows and 10 columns per epoch, and each sample (sub-table) was randomly selected to 5 rows and 10 columns at most. The maximum length of each cell was limited to 10 tokens at the sub-word level, aligning with the pre-trained language model's constraints, which resulted in 502 tokens as the maximum input length. This ensures that the generated sub-tables capture a representative portion of the table while maintaining a manageable size for training efficiency. Ablation studies were also performed to validate the effectiveness and efficiencies of our method towards large tables, including random selection, context column addition, and sub-table sampling.

We applied the Adam optimizer [39] with a learning rate of 2×10^{-5} for SOTAB-SCH (5×10^{-5} for SOTAB-DBP), with no warm-up but a linear decay schedule. The batch size was configured to be 30, all models were trained for 30 epochs using the training set only. The best models were selected based on the Macro F1-score of the validation set from the training iterations. DODUO refers to models co-trained with another task as a combination (CTA & CPA), where all labels are sourced from the same domain. DOSOLO refers to those trained with a single task. All classification-based experiments were conducted on a CentOS 7 server with Intel Xeon Gold 6240 CPUs and an NVIDIA TESLA V100 GPU (32GB GDDR). The models were fine-tuned using the pre-trained weights of the "bert-base-multilingual-cased" version[14].

DODUO outperformed DOSOLO by at least 2 percentage points in terms of performance across all tasks with different pre-defined labels, benefiting from the multitask learning (Table 3). These improvements were particularly evident in CPA prediction across different datasets, which was expected as it involves columns' types as the primary factor followed by the relationships among these columns. As the number of pre-defined terms for the label space increased, the

[14]https://huggingface.co/bert-base-multilingual-cased

Table 3
SOTAB CTA & CPA Peformance

Task	SOTAB	Method	val				test	
			P	R	Macro F1	Micro F1	P	Micro F1
CTA	R1-SCH	DODUO	93.28	94.83	92.76	94.14	-	-
		DOSOLO	92.08	93.18	91.33	92.80	-	-
	R2-SCH	DODUO	90.68	91.27	89.78	91.21	89.80	89.66
		DOSOLO	89.30	89.96	88.47	89.59	-	-
	R2-DBP	DODUO	91.97	93.77	91.25	92.34	90.93	89.72
		DOSOLO	88.95	90.15	88.14	89.47	-	-
CPA	R1-SCH	DODUO	94.66	94.87	94.51	94.96	-	-
		DOSOLO	91.99	92.03	91.62	92.06	-	-
	R2-SCH	DODUO	87.56	87.25	86.54	86.58	88.00	87.11
		DOSOLO	85.32	85.71	84.51	84.47	-	-
	R2-DBP	DODUO	91.05	93.31	91.69	92.72	90.68	90.49
		DOSOLO	86.01	88.88	86.49	88.58	-	-

model's inference ability decreased; we suspect this occured because R2-SOTAB-SCH contains twice as many labels in both CTA & CPA tasks (Table 2). Despite having a similar size of label space and fewer training tables, R1-SOTAB-SCH managed to outperform R2-SOTAB-DBP. This suggests that the labeling scheme plays a crucial role in determining the model's performance.

6. Conclusions and Future Work

We presented *TorchicTab* a table annotation system, with two complementary subsystems, *TorchicTab-Heuristic* and *TorchicTab-Classification*, to semantically interpret tables. Its heuristic data mining and ability to annotate tables lacking a subject column, infer complex n-ary relations, and train on existing annotated tables, showcase its adaptability and effectiveness, while its sub-table sampling strategy tackles the challenges of extra-large tables regarding classification. The results for all SemTab tasks are promising, ranking *TorchicTab* amongst the top systems in SemTab 2023 challenge and laying strong foundations for further advancements. In future work, we will apply optimizations to its heuristics to improve its efficiency, e.g., candidate scoring factors and alternative candidate ranking methods, and enhance its classification efficiency.

Acknowledgement

This research was partially supported by Flanders Make, the strategic research centre for the manufacturing industry and the Flanders innovation and entrepreneurship (VLAIO). The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

References

[1] Z. Zhang, Effective and Efficient Semantic Table Interpretation using TableMiner+, Semantic Web 8 (2017) 921–957. doi:10.3233/SW-160242.

[2] G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and Searching Web Tables Using Entities, Types and Relationships, Proceedings of the VLDB Endowment 3 (2010) 1338–1347. doi:10.14778/1920841.1921005.

[3] Z. Syed, T. Finin, V. Mulwad, A. Joshi, et al., Exploiting a Web of Semantic Data for Interpreting Tables, in: Proceedings of the second web science conference, 2010.

[4] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems, in: The Semantic Web – ESWC 2020, volume 12123, Springer, Cham, 2020, pp. 514–530. doi:10.1007/978-3-030-49461-2_30.

[5] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, V. Cutrona, Results of SemTab 2020, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 2775, CEUR Workshop Proceedings, 2020, pp. 1–8. URL: https://ceur-ws.org/Vol-2775/paper0.pdf.

[6] V. Cutrona, J. Chen, V. Efthymiou, O. Hassanzadeh, E. Jiménez-Ruiz, J. Sequeda, K. Srinivas, N. Abdelmageed, M. Hulsebos, D. Oliveira, et al., Results of SemTab 2021, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3103, CEUR Workshop Proceedings, 2021, pp. 1–12. URL: https://ceur-ws.org/Vol-3103/paper0.pdf.

[7] N. Abdelmageed, J. Chen, V. Cutrona, V. Efthymiou, O. Hassanzadeh, M. Hulsebos, E. Jiménez-Ruiz, J. Sequeda, K. Srinivas, Results of SemTab 2022, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3320, CEUR Workshop Proceedings, 2022, pp. 1–13. URL: https://ceur-ws.org/Vol-3320/paper0.pdf.

[8] A. Alobaid, O. Corcho, Balancing coverage and specificity for semantic labelling of subject columns, Knowledge-Based Systems 240 (2022) 108092. doi:10.1016/j.knosys.2021.108092.

[9] X. Deng, H. Sun, A. Lees, Y. Wu, C. Yu, Turl: Table Understanding through Representation Learning, SIGMOD Rec. 51 (2022) 33–40. doi:10.1145/3542700.3542709.

[10] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, W.-C. Tan, Annotating Columns with Pre-trained Language Models, in: Proceedings of the 2022 International Conference on Management of Data, SIGMOD '22, Association for Computing Machinery, 2022, pp. 1493–1503. doi:10.1145/3514221.3517906.

[11] D. Vrandečić, M. Krötzsch, Wikidata: A Free Collaborative Knowledgebase, Communications of the ACM 57 (2014) 78–85. doi:10.1145/2629489.

[12] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A nucleus for a web of open data, in: The Semantic Web, volume 4825, Springer, 2007, pp. 722–735. doi:10.1007/978-3-540-76298-0_52.

[13] R. V. Guha, D. Brickley, S. Macbeth, Schema.org: Evolution of Structured Data on the Web, Communications of the ACM 59 (2016) 44–51. doi:10.1145/2844544.

[14] N. Abdelmageed, E. Jimènez-Ruiz, O. Hassanzadeh, B. König-Ries, tFood: Semantic Table

Annotations Benchmark for Food Domain, 2023. doi:10.5281/zenodo.7828163.
[15] V.-P. Huynh, Y. Chabot, T. Labbé, J. Liu, R. Troncy, From Heuristics to Language Models: A Journey Through the Universe of Semantic Table Interpretation with DAGOBAH, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3320, CEUR Workshop Proceedings, 2022, pp. 45–58. URL: https://ceur-ws.org/Vol-3320/paper6.pdf.
[16] X. Li, S. Wang, W. Zhou, G. Zhang, C. Jiang, T. Hong, P. Wang, KGCODE-Tab Results for SemTab 2022, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3320, CEUR Workshop Proceedings, 2022, pp. 37–44. URL: https://ceur-ws.org/Vol-3320/paper5.pdf.
[17] N. Abdelmageed, S. Schindler, JenTab: Do CTA Solutions Affect the Entire Scores?, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3320, CEUR Workshop Proceedings, 2022, pp. 72–79. URL: https://ceur-ws.org/Vol-3320/paper8.pdf.
[18] M. Cremaschi, R. Avogadro, D. Chieregato, s-elBat: a Semantic Interpretation Approach for Messy taBle-s, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3320, CEUR Workshop Proceedings, 2022, pp. 59–71. URL: https://ceur-ws.org/Vol-3320/paper7.pdf.
[19] J. Liu, R. Troncy, DAGOBAH: An End-to-End Context-Free Tabular Data Semantic Annotation System, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 2553, CEUR Workshop Proceedings, 2019, pp. 41–48. URL: https://ceur-ws.org/Vol-2553/paper6.pdf.
[20] A. Thawani, M. Hu, E. Hu, H. Zafar, N. T. Divvala, A. Singh, E. Qasemi, P. A. Szekely, J. Pujara, Entity Linking to Knowledge Graphs to Infer Column Types and Properties, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 2553, CEUR Workshop Proceedings, 2019, pp. 25–32. URL: https://ceur-ws.org/Vol-2553/paper4.pdf.
[21] D. Oliveira, M. d'Aquin, ADOG-Annotating Data with Ontologies and Graphs, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 2553, CEUR Workshop Proceedings, 2019, pp. 1–6. URL: https://ceur-ws.org/Vol-2553/paper1.pdf.
[22] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, MTab4Wikidata at SemTab 2020: Tabular Data Annotation with Wikidata, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 2775, CEUR Workshop Proceedings, 2020, pp. 86–95. URL: https://ceur-ws.org/Vol-2775/paper9.pdf.
[23] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, SemTab 2021: Tabular Data Annotation with MTab Tool, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3103, CEUR Workshop Proceedings, 2021, pp. 92–101. URL: https://ceur-ws.org/Vol-3103/paper8.pdf.
[24] G. Limaye, S. Sarawagi, S. Chakrabarti, LinkingPark: An Integrated Approach for Semantic Table Interpretation, Journal of Web Semantics 74 (2020) 100733. doi:10.1016/j.websem.2022.100733.
[25] B. Steenwinckel, G. Vandewiele, F. De Turck, F. Ongenae, CSV2KG: Transforming Tabular Data into Semantic Knowledge, in: Proceedings of the Semantic Web Challenge on Tabular

Data to Knowledge Graph Matching (SemTab), volume 2553, CEUR Workshop Proceedings, 2019, pp. 33–40. URL: https://ceur-ws.org/Vol-2553/paper5.pdf.

[26] M. Cremaschi, R. Avogadro, A. Barazzetti, D. Chieregato, E. Jiménez-Ruiz, MantisTable SE: an Efficient Approach for the Semantic Table Interpretation, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 2775, CEUR Workshop Proceedings, 2020, pp. 75–85. URL: https://ceur-ws.org/Vol-2775/paper8.pdf.

[27] B. Steenwinckel, F. De Turck, F. Ongenae, MAGIC: Mining an Augmented Graph using INK, starting from a CSV, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3103, CEUR Workshop Proceedings, 2021, pp. 68–78. URL: https://ceur-ws.org/Vol-3103/paper6.pdf.

[28] P. Nguyen, N. Kertkeidkachorn, R. Ichise, H. Takeda, Mtab: Matching Tabular Data to Knowledge Graph using Probability Models, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 2553, CEUR Workshop Proceedings, 2019, pp. 7–14. URL: https://ceur-ws.org/Vol-2553/paper2.pdf.

[29] V. Mulwad, T. Finin, A. Joshi, Semantic Message Passing for Generating Linked Data from Tables, in: The Semantic Web – ISWC 2013, volume 8218, Springer International Publishing, 2013, pp. 363–378. doi:10.1007/978-3-642-41335-3_23.

[30] K. Korini, R. Peeters, C. Bizer, SOTAB: The WDC Schema. org table annotation benchmark, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), volume 3320, CEUR Workshop Proceedings, 2022, pp. 14–19. URL: https://ceur-ws.org/Vol-3320/paper1.pdf.

[31] J. Liu, Y. Chabot, R. Troncy, V.-P. Huynh, T. Labbé, P. Monnin, From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods, Journal of Web Semantics 76 (2022) 100761. doi:10.1016/j.websem.2022.100761.

[32] S. Neumaier, J. Umbrich, J. X. Parreira, A. Polleres, Multi-level Semantic Labelling of Numerical Values, in: The Semantic Web – ISWC 2016, volume 9981, Springer International Publishing, 2016, pp. 428–445. doi:10.1007/978-3-319-46523-4_26.

[33] S. K. Ramnandan, A. Mittal, C. A. Knoblock, P. Szekely, Assigning Semantic Labels to Data Sources, in: European Semantic Web Conference, volume 9088, Springer, 2015, pp. 403–417. doi:10.1007/978-3-319-18818-8_25.

[34] M. Pham, S. Alse, C. A. Knoblock, P. Szekely, Semantic labeling: A domain-independent approach, in: The Semantic Web – ISWC 2016, volume 9981, Springer International Publishing, 2016, pp. 446–462. doi:10.1007/978-3-319-46523-4_27.

[35] Y. Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning 2 (2009) 1–127.

[36] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, volume 1, Association for Computational Linguistics, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[37] L. Yujian, L. Bo, A Normalized Levenshtein Distance Metric, IEEE transactions on pattern analysis and machine intelligence 29 (2007) 1091–1095. doi:10.1109/TPAMI.2007.1078.

[38] S. Robertson, H. Zaragoza, et al., The Probabilistic Relevance Framework: BM25 and

Beyond, Foundations and Trends in Information Retrieval 3 (2009) 333–389. doi:10.1561/1500000019.

[39] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, in: Proceedings of International Conference on Learning Representations, ICLR, 2015. URL: http://arxiv.org/abs/1412.6980.