SemTex: A Hybrid Approach for Semantic Table Interpretation

Emil G. Henriksen[1], Alan M. Khorsid[1], Esben Nielsen[1], Adam M. Stück[1], Andreas S. Sørensen[1] and Olivier Pelgrin[1,*]

[1]Aalborg University (AAU), Department of Computer Science, Selma Lagerlöfs Vej 300, 9220 Aalborg East, Denmark

Abstract

Accurate Semantic Table Interpretation (STI) annotation has a significant impact on interpreting unlabeled data for data analysis. In this paper, we present SemTex, a system for solving the three tasks Cell Entity Annotation (CEA), Cell Type Annotation (CTA) and Cell Property Annotation (CPA) in the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). We utilise a hybrid approach combining analysis of relationships in knowledge graphs and gradient boosting for annotation. We document and benchmark our performance using datasets from the SemTab challenge 2022 and 2023. Our approach yields competitive results compared to the current state-of-the-art tools in all three tasks.

Keywords

Semantic Table Interpretation, Tabular Data, SemTab Challenge, Knowledge Graph, Gradient Boosting

1. Introduction

Tabular data constitutes one of the most prevalent formats for the dissemination and analysis of information. However, such tables often contain inaccurate data, incomplete annotations, and more. Semantic Table Interpretation (STI) aims at providing table annotations by matching elements such as columns, cells, and column relationships, with known entities from existing Knowledge Graphs. In this paper, we present SemTex, a tool created to perform three types of STI annotations, namely cell annotations, property annotations and type annotations, by exploring the Wikidata knowledge graph. We participate in the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) 2023 challenge, where the performance and accuracy of SemTex is evaluated in a controlled environment against similar systems. SemTab is a yearly challenge that covers four categories, namely Cell Entity Annotation (CEA), Cell Property Annotation (CPA), Cell Type Annotation (CTA) and the newly added Table Topic Detection (TD). SemTex will only participate in Round #1, specifically focusing on the first three

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.

emil-g-h@hotmail.com (E. G. Henriksen); alan18@hotmail.dk (A. M. Khorsid); esbenn179@gmail.com (E. Nielsen); adam@adast.dk (A. M. Stück); todes92@protonmail.com (A. S. Sørensen); olivier@cs.aau.dk (O. Pelgrin)

https://alankhorsid.github.io/AlkhorithmCV/ (A. M. Khorsid); https://adast.dk (A. M. Stück)

0009-0002-3130-750X (E. G. Henriksen); 0009-0002-6115-7282 (A. M. Khorsid); 0009-0000-6771-4655 (E. Nielsen); 0009-0003-9159-9150 (A. M. Stück); 0000-0003-0258-6001 (A. S. Sørensen); 0000-0002-1025-9687 (O. Pelgrin)

categories mentioned above and solely on the WikidataTablesR1 dataset. The SemTab challenge has been running for multiples years, with numerous participants and approaches. s-elBat [1] approaches CEA by scoring potential annotations using measures like Levenshtein distance. Afterwards, they solve CTA and CPA by analysing the frequencies of select properties of the CEA predictions. Another tool, KGCODE-Tab [2], uses the Bing search engine to perform spell-correction of the dataset mentions. Additionally, in their preprocessing phase, they implement ways to identify the entity columns of tables using Named Entity Recognition (NER) tools. SemTex's approach draw inspiration from those tools, and extend it further by introducing machine learning step to improve the annotation process.

2. Preliminaries

SemTab consists of two tracks, of which we focus on the Accuracy Track. The goal of this track is to match tabular data to semantic entities, types, and relationships. Specifically, we will work with the three subtasks, Cell Entity Annotation (CEA), Cell Type Annotation (CTA), and Cell Property Annotation (CPA). For each round of the challenge, test and validation datasets representing tables are given as CSV files. The validation set is meant for training purposes and includes corresponding ground truth assignments, while the test set is meant for evaluation.

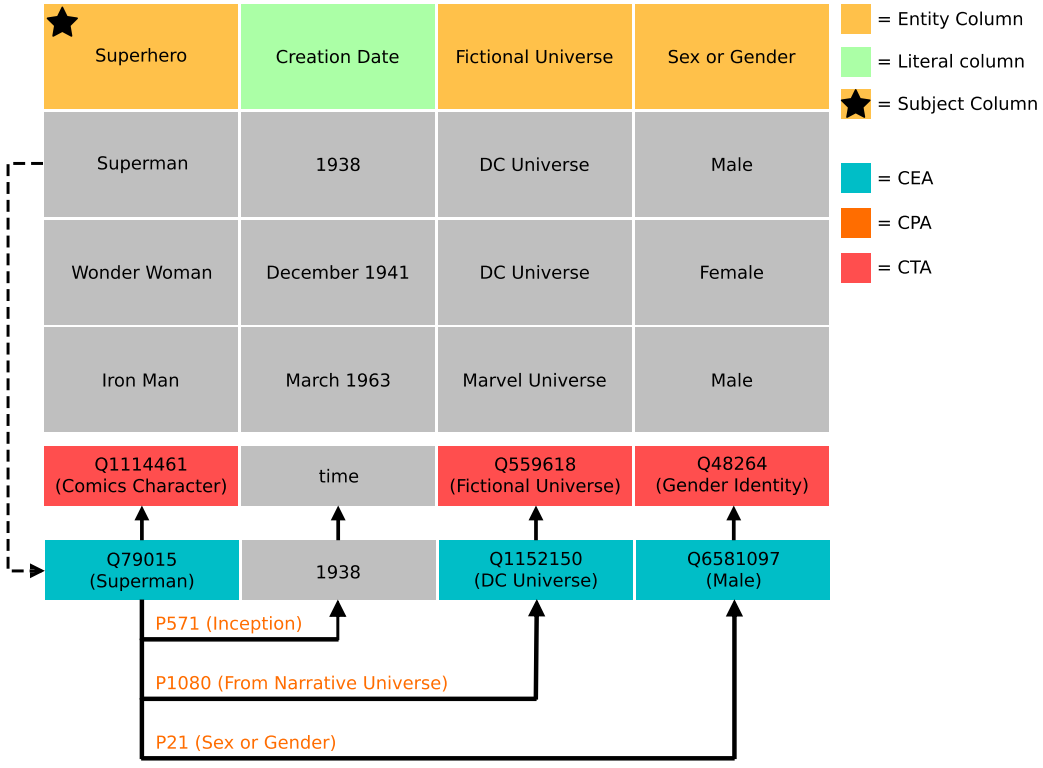

Figure 1: Example of a table with annotations.

Consider Figure 1 as an example table. Each table consists of cells that contain textual data

called *mentions*. The CEA task involves matching mentions to Wikidata entities. However, not all cells represent entities. Some represent literals like the 'Creation Date' column. Columns that contain literals are *literal columns*, while columns that contain mentions representing entities are *entity columns*. Furthermore, each table contains a *subject column*, which is the entity column that the other columns refer to. In the example, 'Superhero' is the subject column. The purpose of the CTA task is to assign semantic types to entity columns. Here, 'Comics Character' is the CTA assignment to the first column. Note that literal columns are not assigned CEA and CTA assignments. Lastly, CPA involves assigning properties to the relationships between pairs of columns. These relationships are primarily between the subject column and other columns. For example, the property 'From Narrative Universe' is the relationship between the 'Superhero' and 'Fictional Universe' columns. Submissions for each task are scored using an F1-score.

3. The SemTex System

3.1. Component Overview

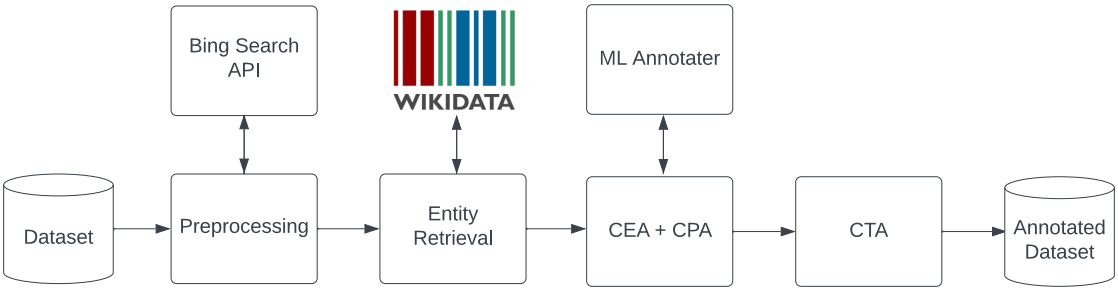

Figure 2: The SemTex pipeline.

To describe the different processes required for performing the tasks in the SemTab challenge, we present the pipeline diagram seen in Figure 2. The pipeline has four general parts: data cleansing, data retrieval, CEA and CPA annotation, and CTA annotation.

In the SemTab 2022 and 2023 datasets, noise such as misspelled words or superfluous punctuation may be present. To address this issue, we preprocess and cleanse the dataset. We employ Microsoft's Azure Bing Search API to aid correct spelling errors.

The preprocessed data advances to the subsequent stage for Entity Retrieval. For each mention within the refined dataset, we query the MediaWiki Action API to obtain potential annotations for the cell. To ensure comprehensive coverage and minimise the risk of overlooking suitable entities, we employ two distinct MediaWiki Action API actions to gather a broad range of potential annotations.

Upon obtaining entities for each mention, we execute the processes of CEA and CPA. This involves determining the subject column for every dataset and then verifying if the mentions in the remaining cells of a row correspond to the properties of the entity in that row's subject column. Mentions for which we cannot confidently establish matches will be compiled into a list and passed to the Machine Learning (ML) Annotator.

After completing the CEA and CPA, we proceed to the final task, namely CTA. This involves examining each column and identifying the most frequently occurring `instance of` property among all cells in that column.

3.2. Enhancing Entity Preprocessing

We use the Bing Search API to correct misspelled mentions in Wikidata entities. Each mention is sent directly to the API with a suffix 'site:www.wikidata.org' search modifier. If a 'Did you mean' or 'Including results for' suggestion is present, we validate it against the search results. We remove the '- Wikidata' suffix from titles before calculating the Levenshtein ratio between the suggested query and each combination of the title. We use a threshold of 86% Levenshtein ratio to determine the best match. Anything below this threshold will not be processed, and the original mention will be used as is. This value was determined through an experiment using the mentions from the Validation 2022 dataset for HardTables in Round 1. By selecting a specific threshold, we can assess the accuracy of the spellchecker by comparing the suggested mention and the ground truth values available to us. Accuracy is defined as the percentage of correctly spell-checked mentions compared to the ground truth value. The results of the experiments can be seen in Figure 4.

If the ratio is below our threshold of 86% or if the Bing Search API did not provide a suggestion, we consider the best match to be the original query itself, as a lower ratio might lead to unintended and/or unrelated matches. Conversely, if the ratio is equal to or above the threshold, we utilise the result appearing in the corresponding title without the suffix '- Wikidata'. If the Bing Search API does not provide a suggestion, we follow a similar methodology, treating the initial query as the baseline. We then proceed to compare this best match with all combinations of the search result titles in an attempt to identify possible matches. If no results are available, we refrain from correction.

Our process achieved 88.90% accuracy with preprocessing against 79.87% without preprocessing on the Validation 2022 dataset for HardTables in Round 1, demonstrating its effectiveness in correcting misspelled mentions in Wikidata entities.

To illustrate, suppose we have a search query 'Tezla Model 3 car' and a title 'Tesla Model 3'. Bing suggests 'Tesla Model 3 car', and we calculate the Levenshtein ratio between the suggestion and title combinations. We obtain the following combinations and ratios, respectively, as illustrated in Figure 3. This task is computationally intensive due to the loop's iteration count, which ranges from 1 to 2^N times where N is the number of title words. This results in a time complexity of $O(2^N)$, indicating the exponential growth in computational requirements as N increases.

The best match has a ratio of 87%, indicating the most similar title. Note that this is done across all titles for that given search, and the final best match is the one that has the highest ratio across those. By selecting the title with the highest ratio, we improve the accuracy of our spell correction process, ensuring the best possible identification of Wikidata entities.

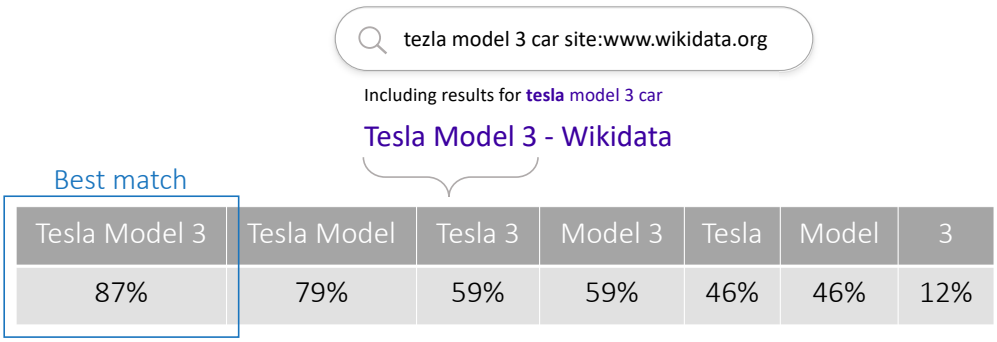

Figure 3: Visual representation of the Levenshtein Ratio (in %) comparison between 'Tesla Model 3 car' and 'Tesla Model 3'.

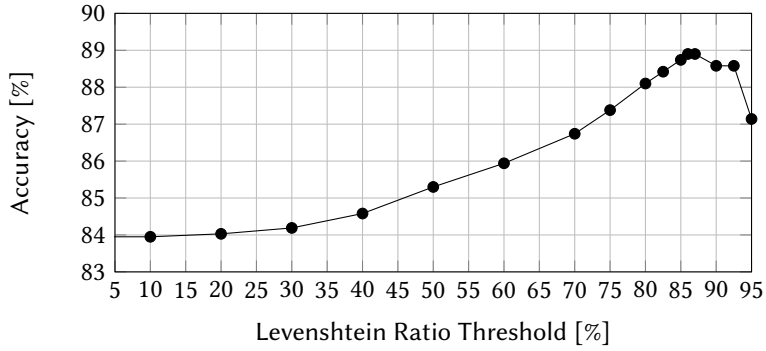

Figure 4: Results showing the best Levenshtein threshold to maximise correction accuracy.

3.3. Entity Retrieval

We now proceed to the entity retrieval phase, feeding in the preprocessed mentions. To retrieve entities for preprocessed mentions in each entity column, we query the MediaWiki Action API twice, using both the 'wbsearchentities' and 'query' actions. The following are examples of how each of the two APIs are called for the mention "Superman". https://www.wikidata.org/w/api.php?action=wbsearchentities&search=Superman&language=en, https://www.wikidata.org/w/api.php?action=query&srsearch=Superman&list=search. We incorporate both actions because we found instances where they both yield unique results. This occurs because the 'query' action attempts to match the search string with entity descriptions in addition to searching directly for entity names.

To determine which columns are entity columns, we look through the target files for SemTab. Any columns that either contain cells to be CEA annotated or are CTA targets, we label as entity columns. It should be noted that this method of "finding" entity columns can not make a wrong choice or miss an entity column, and therefore this may slightly boost the results of SemTex. The method was chosen due to time constraints on the project. As mentioned an alternative approach would be what KGCODE-Tab [2] does using NER tools.

The results from both actions are combined into a list of unique entities. An entity in this list

is a *candidate* for the given cell. We collect each candidate's *id*, *title*, *description*, and *statements*. A candidate's statements are RDF triples where the subject is the candidate itself. Statements have types depending on the information they represent. Specifically, we collect statements of types `wikibase-item`, `quantity`, `monolingualtext`, and `time`. Of these types, statements of type `wikibase-item` describe other entities, i.e., where the object is an entity. For the rest, objects are literals. With this information gathered, all candidates are passed to the CEA and CPA phase.

3.4. Cell Entity Annotation & Cell Property Annotation

To solve CEA and CPA, SemTex uses what we call the Subject Column Approach (SCA). This approach works on the assumption that most (if not all) datasets have a subject column, which is the column that is referred to by the rest of the columns. We assume that the subject column is the first column of the table, as this is the case for the round #1 WikidataTablesR1 dataset. In Figure 5, which is a subset of Figure 1, 'Superhero' is the first column and therefore the subject column.

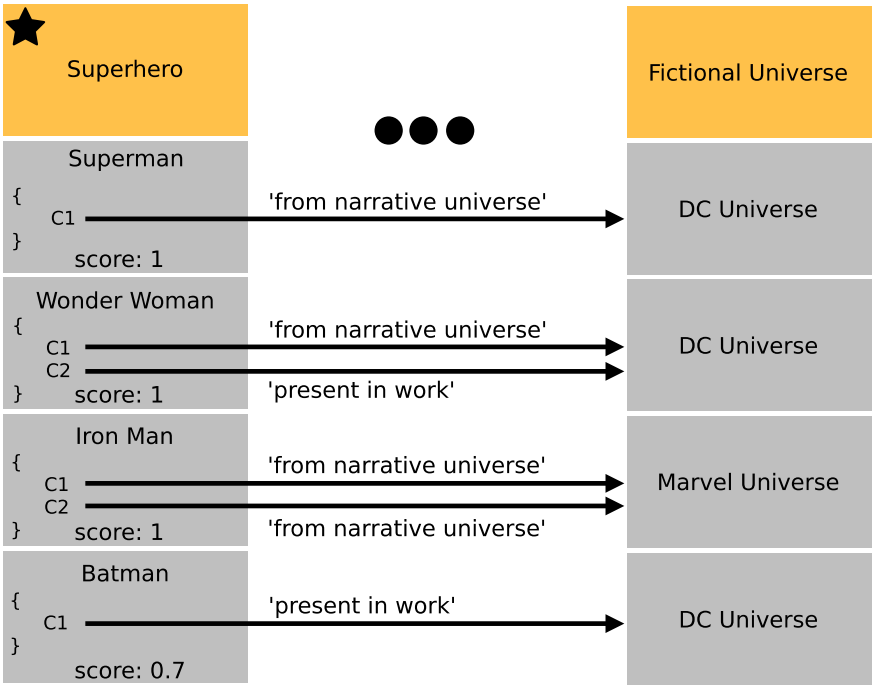

Figure 5: Example showing different candidate sets for subject column cells.

3.4.1. Cell Entity Annotation for Subject Column

The first step when performing CEA for the subject column is to score each candidate in its cells. The algorithm that scores a candidate can be seen in Algorithm 1. As seen in the for loop on line 5, we score a candidate based on the statements of that candidate. The `LevenshteinRatio` used

for `wikibase-item` and `monolingualtext` is a similarity measure between 0 and 1 that uses the Levenshtein distance between two words. `NormAbsDiff` also produces a number between 0 and 1, describing the similarity between two numbers or dates.

Algorithm 1: SemTex's candidate scoring algorithm.

Input: Candidate c, mentions M from the same row as c
Output: Score representing c's similarity to M

```
Function CandidateScore(c, M):
    score := 0
    for m ∈ M do
        bestScore := 0
        for s ∈ c.statements do
            if s.type is equal to "wikibase-item" then
                bestScore := Max(bestScore, LevenshteinRatio(s.object.title, m))
            else if s.type is equal to "quantity" then
                bestScore := Max(bestScore, NormAbsDiff(Number(m), s.object))
            else if s.type is equal to "monolingualtext" then
                bestScore := Max(bestScore, LevenshteinRatio(s.object, m))
            else if s.type is equal to "time" then
                bestScore := Max(bestScore, NormAbsDiff(Date(m), s.object))
        score := score + bestScore
    return score
```

After all candidates are scored for a cell, there are two scenarios:

1. One candidate is remaining, which is the one with the highest score.
2. Multiple candidates are remaining, all with the same score.

Consider the example seen in Figure 5, where the candidate sets and corresponding score for each cell in the subject column is illustrated. In the example, the 'Superman' and 'Batman' cells fall into scenario 1, whereas the two others fall into scenario 2. In the first scenario, we identified our best estimate for the correct candidate and selected it accordingly. In the second scenario, our candidate selection is based on the assumption that the correct candidate will possess a property that all the other cells in the subject column also have. Therefore, we examine each candidate set for each cell that falls into scenario 2. We then keep track of the number of occurrences for each property used to designate a candidate during the scoring process. In the example, 'from narrative universe' has three occurrences throughout the column, whereas 'present in work' only has two.

Afterwards, we look among the candidate sets containing two or more candidates and for each of those, we select the candidate with the most occurring property. For example, in the 'Wonder Woman' cell the candidate $C1$ would be picked. If no candidate possesses the aforementioned property, we proceed to look for a candidate that has the property with the next highest count,

and so on. If multiple candidates have the property with the most occurrences, we submit the candidate set to the Machine Learning (ML) algorithm introduced in Section 3.5, which will then make the selection. For example, in the 'Iron Man' cell we will submit both $C1$ and $C2$ to the ML algorithm.

If necessary, instead of presuming that the first column is the subject column, we can determine it by computing an accumulating score as seen in line 14 Algorithm 1 for *each* column instead of only the subject column. The column with the highest total score will be considered the subject column. The subject column is given a higher score because its statements include the mentions from the other cells in the same row, but not necessarily vice versa.

3.4.2. Cell Property Annotation

At this stage, we have selected a single candidate for every cell in the subject column. To solve CPA, we examine each of these candidates and determine which property is assigned to each cell in the corresponding row. By tallying the properties on a column basis, we identify the property with the highest count as the CPA prediction for each respective subject column-column pair. Note that for each cell, duplicate properties are only counted once. Consider again the example on Figure 5. Here, the property 'from narrative universe' is chosen as the CPA between the columns with a count of 3.

3.4.3. Cell Entity Annotation for Non-Subject Columns

By now, we have successfully carried out CPA for the entire table and performed CEA for the subject column. To do CEA for the non-subject columns, we examine each selected candidate from the subject column and compare its properties with the CPA prediction for the other columns. Suppose the candidate has the exact properties indicated by the CPA prediction. In that case, we have confidence in this candidate and therefore choose the entity objects the candidate's properties point to for the respective row. For example, the CEA prediction for the 'Fictional Universe' column in the 'Superman' row is the entity pointed to by the 'from narrative universe' property on $C1$. However, if the chosen candidate in the subject column lacks any of the properties from the CPA prediction, all candidates for each non-subject cell in the row will be passed to the Machine Learning algorithm, where a CEA prediction will be made for each cell. For example, since the 'Batman' candidate was chosen based on the 'present in work' property, the CEA for the corresponding 'Fictional Universe' cell will be chosen by the ML algorithm.

3.5. Applying Gradient Boosting for Improved Annotation

We implement gradient boosting using the Catboost algorithm [3] for enhanced entity annotation in cases of multi-candidate ambiguity for a regression problem. Gradient boosting, a machine learning method, incrementally builds decision trees or 'weak learners', each correcting errors of the prior. We adopt the Catboost variant, which offers several key optimisations such as ordered boosting, which mitigates overfitting by estimating gradient statistics utilizing only a randomly-selected portion of preceding data. Additionally, Catboost is efficient in managing high-cardinality textual data, an attribute frequently encountered in Wikidata entities, thus

Algorithm 2: SemTex's instance overlap feature extraction algorithm

Input: Candidate c, Other Candidates ocs
Output: Normalised number of instance overlaps of c over ocs

Function `InstanceOverlap`(c, ocs)**:**
  $overlap := 0$
  $total := 0$
  for $oc \in ocs$ **do**
    if *c.id is equal to oc.id* **then**
      Skip to next iteration
    $overlap := overlap + |c.instances \bigcap oc.instances|$
    $total := total + |oc.instances|$
  if $total > 0$ **then**
    $overlap := \frac{overlap}{total}$
  else
    $overlap := 0$
  return $overlap$

signifying its optimality for our task. Another noteworthy feature is its utilization of oblivious trees which contributes to variance minimization, overfitting reduction, and overall model stability. Consequently, Catboost assumes the role of our classifier, computing class probabilities for each potential entity match, with the candidate having the highest score designated as the correct entity.

Our model operates on 17 features, divided into four textual and 13 numerical inputs. Table 1 provides an overview of these features. For instance, the textual feature "Tag" is realized through Flair [4], a sophisticated Natural Language Processing (NLP) model enabling us to leverage state-of-the-art techniques like Named Entity Recognition (NER). To encapsulate the semantics of the candidate, our model is fed with an ample and contextual text corpus, facilitating it to predict the best-suited entity from the given 18 tags. The model is trained on the Validation 2022 and Validation 2023 datasets.

We focus on three elements of a Wikidata entity: Title, Description, and the titles of all 'instance of' statements. We integrate these elements into a unified sentence and feed it to the NER model. The most recurring tag is then selected as the feature value. However, in scenarios of equal tag frequency, the first tag is chosen, typically corresponding to the entity's title. Consider the sentence 'Cristiano Ronaldo. Portuguese footballer (born 1985). Human.'. Post-processing with Flair yields three tags: 'Cristiano Ronaldo' as PERSON, 'Portuguese' as NORP, and '1985' as a DATE. Consequently, the entity 'PERSON' becomes the feature value.

Five features in our model capitalize on the overlap principle, aiming to uncover the correct candidates via shared terms identification. Algorithm 2 shows the pseudocode to compute the instance overlap for a given candidate. Similar algorithms serve for other overlap computations.

Catboost also provides feature importance which quantifies the contribution of each feature

Table 1
Overview of the features used for machine learning.

Name	Type	Importance	Name	Type	Importance
Title Levenshtein	Numerical	31.94%	Claims Overlap	Numerical	1.44%
Description	Textual	20.14%	Tag Overlap	Numerical	1.38%
Instance Ids	Textual	16.76%	Tag	Textual	1.15%
Title	Textual	9.05%	#Words In Title	Numerical	0.91%
Instance Overlap	Numerical	6.09%	#Instances	Numerical	0.76%
Id	Numerical	2.76%	#Words In Description	Numerical	0.46%
Title Length	Numerical	2.44%	Description Length	Numerical	0.43%
#Statements	Numerical	2.04%	Subclass Overlap	Numerical	0.15%
Description Overlap	Numerical	2.04%			

towards model performance. Table 1 lists all 17 features along with their feature importance. The four most influential features are: Title Levenshtein (Levenshtein ratio between the preprocessed mention and Wikidata entity title), Description, Instance Ids (a concatenated text of 'instance of' object IDs), and Title. Interestingly, the latter three are textual features, indicating the possibility of enriching the model further with similar textual features.

3.6. Column Type Annotation

After obtaining the results from CEA, we can generate the type annotations for entity columns. This means that the results for CTA depend on the results for CEA. The type annotations are found by finding the largest overlap of the `instance of` property between all of the candidates from the CEA results for a given column in a table. Once the `instance of` object with the largest overlap has been found, it is selected as the column type annotation for the given column. In case two or more objects have the same amount of overlap within the column, the object which was seen first is picked as the column type annotation.

4. Results

We benchmark our approach on the test set of tables from SemTab. We split our approach into three strategies; `No ML`, `Only ML`, and `With ML`. `Only ML` *only* uses gradient boosting to choose the best candidate for every cell. Vice versa, `No ML` corresponds to the Subject Column Approach, but in place of employing gradient boosting to decide between multiple candidates, `No ML` takes an arbitrary choice instead. Lastly, `With ML` is the Subject Column Approach combined with the ML model.

The F1-scores for the three strategies on both validation and test sets from SemTab 2023 are shown in Table 2. Additionally, we have included results from the 2022 Test to compare the performance of SemTex against the SemTab 2022 tools. We use the SemTab 2022 dataset to evaluate our three strategies, and select the best performing for the SemTab 2023 challenge. Note that since the ML model only performs CEA, no CPA predictions can be made for `Only ML`. Generally, the results on the SemTab 2022 dataset shows that the `With ML` strategy outperforms the two other strategies by a noteworthy margin across all tasks. CPA sees the smallest gain,

Table 2
Round 1 benchmark results of the WikidataTables datasets for the 2022 test and 2023 validation and test sets from SemTab.

	Test 2022 (3691 tables)			**Validation 2023** (500 tables)			**Test 2023** (9917 tables)		
F1-score	No ML	Only ML	W. ML	No ML	Only ML	W. ML	No ML	Only ML	W. ML
CEA	86.91%	84.60%	91.01%	83.13%	82.23%	91.03%			**88.5%**
CTA	92.32%	79.96%	93.85%	92.34%	91.52%	95.35%			**93.4%**
CPA	96.77%		97.05%	97.81%		98.24%			**96.4%**

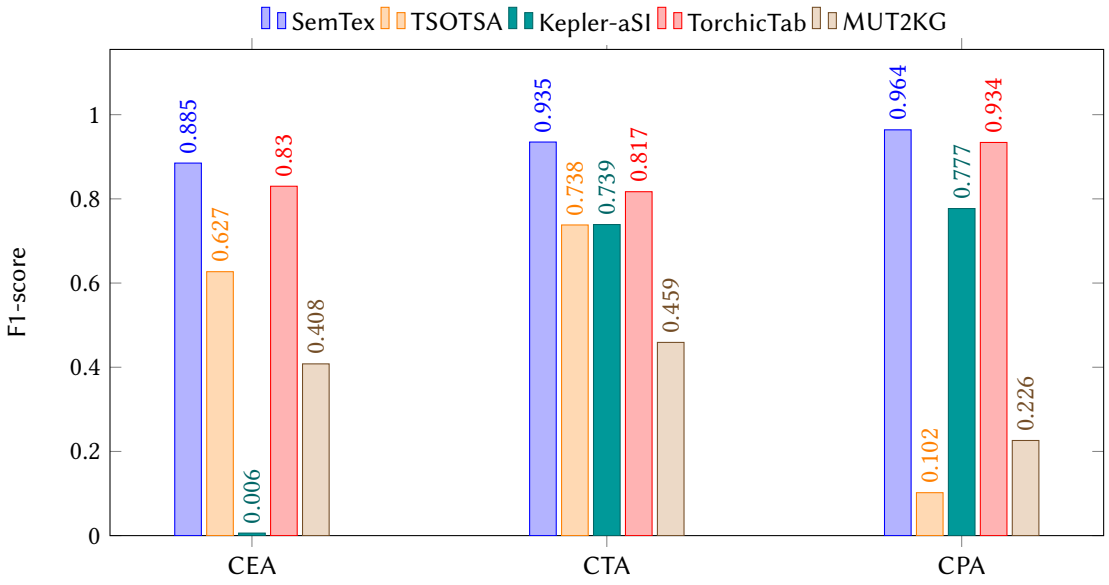

Figure 6: Comparison of Round 1 WikidataTablesR1 dataset results of SemTex and SemTab 2023 participants. Participants are shown in the same order as the legend.

which is reasonable since ML only has a negligible impact on these predictions, whileCEA is improved significantly by the inclusion of ML. We note on the other hand that the `Only ML` strategy consistently perform the worst of all alternatives. However, we highlight that because the ML Model was trained on the validation set, the high scores on validation set are not representative. Therefore, the test set results better indicate our approach's capabilities. Overall, it is clear that the inclusion of the ML model in the annotation process is beneficial, and as a consequence we only evaluate the `With ML` strategy on the SemTab 2023 Test dataset.

The test set is the basis of evaluation for SemTab. Figure 6 compares our results to the official results of the SemTab 2023 tools. The other tools participating in the challenge are TSOTSA, Kepler-aSI, TorchicTab and MUT2KG. SemTex outperform the other participants across all three tasks, as we achieve the highest F1-scores with 88.5%, 93.4% and 96,4% in CEA, CTA and CPA, respectively for the WikidataTablesR1 dataset.

5. Conclusion and Future Works

In this paper, we introduced SemTex, a tool for Semantic Table Interpretation. We employ a unique hybrid approach for annotating cell entities, combining relationships analysis in knowledge graphs with Gradient Boosting for candidate disambiguation. The empirical results on the SemTab challenge demonstrated the efficacy and robust performance of this novel strategy. Although this approach is promising, we believe that further improvements can be made by an additional phase for reviewing candidates that score low in confidence and rerouting these for further examination. This would enhance the overall accuracy and reliability of the annotation process. Other approaches employ Knowledge Graph Embedding, this would be another interesting research avenue in the context of the SemTab challenge, as those approaches have proven considerable value in other fields of knowledge management and processing. As example, the use of embedding have shown promising results in applications such as link prediction and entity recognition. Additionally, the use of approaches based on Large Language Models (LLM) could be envisioned during the spell correction phase. This would potentially provide a robust alternative to the use of the Levenshtein distance, and reduce the reliance on external services such as Bing. Future works on SemTex could involve the development of techniques to enhance the CTA phase by descending through the ancestor tree of the candidate elements. Lastly, we noted significant positive impacts from textual features, suggesting the possibility of integrating more of these in future iterations of SemTex.

References

[1] M. Cremaschi, R. Avogadro, D. Chieregato, s-elbat: A semantic interpretation approach for messy table-s, in: V. Efthymiou, E. Jiménez-Ruiz, J. Chen, V. Cutrona, O. Hassanzadeh, J. Sequeda, K. Srinivas, N. Abdelmageed, M. Hulsebos (Eds.), Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab 2021, co-located with the 21st International Semantic Web Conference, ISWC 2022, Virtual conference, October 23-27, 2022, volume 3320 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 59–71. URL: https://ceur-ws.org/Vol-3320/paper6.pdf.

[2] X. Li, S. Wang, W. Zhou, G. Zhang, C. Jiang, T. Hong, P. Wang, Kgcode-tab results for semtab 2022, in: V. Efthymiou, E. Jiménez-Ruiz, J. Chen, V. Cutrona, O. Hassanzadeh, J. Sequeda, K. Srinivas, N. Abdelmageed, M. Hulsebos (Eds.), Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab 2021, co-located with the 21st International Semantic Web Conference, ISWC 2022, Virtual conference, October 23-27, 2022, volume 3320 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 37–44. URL: https://ceur-ws.org/Vol-3320/paper5.pdf.

[3] A. V. Dorogush, V. Ershov, A. Gulin, Catboost: gradient boosting with categorical features support, CoRR abs/1810.11363 (2018). URL: http://arxiv.org/abs/1810.11363. arXiv:1810.11363.

[4] S. Schweter, A. Akbik, FLERT: document-level features for named entity recognition, CoRR abs/2011.06993 (2020). URL: https://arxiv.org/abs/2011.06993. arXiv:2011.06993.