Transformer-based Subject Entity Detection in Wikipedia Listings

Nicolas Heist[1,*], Heiko Paulheim[1]

[1]*Data and Web Science Group, University of Mannheim, Germany*

Abstract

In tasks like question answering or text summarisation, it is essential to have background knowledge about the relevant entities. The information about entities - and in particular, about long-tail or emerging entities - in publicly available knowledge graphs like DBpedia or CaLiGraph is far from complete. In this paper, we present an approach that exploits the semi-structured nature of listings (like enumerations and tables) to identify the main entities of the listing items (i.e., of entries and rows). These entities, which we call *subject entities*, can be used to increase the coverage of knowledge graphs. Our approach uses a transformer network to identify subject entities on token-level and surpasses an existing approach in terms of performance while being bound by fewer limitations. Due to a flexible input format, it is applicable to any kind of listing and is, unlike prior work, not dependent on entity boundaries as input. We demonstrate our approach by applying it to the complete Wikipedia corpus and extract 40 million mentions of subject entities with an estimated precision of 71% and recall of 77%. The results are incorporated in the most recent version of CaLiGraph.

Keywords

Subject Entity Detection, Named Entity Recognition, Wikipedia Listings, CaLiGraph

1. Introduction

1.1. Motivation

Background knowledge provides an essential advantage in tasks like text summarisation or question answering. With ready-to-use entity linking tools like Falcon [1], entities in text can be identified and additional information can be drawn from background knowledge graphs (e.g. DBpedia [2] or CaLiGraph[1] [3]). Of course, this is only possible if the necessary information about the entity is included in the knowledge graph [4].

Hence, it is important to equip knowledge graphs with as much entity knowledge as possible. While this is easily possible for prominent entities that are mentioned frequently, the retrieval of information about long-tail and emerging entities that are mentioned only very infrequently is tedious [5]. Still, approaches for automatic information extraction can be applied to increase

ISWC 2022: Deep Learning for Knowledge Graphs, October 23–27, 2022, Virtual Conference

*Corresponding author.
nico@informatik.uni-mannheim.de (N. Heist); heiko@informatik.uni-mannheim.de (H. Paulheim)
http://www.uni-mannheim.de/dws/people/researchers/phd-students/nicolas-heist/ (N. Heist);
http://www.heikopaulheim.com/ (H. Paulheim)
0000-0002-4354-9138 (N. Heist); 0000-0002-4354-9138 (H. Paulheim)

CEUR Workshop Proceedings (CEUR-WS.org)

[1] http://caligraph.org

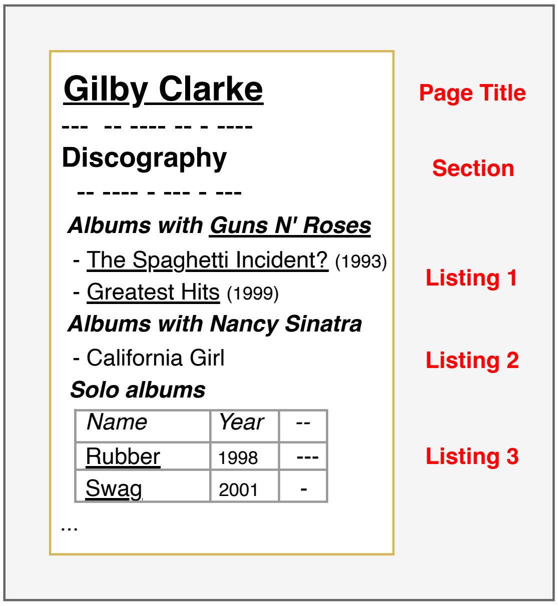


Figure 1: Simplified view on the listings of the Wikipedia page of Gilby Clarke.

the coverage of knowledge graphs to a certain extent. One strand of research is concerned with open information extraction systems that try to extract facts from web text (e.g. [6, 7]). While they perform strongly on well-known entities, the extraction quality for long-tail entities is considerably worse [6].

The extraction of information from semi-structured data is in general less error-prone and has already proven to yield high-quality results as, for example, DBpedia itself is extracted primarily from Wikipedia infoboxes; other approaches use the category system of Wikipedia [8, 9, 10]; many more approaches focus on tables (in Wikipedia or the web) as a semi-structured data source to extract entities and relations (see [11] for a comprehensive survey).

In this work, we generalize over structures like enumerations (*Listings 1 and 2*) and tables (*Listing 3* in Figure 1) by simply considering them as listings with listing items (i.e., enumeration entries or table rows). Further, we call the main entity, that a listing item is about, a *subject entity* (SE). In previous work, we defined SEs as *all entities in a listing appearing as instances to a common concept* [12]. In case of Figure 1, the SEs are the mentioned albums (e.g. *The Spaghetti Incident?* or *California Girl*). Here, the common concept is made explicit through the section labels above the listings (*Albums with..*), but it may as well be the case that it is only implicitly defined through the respective SEs. As a listing item typically mentions only one SE together with some context (in this case, the publication year of the album), we assume that at most one SE per listing item exists.

In the English Wikipedia chapter alone, we find almost five million listings in roughly two million articles. From our estimation, about 80% of the listings are suitable for the extraction of SEs, bearing an immense potential for knowledge graph completion (for details, see Section 3.1). Upon extraction, they can easily be digested by downstream applications: Due to the semi-structured nature of listings, the quality of extraction is higher than extraction from plain text, and SEs are typically extracted in groups of instances sharing a common concept (as given by the definition above). Especially the latter point makes subsequent disambiguation step much easier, as the group of extracted instances provides context for every individual instance. Another example of the downstream use of SEs is a work of ours where we used groups of SEs to learn lexical patterns that entail axioms [12]. For example, if a listing is in a section that starts with *Albums with*, we learn that the SEs are of the type *Album*.

The combination of these two ideas, i.e. of extracting novel SEs and learning defining axioms for them, can bring a big benefit. In Figure 1, instead of simply discovering *California Girl* as a new entity, we additionally assign the type *Album*. Thinking further, we can learn an axiom that all albums mentioned in the discography of *Gilby Clarke* are albums that are authored by him. The additional information can be used to refine the description of the extracted entity in the knowledge graph.

1.2. Problem Statement

Given an arbitrary listing, we want to identify the SEs among all entities mentioned in the listing. In the literature, there are only very few approaches that deal with this problem. The most related approach is a previous work of the authors that is concerned with the detection of SEs in Wikipedia list pages [3].[2] The approach uses a hand-crafted set of features to classify entities in tables or enumerations of list pages as SEs. However, the approach has several limitations:

- It is only applicable to list pages and not to listings in any other context as the features are primarily designed for the list page context.
- Dependencies between individual SEs of listing items are not taken into account as the classification is done separately for every item.
- The approach needs mention boundaries of entities as input for the classification. Consequently, it cannot identify any new entities but only categorize existing entities into subject and non-subject entities.

1.3. Contributions

To harness the information expressed through SEs in more general settings, we aim to overcome the previously mentioned limitations in this work. In particular, we make the following contributions:

- We present a Transformer-based approach for SE detection with a flexible input format that allows us to apply it to any kind of listing. Further, the model takes dependencies between listing items into account (Section 4.1).
- During prediction, the approach detects SEs end-to-end without relying on mention boundaries of the entities in the input sequence (Section 4.2).
- We introduce a novel mechanism for generating negative samples of listings (Section 4.3) and a fine-tuning mechanism on noisy listing labels (Section 4.4) leading to more accurate prediction results.

[2] List pages are special Wikipedia pages that contain only listings describing entities of a certain topic.

- In our evaluation, we show that the performance of our approach is superior to previous work (Section 5.3); further, we analyse its performance in a more general scenario - that is, arbitrary listings of Wikipedia pages (Section 5.4).
- We run the extraction of SEs on the complete Wikipedia corpus and incorporate the results in a new version of CaLiGraph (Section 5.6).

The produced code is publicly available and part of the CaLiGraph extraction framework.[3]

2. Related Work

With the presented approach we detect SEs end-to-end, directly from listing text. For a given listing, we identify mentions of named entities and decide at the same time whether they are SEs of a listing or not. In the following, we first review Named Entity Recognition (NER) and subsequently discuss approaches that detect SEs.

2.1. Named Entity Recognition

NER is a subproblem of Entity Linking (EL) which only tries to identify mentions of named entities in the text without actually disambiguating them [13]. As opposed to general Entity Recognition, NER only deals with the identification of named entities and ignores the linking of concepts (also called Wikification) [14].

Early NER systems were based on hand-crafted rules and lexicons, followed by systems using feature-engineering and machine learning [15]. One of the first competitive NER systems that used neural networks has been presented by Collobert et al. in 2011 [16]. This eventually lead to more sophisticated architectures based on word embeddings and LSTMs (e.g. from Lample et al. [17]).

With the rise of transformer networks [18] like BERT [19] in 2018, they also found their direct application in NER (e.g. by Liang et al. [20]), or as part of an end-to-end EL system like the one from Broscheit [21]. The latter uses a simple but effective prediction scheme, where entities are predicted at token-level and multiple subsequent tokens with the same predicted entity are collapsed into the actual entity prediction. In our work, we use a similar token-level prediction scheme to detect SEs.

2.2. Subject Entity Detection

Although SE detection has not explicitly been addressed in the literature very frequently, there are some approaches that deal with related problems or subproblems of it. In table interpretation, an important task is the identification of the *subject column*, i.e. the column containing the entity with outgoing relations to all other columns. TAIPAN [22] is an approach that aims to recover the semantics of tables and names subject column identification as the first major task towards relation extraction in tables. To identify subject columns, they choose the columns having entities with the most outgoing edges to entities in other columns w.r.t. a background knowledge graph. While this is a viable approach for tables that are already annotated with

[3] https://github.com/nheist/CaLiGraph

entities, it is not broadly applicable to general listings that may not have many known (or even annotated) entities.

Another related approach is from Zhao et al. [23] who deal with a problem which they call *key entity detection*. Primarily, they do sentiment analysis in financial texts and use the detection of key entities - which they define to be subjects of events related to financial information - in order to attribute the positive or negative sentiment to a concrete entity. Similar to our proposed approach, they use a Transformer to detect key entities. However, they only use it to select the key entities from a predefined set of entities and ignore the NER part.

As mentioned in the introduction, the most closely related approach is the authors' prior work [3]: using manually defined features and a binary XGBoost classifier, entities on list pages are classified into either subject entities or non-subject entities. For the page *List of Japanese speculative fiction writers*,[4] for example, all entities in the enumerations that are *Japanese speculative fiction writers* are classified as SEs.

More concretely, the approach uses page features (e.g. number of sections or tables on the page), positional features (e.g. indentation level of entry in the enumeration), and linguistic features (e.g. whether the column header is synonymous with the list page title). Overall, SEs are extracted with a precision of 90% and a recall of 67%. The classifier is trained and evaluated with a set of list pages that are annotated through distant supervision with DBpedia for background knowledge. This part is discussed in detail in Section 3.2 as the approach presented here relies on this training data generation strategy as well.

3. Preliminaries

3.1. Listings in Wikipedia

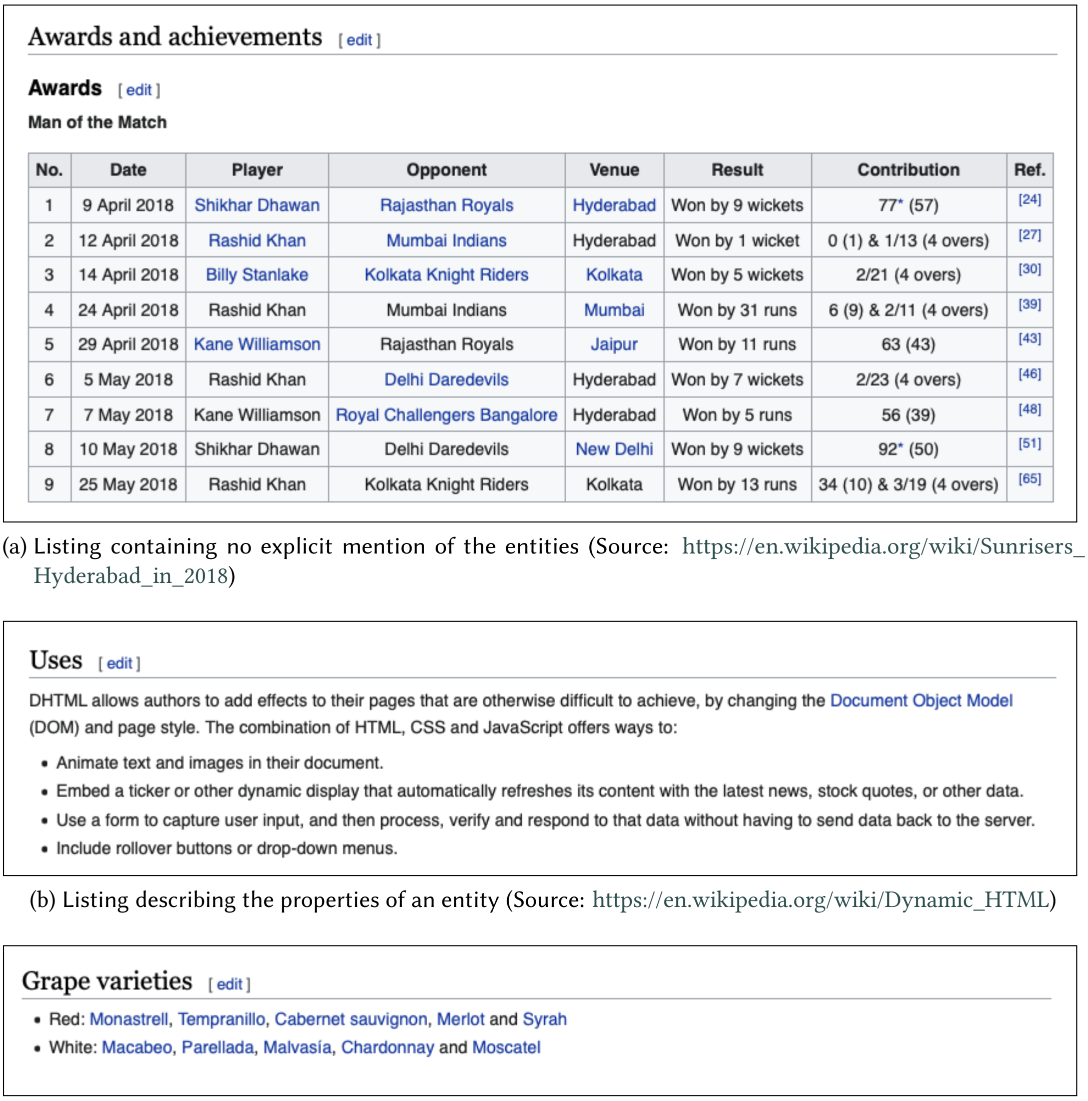
Overall, the English Wikipedia has more than five million articles. Roughly two million of them contain at least one listing in the form of an enumeration or a table. All over these pages, we find 3.5 million enumerations and 1.4 million tables.[5] The roughly 90K list pages in Wikipedia contain the most structured and easily exploitable form of listings. Here, listings are almost exclusively used to list a number of entities that have some common property (e.g. all Japanese speculative fiction writers).

Listings that appear on other Wikipedia pages are used for this purpose as well but not exclusively, which makes the detection of SEs much more complex. From the inspection of a sample of Wikipedia listings, we estimate that approximately 85% of enumerations and 67% of tables are usable for our approach. Especially enumerations are often used to simply structure content (e.g. to list the individual episodes in a biography). But even if listings are used to describe entities, they may not be usable due to various reasons:

- Entity description without explicit mention (example in Figure 2a)
- Description of the properties of a single entity (example in Figure 2b)
- Listing items contain groups of entities (example in Figure 2c)

[4] https://en.wikipedia.org/wiki/List_of_Japanese_speculative_fiction_writers

[5] These numbers exclude very small listings with less than three items, which we do not consider.



Awards and achievements [edit]

Awards [edit]

Man of the Match

No.	Date	Player	Opponent	Venue	Result	Contribution	Ref.
1	9 April 2018	Shikhar Dhawan	Rajasthan Royals	Hyderabad	Won by 9 wickets	77* (57)	[24]
2	12 April 2018	Rashid Khan	Mumbai Indians	Hyderabad	Won by 1 wicket	0 (1) & 1/13 (4 overs)	[27]
3	14 April 2018	Billy Stanlake	Kolkata Knight Riders	Kolkata	Won by 5 wickets	2/21 (4 overs)	[30]
4	24 April 2018	Rashid Khan	Mumbai Indians	Mumbai	Won by 31 runs	6 (9) & 2/11 (4 overs)	[39]
5	29 April 2018	Kane Williamson	Rajasthan Royals	Jaipur	Won by 11 runs	63 (43)	[43]
6	5 May 2018	Rashid Khan	Delhi Daredevils	Hyderabad	Won by 7 wickets	2/23 (4 overs)	[46]
7	7 May 2018	Kane Williamson	Royal Challengers Bangalore	Hyderabad	Won by 5 runs	56 (39)	[48]
8	10 May 2018	Shikhar Dhawan	Delhi Daredevils	New Delhi	Won by 9 wickets	92* (50)	[51]
9	25 May 2018	Rashid Khan	Kolkata Knight Riders	Kolkata	Won by 13 runs	34 (10) & 3/19 (4 overs)	[65]

(a) Listing containing no explicit mention of the entities (Source: https://en.wikipedia.org/wiki/Sunrisers_Hyderabad_in_2018)

Uses [edit]

DHTML allows authors to add effects to their pages that are otherwise difficult to achieve, by changing the Document Object Model (DOM) and page style. The combination of HTML, CSS and JavaScript offers ways to:

- Animate text and images in their document.
- Embed a ticker or other dynamic display that automatically refreshes its content with the latest news, stock quotes, or other data.
- Use a form to capture user input, and then process, verify and respond to that data without having to send data back to the server.
- Include rollover buttons or drop-down menus.

(b) Listing describing the properties of an entity (Source: https://en.wikipedia.org/wiki/Dynamic_HTML)

Grape varieties [edit]

- Red: Monastrell, Tempranillo, Cabernet sauvignon, Merlot and Syrah
- White: Macabeo, Parellada, Malvasía, Chardonnay and Moscatel

(c) Listing containing groups of entities (Source: https://en.wikipedia.org/wiki/Ibiza_(Vino_de_la_Tierra))

Figure 2: Examples of Wikipedia page listings with layout or content that is challenging for SE detection.

Especially the first point renders a big portion of tables useless for our approach as an entity is implicitly described through entities and literals mentioned in multiple table columns (e.g. a sports match is described through date, player, opponent, and result).

3.2. Distantly-Supervised Training Data Generation for List Pages

In our experiments, we will use the training data generation strategy that we introduced in previous work [3]; to make this paper self-contained, we will give an overview of this strategy here. The strategy is based on the observation that DBpedia classes, Wikipedia categories, and Wikipedia list pages can be transformed into an immense taxonomy through linguistic and statistical methods. For example, the taxonomy contains the hierarchy *Person > Writer > Speculative fiction writer > Japanese speculative fiction writer.* The first two elements originate from DBpedia classes, the third from a category, and the last from a list page.

As a consequence, we can use this hierarchy to infer the DBpedia classes of SEs for many list pages. To label the list page *List of Japanese speculative fiction writers*, we assign every entity with the DBpedia class *Writer* a positive label and every entity with a class that is disjoint with *Writer* a negative label. Then we include all listing items into our training set that either have an entity with a positive label or only entities with negative labels. Other listing items are ignored as we cannot be certain that they may contain SEs which we could not identify due to the incompleteness of DBpedia.

The knowledge graph CaLiGraph [9, 3] uses this extended taxonomy of DBpedia classes, categories, and list pages as a type hierarchy, and enriches the original DBpedia instances with additional, more fine-grained types. Furthermore, CaLiGraph contains a higher number of instances than DBpedia as it additionally contains the extracted SEs from list pages.

3.3. Transformers for Token Classification

Pre-trained transformer networks [18] like BERT [19] or DistilBERT [24] produced new state-of-the-art results for various NLP tasks including NER and question answering. To a large extent, their ubiquitous application is due to the fact that only a comparably small amount of fine-tuning is necessary to fit them to various tasks. BERT, for instance, consists of 12 multi-head attention layers followed by a simple linear layer as classification head. To apply a transformer model to a token classification problem, it is oftentimes sufficient to fine-tune the final classification head.

The input for a transformer model can consist of plain text and needs to be tokenized before it can be processed. Every word in the input sequence is transformed into one or more tokens (if the word is not contained in the vocabulary, multiple word-piece tokens are used). Further, the input sequence has to contain special tokens that indicate, for example, the start and the end of the sequence. Using BERT for token classification, the input sequence has a fixed length of 512 tokens, has to start with a *[CLS]* token and end with a *[SEP]* token. Additional special tokens may be introduced to provide more context information to the model.

4. Subject Entity Detection with Transformers

To detect SEs in listings, we phrase the problem as a token classification problem where we, similar to the work of Broscheit [21], produce a label for every token of the input sequence. In a subsequent step, we aggregate the token labels to predictions of SE mentions. We use 13 different token labels, such as *Person* or *Organisation*, to identify SEs and additionally make a prediction of their types (refer to Table 5 for the full list of labels). In Section 4.1 we explain how to create input sequences that preserve the context and the structure of a listing. In Section 4.2 we show our choice of labels for SE prediction, and in Section 4.3 we introduce a mechanism to generate negative samples of listings. Finally, Section 4.4 explains how to use noisy SE labels on page listings for further fine-tuning of our models.

4.1. Token-level Subject Entity Detection

To pass a listing for SE detection to the transformer model, we use multiple special tokens in order to encode context information (page, section, potential table header) and structural information (entries, rows, columns) of the listing into the input sequence. Every sequence consists of the listing context, followed by the special token indicating the end of context *[CXE]*, and one or more listing items:

```
[CLS] <context> [CXE] <listing items> [SEP]
```

We use the special token *[CXS]* to separate context elements. Within listing items, table rows and columns are indicated with *[ROW]* and *[COL]*, respectively. For enumerations, we use the tokens *[E1]* to *[En]* to indicate the start of an entry with the indentation level 1 to n.

Ignoring that some words may be split into multiple tokens, the input for the first listing item of *Listing 1* in Figure 1 looks as follows:

```
[CLS] Gilby Clarke [CXS] Discography [CXS] Albums with Guns N' Roses [CXE]
[E1] The Spaghetti Incident? (1993) [SEP]
```

We want the model to take dependencies between listing entities into account. For example, if the SE in the first listing item is mentioned right in the beginning, it is very likely that this is the case for the remaining listing items as well. Instead of only providing one listing item per input sequence, we can provide as many as the input sequence length permits. Through the attention layers within the Transformer architecture, the model is able to take these dependencies within the input sequence into account. Hence, we put *Listing 1* into one input sequence:

```
[CLS] Gilby Clarke [CXS] Discography [CXS] Albums with Guns N' Roses [CXE]
[E1] The Spaghetti Incident? (1993)
[E1] Greatest Hits (1999) [SEP]
```

Likewise, we encode *Listing 3* as one input sequence:

```
[CLS] Gilby Clarke [CXS] Discography [CXS] Solo albums [CXS]
[ROW] Name [COL] Year [CXE]
[ROW] Rubber [COL] 1998
[ROW] Swag [COL] 2001 [SEP]
```

If the listing is too long to fit into one input sequence, we split the listing items into chunks and process them one after another. Each chunk is augmented with the same context information and a different set of listing items. Depending on the length of listing items, it is possible to fit 20 or more items into one input sequence. In our ablation study in Section 5.5 we show that this item chunking strategy has a strongly positive effect on the recall of the model. But apart from that we immensely reduce the run time of the model for training and prediction. The number of processed input sequences is reduced by a factor that is roughly equivalent to the median number of items per listing.[6]

[6] We deliberately use the median and not the average of items per listing as large listings will be split into multiple input sequences due to the size limitation.

4.2. Coarse-grained Entity Type Prediction

The most common notation to tag tokens in NER is the BIO notation (*Begin*, *Inside*, and *Outside* of an entity) together with an entity type (e.g. *Person* or *Organisation*). We decided not to use the BIO notation as, per definition, there is at most one SE per listing item. Instead of making the task even simpler and getting rid of the entity type prediction in favor of a simple binary SE prediction task as well, we decided to stick with the coarse-grained entity type prediction. This has the advantage that the entity types can be used as additional information in downstream tasks - most importantly in a subsequent entity disambiguation step. In addition to that, we show in our ablation study in Section 5.5 that the more difficult task of entity type prediction even slightly increases the precision of the model.

Context and special tokens are annotated with the *IGNORE* label to indicate the model that we need no prediction for these tokens. SEs are annotated with the respective entity type, everything else is annotated with *NONE*. Again ignoring word-piece tokenization, the labels for *Listing 1* of Figure 1 look as follows:

```
    IGNORE IGNORE IGNORE IGNORE IGNORE IGNORE IGNORE IGNORE IGNORE IGNORE IGNORE IGNORE
    IGNORE WORK_OF_ART WORK_OF_ART WORK_OF_ART NONE
    IGNORE WORK_OF_ART WORK_OF_ART NONE IGNORE
```

4.3. Negative Sampling through Shuffled Listings

It is difficult to find negative examples of complete listings if the training data is generated heuristically and with distant supervision as described in Section 3.2. Positives can be found easily (i.e., there is an entity in the listing item that has the correct type), but the inverse does not always hold. If we do not find a positive, this may mean that the listing item does not contain one, but it is as well possible that the annotation is missing. From a logical standpoint, it is even unlikely that some items in a listing contain SEs while others do not.

To mitigate this problem, we equipped our approach with a sampling mechanism for negatives that randomly assembles them from the contexts and items of all positives in the training set. If the context and items are assembled randomly, the differences between the individual items (and the difference in the context) should be higher than in a real listing. The intention of this mechanism is that the model learns to identify the coherence between SEs of listing items as well as between items and the context.

For enumeration listings, the mechanism is simple as we pick the context from one listing and a random number of items (between three and the maximum number of items per chunk) from other listings. For table listings, we have to take care that the number of columns of an assembled listing is consistent. Hence, the positives from the training set are divided into groups of the same column size and listings are only assembled from within a single group. A negative example produced from four different listings could look as follows:

```
[CLS] Gilby Clarke [CXS] Discography [CXS] Albums with Guns N' Roses [CXE]
[E1] James Stewart as Billy Jim Hawkins
[E1] Curzon Mill Company, part of Ashton syndicate.
[E1] Brepholoxa Van Duzee, 1904 [SEP]
```

The mechanism has exactly one hyper-parameter which is the proportion of negative listings to generate. We experiment with values between 0.0 (no negative samples at all) and 1.0 (as many negatives as we have positives).

4.4. Fine-Tuning on Noisy Page Labels

The training data generation strategy described in Section 3.2 lets us create labels for listings of list pages that we use for the initial training of our models. To train a model that works well on listings of any pages, additional training data of listings that are not on list pages may be beneficial (the differences in listings have been described in Section 3.1).

We gather this data by first training a model using the heuristically labelled list pages. We apply the model to listings of all pages for noisy labels of SEs. We then filter them by discarding any listings where multiple types of SEs have been predicted (e.g., if the first SE of a listing is labelled as *PERSON* and the second is labelled as *WORK_OF_ART*).

5. Experiments

The goal of our experiments is to compare the performance of our approach against previous work on SE detection in list pages (Section 5.3) and evaluate its performance in the more general setting of Wikipedia page listings (Section 5.4). Further, we analyze some of our design choices in an ablation study (Section 5.5). Finally, we apply our best model to the complete Wikipedia corpus and report our extraction results (Section 5.6).

5.1. Metrics

For the evaluation of our SE detection models, we stick to the common metrics for NER introduced in SemEval-2013 [25]. We report precision, recall, and F1-scores of the following scenarios:

- **Partial:** Prediction matches the boundary of the true entity at least partially.
- **Exact:** Prediction exactly matches the boundary of the true entity.
- **Ent-Type:** At least partial boundary match and entity type matches.
- **Strict:** Predicted boundary and type exactly match with the true entity.

5.2. Datasets

In the experiments, we primarily focus on Wikipedia as a data corpus due to its encyclopedic structure and the convenient mapping of entities to DBpedia and CaLiGraph. From the main dataset D which consists of all Wikipedia pages that contain listings, we create the subsets $D\text{-}LP_{train}$ and $D\text{-}LP_{test}$ (from list pages) as well as $D\text{-}P_{train}$ and $D\text{-}P_{test}$ (from any pages with listings). The statistics of the datasets are shown in Table 1. For the experiments, we use a dump of the English Wikipedia from October 2020 to be compatible with the latest release of CaLiGraph.

Table 1
Statistics of the datasets used for the experiments. The complete corpus D contains all Wikipedia pages that have listings. $D\text{-}LP_{train}$ and $D\text{-}LP_{test}$ are extracted from all Wikipedia list pages and are labelled through distant supervision; $D\text{-}P_{train}$ contains listings from arbitrary pages and contains noisy labels from a model trained on list pages while $D\text{-}P_{test}$ is annotated manually.

Dataset	**#Pages**	**#Listings**		**Items per Listing (Avg.)**		**Items per Listing (Med.)**	
		Enums	**Tables**	**Enums**	**Tables**	**Enums**	**Tables**
D	1,980,021	3,463,053	1,352,848	10.57	14.43	6	8
$D\text{-}LP_{train}$	68,494	289,666	116,715	18.06	31.26	8	12
$D\text{-}LP_{test}$	17,123	75,063	28,688	18.17	31.32	8	12
$D\text{-}P_{train}$	546,667	663,455	306,399	18.72	24.53	12	13
$D\text{-}P_{test}$	502	763	265	8.42	11.25	6	7

Table 2
Evaluation results for SE detection on Wikipedia list pages (evaluating on $D\text{-}LP_{test}$). Precision, recall and F1-score (in %) are given for the *Exact* scenario. $Ours_{LP}$ is the best configuration for $D\text{-}LP_{test}$ while $Ours_{P}$ is the best configuration for $D\text{-}P_{test}$ using $D\text{-}LP_{train}$ as training data.

Approach	**Enums**			**Tables**			**Overall**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Heist and Paulheim [3]	91	82	86	**90**	55	68	90	67	77
$Ours_{LP}$	**93**	**94**	**94**	89	**87**	**88**	**92**	**91**	**92**
$Ours_{P}$	92	93	93	88	86	87	91	90	91

The datasets $D\text{-}LP_{train}$ and $D\text{-}LP_{test}$ are created as explained in Section 3.2. For the experiments, we use a part of $D\text{-}LP_{train}$ for validation so that we have a distribution of 60% training, 20% validation, and 20% test set (similar to [3]).

The datasets $D\text{-}P_{train}$ and $D\text{-}P_{test}$ consist of listings from arbitrary Wikipedia pages. Hence, no type information is available to infer the SE labels through distant supervision. For $D\text{-}P_{train}$, we retrieved the labels as described in Section 4.4. For $D\text{-}P_{test}$, we provided the type information by manually annotating the roughly 1K listings with coarse-grained entity types (e.g. *Person* or *Organisation*). We mapped these types to their DBpedia counterparts and used this information to infer the SE labels via distant supervision. This substantially reduced the annotation effort from labelling roughly 10K listing items with concrete SE labels to labelling 1K listings with coarse-grained types. This implies that this dataset is also, in part, heuristically created and the results have to be taken with a grain of salt.

5.3. Evaluation on Wikipedia List Pages

The evaluation results for experiments on the dataset $D\text{-}LP_{test}$ are given in Table 2. We compare the approach Heist and Paulheim [3] with our model in the two configurations $Ours_{LP}$[7] and

[7] Configuration: Model roberta-base trained for 3 epochs with batch size 64, learning rate 5e-5, no warmup or weight decay, negative sample size 0.5

Table 3
Precision, recall and F1-score (in %) for SE detection on Wikipedia page listings (evaluating on *D-P*$_{test}$) using our best model configuration *Ours*$_{final}$.

Metric	Enums			Tables			Overall		
	P	R	F1	P	R	F1	P	R	F1
Partial	76	78	77	68	82	74	73	79	76
Exact	73	76	75	67	81	73	71	77	74
Ent-Type	76	78	77	65	78	71	73	78	75
Strict	71	74	73	64	77	70	69	75	72
Baseline	23	85	36	21	90	34	23	86	36

Ours$_P$.[8] Both configurations are trained with training part of *D-LP*$_{train}$ and tuned using the evaluation part. The former model configuration is the best one w.r.t. performance on *D-LP*$_{test}$. The latter model configuration is the best one w.r.t. performance on *D-P*$_{test}$. To train our models, we use the Huggingface transformer library [26].

Both of our model configurations significantly outperform the existing approach Heist and Paulheim [3], especially in terms of recall for both enumerations and tables, showing that our model can identify substantially more entities while keeping a high level of precision. For enumerations, the precision increased slightly and the recall is over ten percent higher. While precision is kept almost constant for tables, the recall increased by more than 30 percent.

5.4. Evaluation on Wikipedia Page Listings

The evaluation results for the model *Ours*$_{final}$[9] on *D-P*$_{test}$ is given in Table 3. Comparing the *Exact* scenario with the results on Wikipedia list pages, it becomes clear that the performance on arbitrary listings is worse. The losses in performance for tables are slightly higher than those for enumerations. This aligns with the observation that a lower portion of tables is usable for our approach. For tables, we have the advantage that mention boundaries are often indicated through column separators but this does not reflect in the results. In general, we notice that training the models for more than two to three epochs on *D-LP*$_{train}$ leads to overfitting on list page data and hence reduced performance on *D-P*$_{test}$.

Unfortunately, it is not possible to apply the approach Heist and Paulheim [3] to this dataset as it contains several features that are specific to list pages. As an alternative, we implemented the pick-first-entity baseline which has already proven as a strong baseline in prior work [3]. In this baseline, we simply label the first mentioned entity in an item as SE. In Table 3 we see that this baseline has a very high recall (as most SEs are mentioned in the beginning) while the precision is far lower than the one of *Ours*$_{final}$. This shows that the model is able to sort out many false positives (tripling precision) by sacrificing only some correct SEs. In cases where coverage is not the only important criterion (as is usually the case in knowledge graph completion), our model should be preferred. The more important point, however, is that our

[8] Configuration: Model roberta-base trained for 2 epochs with batch size 64, learning rate 5e-5, no warmup or weight decay, negative sample size 0.3

[9] Configuration: Similar to *Ours*$_P$ with an additional fine-tuning step of one epoch on D-*P*$_{train}$.

Table 4
Evaluation results for SE detection on Wikipedia page listings (evaluated on $D\text{-}P_{test}$) for variations of our best model configuration $Ours_{final}$. Precision, recall and F1-score (in %) are given for the *Exact* scenario.

Approach	**Enums**			**Tables**			**Overall**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
$Ours_{final}$	**73**	76	**75**	**67**	81	**73**	**71**	77	**74**
.. without item chunks	70	35	47	63	40	49	68	37	48
.. without type prediction	69	**78**	73	54	**84**	66	64	**79**	71
.. without negative sampling	71	74	73	66	81	73	70	76	73
.. without fine-tuning on pages	65	48	55	67	64	66	66	52	58

model does not depend on mention boundaries as input (which might also account for some loss in performance).

5.5. Ablation Study

To verify some assumptions that we made during the design of the SE detection approach, we perform an ablation study using the page listings dataset $D\text{-}P_{test}$. Firstly, we investigate how much chunking of items in input sequences influences the performance of the model. The results in Table 4 show that it has a slightly positive effect on precision (3% for enumerations, 4% for tables) and a roughly doubling effect on recall. The results confirm our assumption that the model is able to improve its predictions by considering the dependencies between the listing items.

Further, we investigate whether the additional prediction of entity types has an influence on the performance (as opposed to a binary prediction of SEs). The results show that there is a positive effect on precision and a slightly negative effect on recall. As the F1 measure increases slightly and as the predicted types provide additional information for downstream tasks, we stick with type prediction instead of binary SE prediction.

Additionally, we see from Table 4 that our negative sampling mechanism slightly increases the precision and recall of our final model. Consequently, the model seems to be able to learn whether there is some consistency between the listing items in the input sequence.

Finally, the fine-tuning on pages has a very strong effect on recall as it comes with an increase of 25% and the precision of the model is also increased by 5%. This result confirms that additional fine-tuning on noisy labels still yields a huge benefit.

5.6. Subject Entity Extraction over Wikipedia

Applying the model $Ours_{final}$ to the complete dataset of Wikipedia listings D took 13 hours on a single NVIDIA RTX A6000 GPU with 48GB of RAM. We extracted a total of 40 million entity mentions from 2.7M enumerations and 1M tables on 1.7 million pages. Of the 40 million entity mentions, 19.5 million can be traced back to 3.8 million known entities (i.e., the predicted mention boundary overlapped with an existing link in Wikipedia, and hence, CaLiGraph), which means that each known entity has on average 5.1 mentions. If we use that same factor of 5.1 to

Table 5
Number of extracted mentions of subject entities for the whole Wikipedia dataset of listings D aggregated by entity type.

Entity Type	#Mentions	Entity Type	#Mentions	Entity Type	#Mentions
PERSON	13,622,704	GPE	1,519,747	NORP	230,707
OTHER	9,398,003	PRODUCT	1,000,117	LANGUAGE	86,354
WORK_OF_ART	7,148,235	SPECIES	964,922	LAW	11,490
ORG	2,916,528	FAC	893,226		
LOC	1,531,452	EVENT	370,440	**Total**	**39,693,925**

estimate the number of entities for the remaining 20.5 million entity mentions, they describe 4 million additional unknown entities that could be added to the knowledge graph.

In Table 5 we display the number of extracted entity mentions aggregated by entity type. Unsurprisingly, the most frequently extracted entities are of the types *Person, Work of Art, Organisation, and Location.* Apart from that, the mention type distribution roughly resembles the distribution of entities in DBpedia [27].

6. Conclusion

In this work, we have presented a Transformer-based SE detection approach that overcomes several limitations of prior work to make it applicable to more general settings and, at the same time, improve extraction performance. An evaluation of listings of Wikipedia pages shows that the performance for such a more general setting is considerably worse than for the scenario of Wikipedia list pages. While the inferior results can partly be attributed to conceptual limitations of SE detection in arbitrary listings (c.f. Section 3.1), further improvement is necessary so that the results can be consumed by downstream applications without extensive post-filtering.

We are developing a post-filtering mechanism that takes the differences within an extracted group of SEs into account. For example, we can discard a group of extracted SEs if their predicted entity types show a high degree of diversion.

In the extraction framework of CaLiGraph, we will integrate a subsequent entity disambiguation step, which matches the identified SE mentions with existing entities or creates new entities in the knowledge graph. The main challenge will be to match SEs with existing entities and at the same time match SEs with one another (as the same entity may be occurring in multiple listings).

Complementary to the disambiguation step, we plan to further enhance CaLiGraph by using the defining axioms extracted from the listing context. The disambiguation step can be supported by the information extracted from the axioms, and similarly, the disambiguated entities can help to refine the axiom extraction.

References

[1] A. Sakor, K. Singh, A. Patel, M.-E. Vidal, Falcon 2.0: An entity and relation linking tool over Wikidata, in: 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 3141–3148.

[2] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semantic web 6 (2015) 167–195.

[3] N. Heist, H. Paulheim, Entity extraction from Wikipedia list pages, in: European Semantic Web Conference, Springer, 2020, pp. 327–342.

[4] M. Van Erp, P. Mendes, H. Paulheim, F. Ilievski, J. Plu, G. Rizzo, J. Waitelonis, Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job, in: Tenth International Conference on Language Resources and Evaluation (LREC'16), 2016, pp. 4373–4379.

[5] M. Färber, A. Rettinger, B. El Asmar, On emerging entity detection, in: European Knowledge Acquisition Workshop, Springer, 2016, pp. 223–238.

[6] G. Liu, X. Li, J. Wang, M. Sun, P. Li, Extracting knowledge from web text with monte carlo tree search, in: The Web Conference 2020, 2020, pp. 2585–2591.

[7] G. Stanovsky, J. Michael, L. Zettlemoyer, I. Dagan, Supervised open information extraction, in: 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 885–895.

[8] F. M. Suchanek, G. Kasneci, G. Weikum, YAGO: a core of semantic knowledge, in: The World Wide Web Conference, 2007, pp. 697–706.

[9] N. Heist, H. Paulheim, Uncovering the semantics of Wikipedia categories, in: International Semantic Web Conference, Springer, 2019, pp. 219–236.

[10] B. Xu, C. Xie, Y. Zhang, Y. Xiao, H. Wang, W. Wang, Learning defining features for categories, in: 25th International Joint Conference on Artificial Intelligence, 2016, pp. 3924–3930.

[11] S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation: A survey, ACM Transactions on Intelligent Systems and Technology (TIST) 11 (2020) 1–35.

[12] N. Heist, H. Paulheim, Information extraction from co-occurring similar entities, in: The Web Conference 2021, 2021, pp. 3999–4009.

[13] X. Ling, S. Singh, D. S. Weld, Design challenges for entity linking, Transactions of the ACL 3 (2015) 315–328.

[14] D. Milne, I. H. Witten, Learning to link with Wikipedia, in: 17th ACM conference on Information and knowledge management, 2008, pp. 509–518.

[15] D. Nadeau, S. Sekine, A survey of named entity recognition and classification, Lingvisticae Investigationes 30 (2007) 3–26.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of machine learning research 12 (2011) 2493–2537.

[17] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, Neural architectures for named entity recognition, in: 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 260–270.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[19] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

[20] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, C. Zhang, BOND: BERT-assisted open-domain named entity recognition with distant supervision, in: 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1054–1064.

[21] S. Broscheit, Investigating entity knowledge in BERT with simple neural end-to-end entity linking, in: 23rd Conference on Computational Natural Language Learning (CoNLL), 2019, pp. 677–685.

[22] I. Ermilov, A.-C. N. Ngomo, TAIPAN: automatic property mapping for tabular data, in: European Knowledge Acquisition Workshop, Springer, 2016, pp. 163–179.

[23] L. Zhao, L. Li, X. Zheng, J. Zhang, A BERT based sentiment analysis and key entity detection approach for online financial texts, in: 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2021, pp. 1233–1238.

[24] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108 (2019).

[25] I. Segura-Bedmar, P. Martínez Fernández, M. Herrero Zazo, SemEval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013), ACL, 2013.

[26] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Transformers: State-of-the-art natural language processing, in: 2020 conference on empirical methods in natural language processing: system demonstrations, 2020, pp. 38–45.

[27] N. Heist, S. Hertling, D. Ringler, H. Paulheim, Knowledge graphs on the web–an overview, Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges (2020) 3–22.