HapPenIng: Happen, Predict, Infer — Event Series Completion in a Knowledge Graph

Simon Gottschalk[0000−0003−2576−4640] and
Elena Demidova[0000−0002−5134−9072]

L3S Research Center, Leibniz Universität Hannover, Hannover, Germany
{gottschalk, demidova}@L3S.de

Abstract. Event series, such as the Wimbledon Championships and the US presidential elections, represent important happenings in key societal areas including sports, culture and politics. However, semantic reference sources, such as Wikidata, DBpedia and EventKG knowledge graphs, provide only an incomplete event series representation. In this paper we target the problem of event series completion in a knowledge graph. We address two tasks: 1) prediction of sub-event relations, and 2) inference of real-world events that happened as a part of event series and are missing in the knowledge graph. To address these problems, our proposed supervised HapPenIng approach leverages structural features of event series. HapPenIng does not require any external knowledge - the characteristics making it unique in the context of event inference. Our experimental evaluation demonstrates that HapPenIng outperforms the baselines by 44 and 52 percentage points in terms of precision for the sub-event prediction and the inference tasks, correspondingly.

1 Introduction

Event series, such as sports tournaments, music festivals and political elections are sequences of recurring events. Prominent examples include the Wimbledon Championships, the Summer Olympic Games, the United States presidential elections and the International Semantic Web Conference. The provision of reliable reference sources for event series is of crucial importance for many real-world applications, for example in the context of Digital Humanities and Web Science research [7, 9, 25], as well as media analytics and digital journalism [15, 23].

Popular knowledge graphs (KGs) such as Wikidata [29], DBpedia [14] and EventKG [8,10] cover event series only to a limited extent. This is due to multiple reasons: First, entity-centric knowledge graphs such as Wikidata and DBpedia do not sufficiently cover events and their spatio-temporal relations [6]. Second, reference sources for knowledge graphs such as Wikipedia often focus on recent and current events to the detriment of past events [11]. This leads to the deficiency in supporting event-centric applications that rely on knowledge graphs.

In this work we tackle a novel problem of event series completion in a knowledge graph. In particular, we address two tasks: 1) We predict missing sub-event relations between events existing in a knowledge graph; and 2) We infer real-world events that happened within a particular event series but are missing in

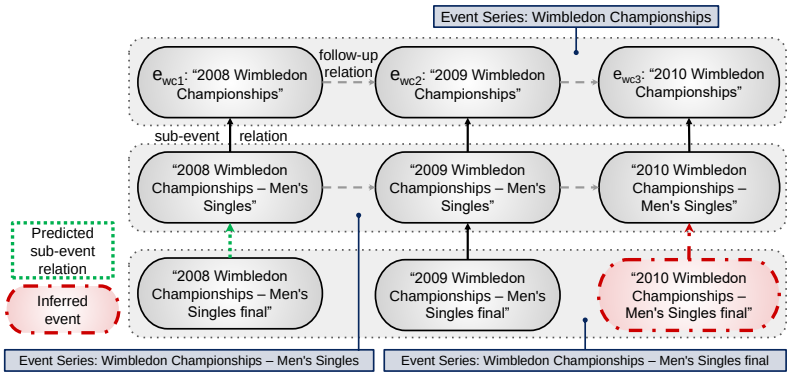


Fig. 1. A fraction of the *Event Graph* containing the Wimbledon Championships (WC) events. Nodes represent events. Solid arrows represent sub-event relations. Dashed arrows represent follow-up event relations. The three upper events are the *WC* editions.

the knowledge graph. We also infer specific properties of such inferred events such as a label, a time interval and locations, where possible. Both addressed tasks are interdependent. The prediction of sub-event relations leads to a more complete event series structure, facilitating inference of further missing events. In turn, event inference can also lead to the discovery of new sub-event relations.

The proposed `HapPenIng` approach exclusively utilizes information obtained from the knowledge graph, without referring to any external sources. This characteristic makes `HapPenIng` approach unique with respect to the event inference task. In contrast, related approaches that focus on the knowledge graphs population depend on external sources (e.g. on news [12, 31]).

The contributions of this paper include:

- A novel supervised method for sub-event relation prediction in event series.
- An event inference approach to infer real-world events missing in an event series in the knowledge graph and properties of these events.
- A dataset containing new events and relations inferred by `HapPenIng`:
  - over 5,000 events and nearly 90,000 sub-event relations for Wikidata, and
  - over 1,000 events and more than 6,000 sub-event relations for DBpedia.

Our evaluation demonstrates that the proposed `HapPenIng` approach achieves a precision of 61% for the sub-event prediction task (outperforming the state-of-the-art embedding-based baseline by 52 percentage points) and 70% for the event inference task (outperforming a naive baseline by 44 percentage points). Our dataset with new sub-event relations and inferred events is available online[1].

1.1 Example: Wimbledon Championships

The Wimbledon Championships (WC), a famous tennis tournament, are an *event series* that takes place in London annually since 1877. Wikidata currently includes 132 WC editions and 915 related sub-events, for example Women's and

[1] `http://eventkg.l3s.uni-hannover.de/happening`

Men's Singles and Wheelchair competitions. However, according to our analysis, this event series is incomplete. In particular, the `HapPenIng` approach proposed in this paper was able to generate 125 sub-event relations and 15 event instances related to this event series that are currently missing in Wikidata.

Fig. 1 illustrates a small fraction of the *Event Graph* that contains event nodes and their relations as available in Wikidata as of Sep. 18^{th}, 2018. For each year, Wikidata includes an event edition, such as the *2008 WC*. The individual competitions such as the *Men's Singles* are provided as sub-events of the corresponding edition.

In this example we can illustrate two tasks of the event series completion tackled in this paper: (i) Sub-event prediction: The missing sub-event relation between the Men's Singles final of 2008 and the Men's Singles competition in 2008 can be established; and (ii) Event inference: The missing event instance labeled *2010 WC — Men's Singles final* can be inferred as a sub-event of the Men's Singles 2010.

2 Problem Statement

We consider a typical *Knowledge Graph* that contains nodes representing real-world entities and events. The edges of a *Knowledge Graph* represent relations between entities and events. More formally:

Definition 1. ***Knowledge Graph:*** *A Knowledge Graph* $KG : \langle V, U \rangle$ *is a directed multi-graph. The nodes in* V *represent real-world entities and events. The directed edges in* U *represent relations of the entities and events in* V*.*

The *Event Graph* G is a sub-graph of the *Knowledge Graph*. The nodes of G represent real-world events. The edges represent their relations relevant in the context of event series (sub-event and follow-up relations). More formally:

Definition 2. ***Event Graph:*** *Given a Knowledge Graph* $KG : \langle V, U \rangle$*, an Event Graph* $G : \langle\ E,\ R \cup F\ \rangle$ *is a directed graph. The nodes of the Event Graph* $E \subseteq V$ *represent real-world events. The edges* R *represent sub-event relations:* $R \subseteq E \times E, R \subseteq U$*. The edges* F *represent follow-up event relations:* $F \subseteq E \times E, F \subseteq U$*.*

Events in G represent real-world happenings; the key properties of an event in the context of event series include an event identifier, an event label, a happening time interval and relevant locations.

Definition 3. ***Event:*** *Given an Event Graph* $G : \langle E, R \cup F \rangle$*, an event* $e \in E$ *is something that happened in the real world.* e *is represented as a tuple* $e = \langle uri, l, t, L \rangle$*, where* uri *is an event identifier,* l *is an event label,* $t = \langle t_s, t_e \rangle$ *is the happening time interval with* t_s*,* t_e *being its start and end time.* L *is the set of event locations.*

An event can have multiple sub-events. For example, the *WC Men's single final 2009* is a sub-event of *2009 WC*.

Definition 4. ***Sub-event:*** *An event* $e_s \in E$ *is a sub-event of the event* $e_p \in E$*, i.e.* $(e_s, e_p) \in R$*, if* e_s *and* e_p *are topically related and* e_s *is narrower in scope.*

We refer to e_p as a parent event of e_s. Typically, e_s happens in a temporal and a geographical proximity of e_p.

An event can be a part of an event series. An example of an event series is the *WC* that has the *2008 WC* as one of its editions.

Definition 5. ***Event series and editions:*** *An event series* $s = \langle e_1, e_2, \ldots, e_n \rangle$, $\forall e_i \in s : e_i \in E$, *is a sequence of topically related events that occur repeatedly in a similar form. The sequence elements are ordered by the event start time and are called editions. We refer to the set of event series as* S.

The follow-up relations F connect event editions within an event series. For example, the *2009 WC* is the follow-up event of the *2008 WC*.

Definition 6. ***Follow-up relation:*** *Given an event series* $s = \langle e_1, e_2, \ldots, e_n \rangle$, e_j *is a follow-up event of* e_i, *i.e.* $(e_i, e_j) \in F$, *if* $e_i \in s$ *and* $e_j \in s$ *are the neighbor editions in* s *and* e_i *precedes* e_j.

The sub-event relations in an *Event Graph* are often incomplete. In particular, we denote the set of real-world sub-event relations not included in the *Event Graph* as R^+. Then the task of sub-event prediction can be defined as follows:

Definition 7. ***Sub-event prediction:*** *Given an Event Graph* $G : \langle E, R \cup F \rangle$ *and events* $e_s \in E$, $e_p \in E$, *the task of sub-event prediction is to decide if* e_s *is a sub-event of* e_p, *i.e. to determine if* $(e_s, e_p) \in R \cup R^+$, *where* R^+ *is a set of real-world sub-event relations not included in the Event Graph.*

The set of real-world event representations included in an *Event Graph* is often incomplete (open world assumption). The context of event series can help to infer real-world events missing in particular editions.

Definition 8. ***Event inference:*** *Given and Event Graph* $G : \langle E, R \cup F \rangle$ *and an event series* $s = \langle e_1, e_2, \ldots, e_n \rangle$, *with* $e_1, e_2, ..., e_n \in E$, *the task of event inference is to identify a real-world event* $e_f \in E \cup E^+$ *that belongs to the series* s. *Here,* E^+ *is a set of real-world events that are not included in the Event Graph. In particular,* e_f *is a sub-event of the edition* $e_i \in s$, *i.e.* $(e_f, e_i) \in R \cup R^+$.

3 Event Series Completion

We address event series completion in two steps: First, we adopt a classification method to predict sub-event relations among event pairs. Second, we develop a graph-based approach to infer events missing in particular editions through event series analysis. A pipeline of the overall approach is shown in Fig. 2.

3.1 Sub-Event Prediction

We model the problem of sub-event prediction as a classification problem. Given an event pair (e_s, e_p), we aim to predict whether e_s is a sub-event of e_p:

$$sub-event(e_s, e_p) = \begin{cases} true, & \text{if} (e_s, e_p) \in R \cup R^+; \\ false, & \text{otherwise.} \end{cases} \quad (1)$$

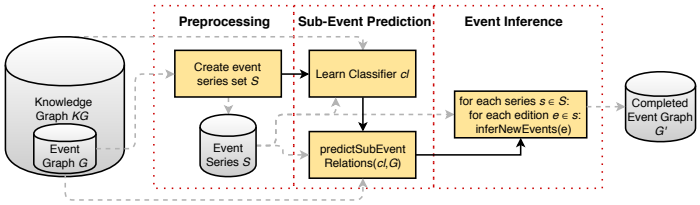


Fig. 2. The `HapPenIng` pipeline. Solid arrows represent the processing order. Dashed arrows represent the data flow.

Features We adopt textual, spatio-temporal and embeddings features.

Textual features (`TEX`): Events connected through a sub-event relation can have similar or overlapping labels whose similarity is measured using textual features. Such features are also applied on the *template labels*. Template labels are series labels obtained from the original event labels after removal of any digits. The textual features we consider include:

- Label Containment: 1, if $e_p.l$ is a sub-string of $e_s.l$, 0 otherwise.
- LCS Fraction: The length of the Longest Common Sub-string (LCS) of $e_s.l$ and $e_p.l$, compared to the shorter label: $f_{\text{LCS Fraction}}(e_s, e_p) = \frac{LCS(e_s.l, e_p.l)}{min(|e_s.l|, |e_p.l|)}$.
- Unigram Similarity: The labels of both events are split into word unigrams. The feature value is the Jaccard similarity between the unigram sets: $f_{\text{Unigram Similarity}}(e_s, e_p) = \frac{\text{unigrams}(e_s.l) \cap \text{unigrams}(e_p.l)}{\text{unigrams}(e_s.l) \cup \text{unigrams}(e_p.l)}$.
- Template Containment, Template LCS Fraction, Template Unigram Similarity: These features are computed equivalent to the label features, but are based on the template labels.
- Label Cosine Similarity: The cosine similarity between event labels based on tf-idf vectors to take frequency and selectivity of terms into account.
- Parent Event Label Length: $f_{\text{Parent Event Label Length}}(e_s, e_p) = |e_p.l|$.
- Sub-Event Label Length: $f_{\text{Sub-Event Label Length}}(e_s, e_p) = |e_s.l|$.

Spatio-temporal features (`STP`): We assume that sub-events happen in the temporal proximity of their parent events. We consider the temporal proximity through temporal overlap, containment and equality.

- Time Overlap: 1 if $e_s.t \cap e_p.t \neq \varnothing$, 0 otherwise.
- Time Containment: 1 if $e_s.t \subseteq e_p.t$, 0 otherwise.
- Time Equality: 1 if $e_s.t = e_p.t$, 0 otherwise.

Sub-events typically happen in the geographical proximity of their parent events. Therefore, we introduce Location Overlap - a spatial feature that assigns a higher score to the event pairs that share locations:

- Location Overlap: 1 if $e_s.L \cap e_p.L \neq \emptyset$, 0 otherwise.

Embedding features (`EMB`): The link structure of the *Knowledge Graph* can be expected to provide important insights into possible event relations. First, we can expect that this structure provides useful hints towards predicting sub-event relations, e.g. follow-up events can be expected to have a common parent event.

Second, events related to different topical domains (e.g. politics vs. sports) are unlikely to be related through a sub-event relation. To make use of this intuition, we train an embedding on the *Knowledge Graph* using any relations connecting two events in E. For this feature, we pre-train the embeddings following the STransE embedding model [18] which provides two relation-specific matrices $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$, a relation vector $\boldsymbol{r}$ and entity vectors (here, $\boldsymbol{e_s}$ and $\boldsymbol{e_p}$). Intuitively, given that model, we can compare the embedding of an event with the embedding of the assumed parent event plus the embedding of the sub-event relation (sE):

– Embedding Score: $f_{\text{Embedding}}(e_s, e_p) = \|\boldsymbol{W}_{r_{\text{sE}},1}\boldsymbol{e}_p + \boldsymbol{r}_{\text{sE}} - \boldsymbol{W}_{r_{\text{sE}},2}\boldsymbol{e}_s\|_{\ell_1}$

Training the sub-event classifier To train a classifier given the features presented above, a set of labeled event pairs is required. The set of positive examples contains all event pairs with known sub-event relations in the *Event Graph* G. Formally, given the set E of events, this is the set $C_+ = \{(e_s, e_p) | (e_s, e_p) \in R\}$.

In addition, a set of negative examples, i.e. event pairs without sub-event relation is required. When composing event pairs randomly, most of the paired events would be highly different (e.g. having highly dissimilar labels and no spatio-temporal overlap). Consequently, the model would only learn to distinguish the most simple cases. To address this problem, we collect a set of negative examples C_- that has as many event pairs as C_+, and consists of four equally-sized subsets with the following condition for each contained event pair (e_s, e_p):

– Both events are from the same event series, but $(e_s, e_p) \notin R$. Example: (*1997 WC — Women's Doubles*, *2009 WC — Men's Singles final*).
– Both events have the same parent event. Example: (*2009 WC — Men's Singles*, *2009 WC — Women's Singles*).
– The parent of e_s's parent is the same as e_p's parent. Example: (*2009 WC — Men's Singles final*, *2009 WC — Women's Singles*).
– e_s is a transitive, but not a direct sub-event of e_p. Example: (*2009 WC — Men's Singles final*, *2009 WC*).

Note that we only consider direct sub-event relations to be valid positive examples. In particular, we aim to learn to distinguish the directly connected sub-events from transitive relations, as well as to distinguish similar events that belong to different editions. Due to the inherent incompleteness of the *Event Graph*, a missing sub-event relation does not necessarily imply that this relation does not hold in the real world. However, we expect that false negative examples would occur only rarely in the training set, such that the resulting model will not be substantially affected by such cases.

Overall, the set of training and test instances C contains all positive sub-event examples C_+ found in the *Event Graph*, and an equally sized set of negative examples C_- that consists of the four event pair sets described above.

Predicting sub-event relations using the classifier The trained classifier is adopted to predict missing sub-event relations within event series. We apply an iterative algorithm, given a classifier cl and the *Event Graph* G. As it is not feasible to conduct a pairwise comparison of all events in G, we limit the number of events compared with their potential parent event: For each potential parent event e_p that is part of an event series, a set of candidate sub-events is selected

Algorithm 1 Event Inference

```
1: procedure InferSubEvents(e)
2:     M ← getSubSeries(e.series)
3:     for each e_s ∈ {e_s|(e_s, e) ∈ R} do M = M \ e_s.series
4:     for each m ∈ M do
5:         if constraintsNotSatisfied(m, e) then continue
6:         newEvent ← inferEvent(e, m)
7:         if oldEvent ← findEvent(E, newEvent.l) ≠ ∅ then
8:             R = R ∪ (e, oldEvent)
9:         else
10:            E = E ∪ newEvent; R = R ∪ (e, newEvent)
11:    for each e_s ∈ {e_s|(e_s, e_p) ∈ R} do inferSubEvents(e_s)
```

as the set of events with the largest term overlap with the potential parent event label. For each candidate event, the classifier cl predicts whether this event is a sub-event of e_p. To facilitate prediction of sub-event relations in cases where the parent event is not a part of the series initially, the procedure is run iteratively until no new sub-event relations are found.

3.2 Event Inference

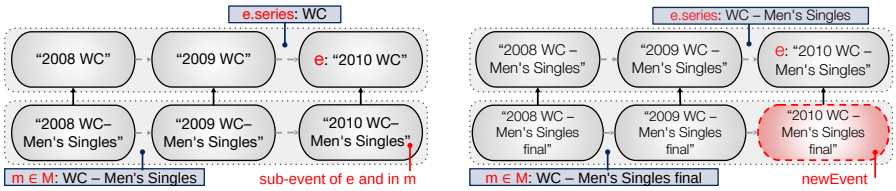
The task of event inference is to infer real-world events not initially contained in the *Event Graph* (i.e. events in the set E^+). We infer such missing events and automatically generate their key properties such as label, time frame and location, where possible. The intuition behind event inference is that the *Event Graph* indicates certain patterns repeated across editions. Thus, we approach this task via comparison of different editions of the same event series to recognize such patterns. Consider the WC example in Fig. 1. Although there is no event instance for the *2010 Men's Singles final*, we can infer such instance from the previous edition *2009 Men's Singles final*.

Event Series Pre-processing We pre-process the set S of event series to avoid cycles or undesired dependencies within the single series. Each event series is transformed into a sequence of acyclic rooted trees where each root represents one particular edition of the series. Events or relations violating that structure are removed from the series. If removal is not possible, we exclude such series from S.

An important concept of the event inference is the concept of a sub-series: A series s_p has a sub-series s_s if the sub-series contains sub-events of s_p. For example, the *WC — Men's Singles final* series is a sub-series of the *WC — Men's Singles*, because the event *2009 WC — Men's Singles final* is a sub-event of *2009 WC — Men's Singles*. We determine sub-series relation as:

Definition 9. ***Sub-series:*** *An event series $s_s \in S$ is a sub-series of $s_p \in S$, if for an event $e_p \in s_p$ there is a sub-event in s_s: $\exists (e_s, e_p) \in R : e_p \in s_p \wedge e_s \in s_s$.*

Inferring New Events The intuition behind event inference is to identify similar patterns in the different editions of an event series. According to Definition 5, the editions of an event series occur repeatedly in a similar form. This



(a) Step 1: The event inference algorithm is invoked with the *2010 WC* event e. For the sub-series m of $e.series$, *2010 WC — Men's Singles* is already a sub-event of e. No new event is inferred.

(b) Step 2: The algorithm is now invoked with the *WC 2010 — Men's Singles* event e. For the sub-series m of $e.series$, there is no sub-event of e. A new event is inferred.

Fig. 3. Event inference example for the Wimbledon Championships.

way, events repeated in most of the editions of the series, but missing in a particular edition can be inferred. To do so, we process all editions in the *Event Graph* and inspect whether its neighbored editions have a sub-event not covered in the particular edition.

Algorithm 1 illustrates our event inference approach. As shown in our pipeline (Fig. 2), this algorithm is invoked for each edition e of the event series in S. First, a set M is constructed that contains all sub-series of the current edition's series, i.e. $e.series$ (line 2). Then, the algorithm removes all series from M for which the current edition contains events already (line 3). That way, M is reduced to a set of event series not covered by the sub-events of the current edition e.

For each remaining sub-series $m \in M$, a new event is inferred that is a sub-event of the current edition e and a part of m. Within the respective method `inferEvent`(e, M), a new label, time span and set of locations is generated as described later. The algorithm is invoked recursively with all known (also newly identified) sub-events. To increase precision, a sub-series m is only retained in M if a set of constraints is satisfied (line 5). These constraints are described later in this section.

The event inference algorithm can infer an event for which an equivalent event already exists in the *Event Graph*. To avoid the generation of such duplicate events, we check if an event with the same label as the newly inferred event exists in the *Event Graph*. In this case, the algorithm adds a new sub-event relation across the existing events to the *Event Graph* and discards the inferred event (line 8).

Wimbledon Championships Example: Consider the example in Fig. 1, with the goal to infer new events within the edition e_{wc_3}: *2010 WC*. Fig. 3a depicts the first step when invoking the algorithm `InferSubEvents`(e_{wc_3}) (without constraints). The edition becomes the input event e and its series $e.series$ is *WC*. The event series *WC — Men's Singles* (m) is identified as one of its immediate sub-series in M. However, as there is already an event *2010 WC — Men's Singles* that is a sub-event of e and part of that sub-series m, it is removed from M. Therefore, M is empty and no new events are inferred at this point.

Algorithm 2 Label Generation

```
1: procedure GenerateLabel(e, m)
2:     mostSimilarEvents ← getSimilarEvents(e, e.series)
3:     sortEventsByEditionCloseness(e, mostSimilarEvents)
4:     c ← mostSimilarEvents[0]
5:     c′ ← c′, s.t. (c′, c) ∈ R ∧ c′ ∈ m
6:     l ← ""; r ← c′.l; δ_prev ← ∅
7:     for each δ ∈ getEdits(c.l, e.l) do
8:         if δ.op = DELETE then δ_prev ← δ
9:         else if δ.op = INSERT ∧ δ_prev.op = DELETE then
10:            l ← l + r[: r.indexOf(δ_prev.text)] + δ.text
11:            r ← l + r[r.indexOf(δ_prev.text) + len(δ_prev.text) :]
12:        else if not (δ.op = EQUAL ∧ δ_prev = ∅) then return ∅
       return l + r
```

step	δ.op	δ_{prev}.op	δ.text	l	r
init					2009 WC - Men's Singles final
1	DELETE		2009		2009 WC - Men's Singles final
2	INSERT	DELETE	2010	2010	WC - Men's Singles final
3	EQUAL		WC - Men's Singles	2010	WC - Men's Singles final

Table 1. Generating the label *2010 WC - Men's Singles*. The edit operations δ are the result of Myers' algorithm to detect the edit operations between *2009 WC - Men's Singles* and *2010 WC - Men's Singles*. The final label is the concatenation of l and r.

Subsequently, Algorithm 1 is executed with the sub-event *2010 WC — Men's Singles* as input edition e, as shown in Fig. 3b. Here, the sub-series is *WC — Men's Singles final* which is inserted in M. Consequently, a new event is created that is a sub-event of e and part of the event series *WC — Men's Singles final*.

Label Generation Each newly generated event requires a label. This label is generated by exploiting the labels within its event series, as shown in Algorithm 2. The input is its future parent event e and its event series m. First, the events in the parent series $e.series$ whose labels are most similar to the label of e are collected (line 2). Then, within this set of events, the one from the closest event edition and its sub-event in m is selected (lines 3 - 5). Finally, the label of that event is transformed into the new label by applying the same edit operations δ (i.e. equality, delete or insert) as if we transformed the parent event labels (lines 6 - 12). To identify the edits, we adopt the difference algorithm by Myers [16].

Example: Consider the newly added event in Fig. 3b. As an input to the algorithm, there is e which is the event *2010 WC — Men's Singles* and the series m consisting of the Men's Singles finals of 2008 and 2009. First, the event *2009 WC — Men's Singles* within $e.series$ is identified as the most similar event c. c' is the sub-event of c that is also in m: *2009 WC — Men's Singles final*. Given $c'.l$ and the edit operations δ between the labels of e and c, Table 1 shows how they are used to generate the correct label *2010 WC — Men's Singles final*.

Location and Time Generation Each event can be assigned a happening time and a set of locations. In both cases, we use a rule-based approach.

Locations: Some events such as the Olympic Games change their location with every edition. Currently we reconstruct event locations only if they remain unchanged across editions: If there is a location assigned to every event $s \in m$, this location is also assigned to e. In future work we intend to utilize sub-location relations, that facilitate the generation of correct locations at a lower level of geographical granularity.

Happening Times: Three rules are applied in the following order until a happening time is identified: a) If the happening time of each event $s \in m$ equals its parent event's happening time, also e adopts its happening time directly from its parent event; b) If the happening time of each event $s \in m$ is modelled as a whole year, the happening time of e is also modelled as the same year as any of its (transitive) parent events; c) If the event label contains a year expression, that part is transformed into its happening time.

Constraints We propose several configurations of constraints to decide whether an event should be created:

- Baseline (`BSL`): No constraints.
- Time Evolution (`EVO`): The constraints are only satisfied if there was at least one event in the series that happened before e. For example, the *Wimbledon Women's Doubles* were held for the first time in 1913, so it would be wrong to generate an event for the *Women's Doubles* series in 1912 and before.
- Interval (`INT`): The constraints are only satisfied if there was at least one event in the series that happened before and at least one event in the series that happened after e. Under this constraint, events that re-occurred only until a specific edition are not generated for each edition. An example is the tug of war which was part of only six Olympic Summer Games.
- Window (`WIN`): Given a start and an end thresholds a and b, this constraint is satisfied if there is at least one event within the last a editions of the series that happened before e and at least one event in the following b editions that happened after e. For example, Tennis competitions in the Olympic Summer Games were held between 1896 and 1924, and then only since 1984. The Window constraint helps to identify such gaps.
- Coverage (`COV`): Event series are only valid if they are part of a sufficient fraction of the editions: $|m|/|S| \geq \alpha$, given a threshold α.
- Coverage Window (`CWI`): A combination of `WIN` and `COV`: The coverage is only computed after restricting both event series to the dynamic time window.
- Evolution Coverage Window (`ECW`): A combination of `EVO`, `WIN` and `COV`: The coverage is only computed after restricting both event series to the dynamic time window, and if at least one event in the series happened before e.

4 Evaluation

The goal of the evaluation is to assess the performance of the `HapPenIng` approach with respect to the sub-event prediction and event inference tasks.

4.1 Data Collection and Event Graph Construction

We run our experiments on *Event Graphs* extracted from two sources: (i) Wikidata [29] as of October 25, 2018 (*Wikidata Event Graph*), and (ii) DBpedia [14]

from the October 2016 dump (*DBpedia Event Graph*). Both datasets are enriched with additional information regarding events obtained from the EventKG knowledge graph [8]. Compared to other knowledge graphs, EventKG contains more detailed information regarding spatio-temporal characteristics of events. More concretely, events in the *Event Graph* are enriched with location and time information using the properties *sem:hasPlace*, *sem:hasBeginTimeStamp* and *sem:hasEndTimeStamp* of EventKG.

One *Event Graph* containing events, sub-event relations and follow-up relations, as well as a set S of event series is constructed for each dataset. For the *Wikidata Event Graph*, we collect as events all data items that are (transitive) instances of the "event" class[2]. Event series are extracted using the "instance of"[3] and the "series"[4] properties in Wikidata. For the *DBpedia Event Graph*, we extract events using the "dbo:Event" class and series assignments using the provided Wikipedia categories. In both cases, we apply two heuristics to ensure that only event series compatible with Definition 5 are extracted: (i) We only consider series with mostly homogeneous editions. To this end, we make use of the Gini index [21], a standard measure for measuring impurity. In our context it is used to assess the diversity of the template labels of editions in an event series. We reject the (rare) cases of event series with high Gini impurity, where the edition labels do not follow any common pattern.[5] An event is kept in S, if the set of template labels of its editions shows a Gini impurity less than 0.9. Besides, we ignore editions whose removal decreases that impurity. (ii) We ignore events typed as military conflicts and natural disasters, because such events typically do not follow any regularity. If we can find connected sub-graphs of events in the *Event Graph* through sub-event and follow-up relations, but the data item representing that series is missing in the dataset, we add a new unlabeled event series to S. To train the embeddings, we collect all relations connected to events.

The extraction process results in a *Wikidata Event Graph* G containing $|E| = 352,235$ events (*DBpedia Event Graph*: $92,523$) and $|S| = 9,007$ event series (*DBpedia Event Graph*: $1,871$). As input to train the embeddings, there are $279,004,908$ relations in Wikidata and $18,328,678$ relations in DBpedia. Both *Event Graphs*, as well as embeddings, annotated samples and other evaluation datasets described in the remainder of this section are available online.[6]

4.2 Sub-Event Prediction

Training and Test Set Generation Before running the experiments, a set of positive and negative sub-event relations is created from the *Event Graphs* as described in Section 3.1. In total, this collection of relations consists of $55,217$ event pairs within S that were extracted as correct sub-event pairs from Wikidata

[2] `https://www.wikidata.org/wiki/Q1656682`
[3] https://www.wikidata.org/wiki/Property:P31
[4] https://www.wikidata.org/wiki/Property:P179
[5] For example, the event series "TED talk", whose set of edition template labels (e.g. "Avi Reichental: What's next in 3D printing" and "Amanda Palmer: The art of asking") has a high Gini impurity, is not included in the set of event series.
[6] `http://eventkg.l3s.uni-hannover.de/happening`

Method		Wikidata					DBpedia
		TP	**TN**	**FP**	**FN**	**Accuracy**	**Accuracy**
Baseline	`STransE`	46,479	43,143	6,949	13,859	0.81	0.50
`HapPenIng` configurations	`LOG`	54,345	46,605	3,487	5,993	0.91	0.87
	`SVM`	55,958	48,825	1,267	4,380	0.95	0.92
	`RF`	**58,649**	**49,497**	**595**	**1,689**	**0.98**	**0.97**

Table 2. 10-fold cross-validation of the sub-event prediction using different classifiers and all the introduced features. `STransE` is the baseline we compare to.

(DBpedia: 16,763) and the same number of negative event pairs.[7] This collection is split into ten folds to allow 10-fold cross-validation. We learn the STransE embeddings as described in Section 3.1 for each fold, with its parameters set as follows: SGD learning rate $\lambda = 0.0001$, the margin hyper-parameter $\gamma = 1$, vector size $k = 100$ and 1,000 epochs. While learning the embeddings on the folds, we exclude the sub-event relations from the respective test set.

Baseline As a baseline for sub-event prediction, we utilize an embedding-based link prediction model based on the STransE embeddings [18]. Given an input event, this model retrieves a ranked list of candidate sub-events with the corresponding scores. We use these scores to build a logistic regression classifier. STransE is a state-of-the-art approach that had been shown to outperform previous embedding models for the link prediction task on the FB15K benchmark [3].

Classifier Evaluation Table 2 shows the results of the 10-fold cross-validation for the sub-event prediction task, with three different classifiers: `LOG` (Logistic Regression), `RF` (Random Forest) and `SVM` (Support Vector Machine with linear kernel and normalization) in terms of classification accuracy ($\frac{TP+TN}{TP+TN+FP+FN}$, where TP are true positives, TN true negatives, FP false positives and FN false negatives). Among our classifiers, the `RF` classifier performs best, with an accuracy of nearly 0.98 in the case of the *Wikidata Event Graph* and 0.97 for the *DBpedia Event Graph*. The results show a clear improvement over the `STransE` baseline, outperforming the baseline by more than 16 percentage points in case of the `RF` classifier for Wikidata. For DBpedia, the `STransE` baseline is outperformed by a larger margin using our proposed features. This can be explained by the insufficient number of relations for training the embeddings in DBpedia.

Table 3 shows the performance of the `RF` classifier under cross-validation with different feature groups. The combination of all features leads to the best performance in terms of accuracy. Although the use of textual features already leads to a high accuracy (0.97), embedding features and spatio-temporal features help to further increase accuracy in the case of Wikidata (0.98). Again, while DBpedia does profit from the spatio-temporal features, there is no improvement when using embeddings, due to the insufficient data size.

[7] Existing benchmark datasets do not contain a sufficient amount of sub-event relations. For example, FB15K [3] only contains 224 triples containing one of the Freebase predicates */time/event/includes_event*, */time/event/included_in_event* or */time/event/instance_of_recurring_event*.

Feature Group	Wikidata Accuracy	DBpedia Accuracy
All features: TEX, STP, EMB	**0.98**	0.97
No spatio-temp. features: TEX, EMB	0.97	0.96
No textual features: STP, EMB	0.82	0.73
No embedding: TEX, STP	0.98	**0.97**

Table 3. 10-fold cross-validation of the sub-event prediction using the RF classifier for Wikidata and DBpedia with different feature sets.

Wikidata Statistics and Examples While the classifiers demonstrate very accurate results on the test sets, the performance on predicting sub-event relations not yet contained in G requires a separate evaluation. As explained in Section 3.1, a large number of predictions is needed that could potentially also lead to a large number of false positives, even given a highly accurate classifier. The actual label distribution is skewed towards unrelated events and we are now only classifying event pairs not yet contained in R. In fact, running the sub-event prediction algorithm using the best-performing RF classifier with all features leads to the prediction of $85,805$ new sub-event relations not yet contained in Wikidata and $5,651$ new sub-event relations in DBpedia.

To assess the quality of the predicted sub-event relations that are not initially contained in R, we extracted a random sample of 100 sub-event relations consisting of an event and its predicted sub-event and manually annotated each pair as correct or incorrect sub-event relation. According to this manual annotation, 61% of the sub-event relations predicted with our HapPenIng approach that are not yet contained in the *Event Graph* correctly represent real-world sub-event relations in Wikidata (DBpedia: 42%). In comparison, the STransE baseline predicted only $46,807$ new sub-event relations, and only 9% of them are correct based on a manual annotation of a random 100 relations sample (DBpedia: 2%). The difference in performance on the test set and on the predicted sub-event relations not contained in R can be explained by the large class disbalance in the set of relations collected in the sub-event prediction procedure, such that the majority of the candidate relations are negative examples.

4.3 Event Inference Performance

We evaluate the event inference performance in two steps: First, we conduct an automated evaluation of recall by reconstruction of corrupted event series. Second, we assess precision by annotating random samples of new events.

Complementing Corrupted Event Series (Recall) To evaluate the recall of the event series completion, we remove events from the event series and investigate to which extent our *Event Graph* completion constraints are able to reconstruct them (we consider the naive unconstrained approach BSL as our baseline). To this end, we randomly remove leaf nodes (events without sub-events) from the whole set of event series S until a specific percentage (determined by the *corruption factor*) of leaf nodes is removed. For the *Wikidata Event Graph*, there are $45,203$ such leaf events in total before corruption, for DBpedia $9,600$. Table 4 shows the results for three corruption factors (5%, 10% and 15%) and

Constraints		Wikidata Corruption Factor 5%	Wikidata 10%	Wikidata 15%	DBpedia 5%	DBpedia 10%	DBpedia 15%
Baseline	BSL	61.81	63.13	61.83	39.58	38.40	38.17
HapPenIng configurations	EVO	53,63	54.70	53.12	31.04	31.32	30.12
	INT	46.68	47.89	46.39	24.58	24.04	23.46
	WIN	46.06	47.45	45.94	22.71	22.27	21.93
	COV	45.49	45.65	43.64	11.46	11.03	9.30
	CWI	53.36	53.93	51.32	23.96	21.96	19.43
	ECW	48.89	49.17	47.03	21.67	20.71	18.18

Table 4. Complementing corrupted event series. For each corruption factor (i.e. % of removed events), we report the percentage of events that could be reconstructed.

Constraints		Wikidata Inferred Events Number	Wikidata Inferred Events P	Wikidata Relations	DBpedia Inferred Events Number	DBpedia Inferred Events P	DBpedia Relations
Baseline	BSL	**114,077**	0.26	**16,877**	**31,410**	0.24	**3,420**
HapPenIng configurations	EVO	28,846	0.47	10,045	11,295	0.35	1,170
	INT	5,256	0.57	5,376	2,115	0.67	3,419
	WIN	3,363	0.56	4,547	936	**0.71**	783
	COV	7,297	0.54	2,712	1,313	0.45	417
	CWI	7,965	0.59	4,442	1,965	0.61	718
	ECW	5,010	**0.70**	3,687	1,364	0.70	655

Table 5. Manual evaluation of the correctness of inferred events. For the baseline, each HapPenIng constraint and *Event Graph*, 100 inferred events were randomly sampled and judged as correct or not. The number of additional sub-event relations found during the event inference process is reported as well (P: Precision).

the constraints introduced in Section 3.2 (we set the parameters to $a = b = 5$ and $\alpha = 0.5$). As expected, the unconstrained naive approach BSL results in the highest percentage of correctly reconstructed events: More than 60% of the Wikidata and nearly 40% of the DBpedia events can be recovered including their correct labels. If applying constraints, less events are reconstructed. In particular, the WIN constraint results in the lowest recall, as it demands to cover the event before and after the series edition within 5 editions.

Overall, we observe that HapPenIng is able to reconstruct more than 60% of missing events from a knowledge graph and correctly infer event labels.

Manual Assessment (Precision) To access precision, we created random samples of 100 newly inferred events for each of the constraints proposed in Section 3.2 and both *Event Graphs*, and manually annotated their correctness. Table 5 provides an overview of the results. While the naive unconstrained approach results in a precision of less than 0.30 for both *Event Graphs*, the inclusion of constraints leads to clear improvement, with a precision of up to 0.70 for the ECW constraint for Wikidata and 0.71 for the WIN constraint for DBpedia. Table 5 also reports the number of additional sub-event relations created during the event inference procedure when checking for duplicate events.

Discussion and Additional Statistics The manual assessment shows that `HapPenIng` with the `ECW` constraints is able to infer 5,010 new events with a precision of 70% in Wikidata and 1,364 new DBpedia events with similar precision. Events are inferred wrongly in cases where sub-events are happening in an irregular manner. This includes e.g. the wrongly inferred event "1985 Australian Open – Mixed Doubles" that was extracted although there were no Mixed Doubles in that event series between 1970 and 1985 or competitions like the men's single scull in the World Rowing Championships that used to follow a highly unsteady schedule. In future, external knowledge can be used to verify the inferred events. Differences between the Wikidata and the DBpedia results can be explained by the less complete event type assignments and the lack of a proper sub-event relation in DBpedia, where we use category assignments instead.

As the `ECW` constraint is most precise for the *Wikidata Event Graph*, we provide more insights for this constraint and *Event Graph* in the following:

- Impact of the sub-event prediction on the event inference: If the sub-event prediction step is skipped, only 3,558 new events are inferred, compared to 5,010 events otherwise.
- Additional relations: 3,687 new sub-event relations were created during the event inference step in addition to the 85,805 sub-event relations from the sub-event prediction step (in total: 89,492 new sub-event relations).
- Happening times: 99.36% of the inferred events are assigned a happening time. 0.38% of them were inferred by the first, 81.52% by the second and 18.10% by the third rule from Section 3.2.
- Locations: Only 79 of the 5,010 inferred events were assigned a location under the strict conditions proposed in Section 3.2.

Overall, the two steps sub-event prediction and event inference enable `HapPenIng` to generate ten thousands of new sub-event relations and events. These relations and new instances can be given as a suggestion to be inserted in the respective dataset using human confirmation with external tools, such as the Primary Sources Tool for Wikidata [26].

5 Related Work

Knowledge Graph Completeness. Completeness is an important dataset quality dimension [5]. Due to the *open-world assumption* knowledge graphs are notoriously incomplete. The facts not present in the knowledge graph are unknown, and may or may not be true [22,27]. There has been research on several exemplary aspects of knowledge graph completeness, for example on the incompleteness of Wikidata [1,2] and the relation between obligatory attributes and missing attribute values [13]. In our previous work, we considered the problem of integration and fusion of event-centric information spread across different knowledge graphs and created the EventKG knowledge graph that integrates such information [8,10]. [28] addressed the inference of missing categorical information in event descriptions in Web markup. These works emphasize the need for knowledge graph completion, in particular regarding event-centric information.

Knowledge Graph Completion. None of the knowledge graph completion and refinement tasks has yet considered the inference of new nodes given only the

knowledge graph itself [19,30]. Paulheim [19] identifies three different knowledge graph completion approaches: (i) *Type Assertions Completion*. Type assertions completion is the task of predicting a type or a class of an entity [19]. A common approach to this task is to probabilistically exploit type information that is inherent in the statement properties [20]. (ii) *Link Prediction*. With link prediction, a ranked lists of candidates for the missing item of an incomplete triple is generated, typically based on embeddings as performed in the TransE [3], STransE [18] and other graph embedding models [24, 30]. In `HapPenIng` we generate new events not originally present in the knowledge graph and profit from the inclusion of textual and tempo-spatial features on top of embeddings. (iii) *External Methods*. Information extraction approaches and graph algorithms can be used to detect new relations [22] or entity/event nodes [12] from external textual data. Instead, `HapPenIng` solely relies on the information inherent to the knowledge graph and does not depend on the availability of the text corpora.

Knowledge Graph Completion Tools: A recent survey of link discovery frameworks is provided in [17]. As human-curated knowledge graphs such as Wikidata demand a high quality of inserted data, there have been several tools developed that help integrating automatically generated information with the respective knowledge graph. This includes the Primary Sources Tool [26], where suggestions for new relations are confirmed by humans and [4] that provides an overview of potentially missing information. Such tools can help to integrate inferred event series data into existing knowledge graphs.

6 Conclusion

In this paper we addressed a novel problem of event series completion in a knowledge graph. The proposed `HapPenIng` approach predicts sub-event relations and real-world events missing in the knowledge graph and does not require any external sources. Our evaluation on Wikidata and DBpedia datasets shows that `HapPenIng` predicts nearly 90,000 sub-event relations missing in Wikidata (in DBpedia: over 6,000), clearly outperforming the embedding-based baseline by more than 50 percentage points, and infers over 5,000 new events (in DBpedia: over 1,300) with a precision of 70%. These events and relations can be used as valuable suggestions for insertion in Wikidata and DBpedia after manual verification. We make our dataset publicly available to encourage further research.

Acknowledgements. This work was partially funded by the EU Horizon 2020 under MSCA-ITN-2018 "Cleopatra" (812997), and the Federal Ministry of Education and Research, Germany (BMBF) under "Simple-ML" (01IS18054).

References

1. Ahmeti, A., Razniewski, S., Polleres, A.: Assessing the Completeness of Entities in Knowledge Bases. In: ESWC. pp. 7–11. Springer (2017)
2. Balaraman, V., Razniewski, S., Nutt, W.: Recoin: Relative Completeness in Wikidata. In: WWW Companion (2018)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. In: NIPS. pp. 2787–2795 (2013)
4. Darari, F., et al.: COOL-WD: A Completeness Tool for Wikidata. In: ISWC (2017)

5. Ellefi, M.B., Bellahsene, Z., et al.: RDF Dataset Profiling - a Survey of Features, Methods, Vocabularies and Applications. Semantic Web **9**(5), 677–705 (2018)
6. Färber, M., Ell, B., Menne, C., Rettinger, A.: A Comparative Survey of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. Semantic Web Journal **1**, 1–5 (2015)
7. Gottschalk, S., Bernacchi, V., Rogers, R., Demidova, E.: Towards Better Understanding Researcher Strategies in Cross-Lingual Event Analytics. In: TPDL (2018)
8. Gottschalk, S., Demidova, E.: EventKG: A Multilingual Event-Centric Temporal Knowledge Graph. In: ESWC. Springer (2018)
9. Gottschalk, S., Demidova, E.: EventKG+TL: Creating Cross-Lingual Timelines from an Event-Centric Knowledge Graph. In: ESWC Satellite Events (2018)
10. Gottschalk, S., Demidova, E.: EventKG - the Hub of Event Knowledge on the Web - and Biographical Timeline Generation. Semantic Web (2019)
11. Kaltenbrunner, A., Laniado, D.: There is No Deadline - Time Evolution of Wikipedia Discussions. In: WikiSym. ACM (2012)
12. Kuzey, E., Vreeken, J., Weikum, G.: A Fresh Look on Knowledge Bases: Distilling Named Events from News. In: CIKM. pp. 1689–1698. ACM (2014)
13. Lajus, J., Suchanek, F.M.: Are All People Married?: Determining Obligatory Attributes in Knowledge Bases. In: WWW. pp. 1115–1124 (2018)
14. Lehmann, J., Isele, R., Jakob, M., et al.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web **6**(2) (2015)
15. Mishra, A., Berberich, K.: Leveraging Semantic Annotations to Link Wikipedia and News Archives. In: ECIR. Springer (2016)
16. Myers, E.W.: An O(ND) Difference Algorithm and its Variations. Algorithmica **1**(1-4) (1986)
17. Nentwig, M., Hartung, M., Ngomo, A.N., Rahm, E.: A Survey of Current Link Discovery Frameworks. Semantic Web **8**(3), 419–436 (2017)
18. Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M.: STransE: a Novel Embedding Model of Entities and Relationships in Knowledge Bases. In: NAACL HLT (2016)
19. Paulheim, H.: Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. Semantic Web **8**(3), 489–508 (2017)
20. Paulheim, H., Bizer, C.: Type Inference on Noisy RDF Data. In: ISWC (2013)
21. Raileanu, L.E., Stoffel, K.: Theoretical Comparison between the Gini Index and Information Gain Criteria. Annals of Mathematics and Artificial Intelligence **41**(1), 77–93 (2004)
22. Razniewski, S., et al.: But What Do We Actually Know? In: AKBC (2016)
23. Setty, V., Anand, A., Mishra, A., Anand, A.: Modeling Event Importance for Ranking Daily News Events. In: WSDM. ACM (2017)
24. Shi, B., Weninger, T.: ProjE: Embedding Projection for Knowledge Graph Completion. In: AAAI-17. pp. 1236–1242 (2017)
25. Swan, R., Allan, J.: Automatic Generation of Overview Timelines. In: SIGIR (2000)
26. Tanon Pellissier, T., et al.: From Freebase to Wikidata: The Great Migration. In: WWW (2016)
27. Tanon Pellissier, T., Stepanova, D., et al.: Completeness-aware Rule Learning from Knowledge Graphs. In: ISWC. Springer (2017)
28. Tempelmeier, N., Demidova, E., Dietze, S.: Inferring Missing Categorical Information in Noisy and Sparse Web Markup. In: The Web Conference (2018)
29. Vrandečić, D.: Wikidata: A New Platform for Collaborative Data Collection. In: WWW Companion. pp. 1063–1064. ACM (2012)
30. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE TKDE **29**(12), 2724–2743 (2017)
31. Yuan, Q., et al.: Open-Schema Event Profiling for Massive News Corpora. In: CIKM (2018)