Inferring Temporal Knowledge for Near-Periodic Recurrent Events

Dinesh Raghu[*][1][2][4], **Surag Nair**[*][3][5], **Mausam**[1]
[1] IIT Delhi, New Delhi, India
[2] IBM Research, New Delhi, India
[3] Stanford University, Stanford, CA, USA
diraghu1@in.ibm.com, surag@stanford.edu, mausam@cse.iitd.ac.in

Abstract

We define the novel problem of extracting and predicting occurrence dates for a class of *recurrent events* that are held periodically as per a near-regular schedule (e.g., conferences, film festivals, sports championships). Knowledge-bases such as Freebase contain a large number of such recurring events, but they also miss substantial information regarding specific event instances and their occurrence dates. We develop a temporal extraction and inference engine to fill in the missing dates as well as to predict their future occurrences. Our engine performs joint inference over several knowledge sources – (1) information about an event instance and its date extracted from text by our temporal extractor, (2) information about the typical schedule (e.g., "every second week of June") for a recurrent event extracted by our schedule extractor, and (3) known dates for other instances of the same event. The output of our system is a representation for the event schedule and an occurrence date for each event instance. We find that our system beats humans in predicting future occurrences of recurrent events by significant margins. We release our code and system output for further research.

1 Introduction

Extraction of events from text has received significant attention within the information extraction (IE) literature [Konovalov *et al.*, 2017; Foley *et al.*, 2015; Zhang *et al.*, 2015; Intxaurrondo *et al.*, 2015; Reschke *et al.*, 2014; Kuzey *et al.*, 2014; Ritter *et al.*, 2012]. However, to the best of our knowledge, there is little work on the extraction or prediction of *recurrent events* – events that occur more than once. Even though events like bombings (sadly) recur, 'predicting' their next occurrence is a rather challenging problem. In this work, we focus on *near-periodic* recurrent events – events that occur according to some pre-defined near-regular schedule.

[*]D. Raghu and S. Nair contributed equally to this work.

[4]D. Raghu is an employee at IBM Research. This work was carried out as part of PhD research at IIT Delhi.

[5]This work was carried out when S. Nair was an undergrad student at IIT Delhi.

Near-periodic recurrent events are an important subclass of events. They contain many popular events people plan on attending (or avoiding), such as 'Edinburgh Art Festival', 'IJCAI conference', and 'Kumbh Mela'. FreeBase lists a total of 15,770 recurring events, which means it should list nearly 146K event instances for the last 10 years (as per Freebase over 93% of the recurrent events happen yearly or more than once a year). However, FreeBase lists only 16,347 instances, suggesting massive missing information. Hence KB completion for these events is an important technical challenge to be addressed.

Another important aspect of the problem concerns future occurrences of these events. Because recurrent events follow somewhat regular schedules, it is possible to *predict* the future dates without even reading them explicitly in text. Annotating FreeBase with predicted future occurrences of an event will allow tourists to plan accordingly. It can also be of interest to airline companies, which can suitably schedule additional flights proactively to meet the high demand.

We combine both these related problems into a novel joint inference problem that uses three knowledge sources, to output the best occurrence date for each past missing instance, and the best schedule that helps in predicting future occurrences. We call our joint inference component as *TRINE, TempoRal Inference for Near-periodic Events*.

Our work exploits three sources of knowledge. The first is descriptive text about the recurrent event, which might explicitly mention its schedule. For example, Wikipedia description of 'Super Bowl' states that "The Super Bowl is currently played on the first Sunday in February." The second source of knowledge is any known occurrences of the event in the past. This may come from an existing knowledge base such as Freebase. The third source is a text corpus that contains mentions of event instances and associated dates. For example, an article may contain the sentence "The next game of Super Bowl is scheduled for February 4, 2018.".

In this work we focus on *precise* prediction of the occurrence dates. When performing knowledge base completion, a few precise fine grained predictions (day level) are more valuable than many a broad (month or season level) predictions. Most of the existing knowledge bases are hand curated. Providing precise missing knowledge along with provenance can help ease the process for the curators considerably.

There are several challenges for developing the joint inference model in TRINE. First, some sources may be missing or inaccurate. For example, the descriptive text may not list a schedule or the extraction of the schedule may be incorrect. Second, recurrent events may not even follow a strict schedule. We commonly found situations where a specific instance went off schedule, or that the schedule shifted over time. There are also NLP challenges in slot filling of occurrence dates. Due to lack of training data, we need to use distant supervision [Surdeanu and Ji, 2014], which automatically annotates an unlabeled text corpus with recurrent event instance mentions and their occurrence dates using the knowledge in Freebase. Distant supervision typically results in low to mid precision and recall.

Finally, we also need to develop a novel NLP model for extracting schedules from text, as this is a novel problem that arises in the context of recurring events. We solve these challenges by developing a probabilistic model for joint inference in TRINE – it can recover from noise in extracted schedules and occurrence dates, and also allows a few instances to go off-schedule.

We perform several experiments to assess TRINE's benefits. For predicting future events, we perform a human evaluation and find that TRINE beats humans convincingly. For KB completion, when instance extractions are available, TRINE fills in missing instances at a Mean Reciprocal Rank [Craswell, 2009] over 0.5 (i.e., one of top two predictions is correct on average). Overall, our contributions are:

1. We define the novel task of inferring past and future temporal knowledge of near-periodic recurrent events.
2. We present TRINE, a joint model to infer the schedules of recurrent events and occurrence dates for each instance.
3. Our experiments show that our system vastly outperforms several natural baselines and also crowd workers.
4. We release our code and system output for further use by research community.[6]

2 Related Work

Our work is focused towards using a *joint* model for inferring multiple temporal attributes together. This falls in the line of [Talukdar *et al.*, 2012; Do *et al.*, 2012], which perform joint inference over temporal attributes using pre-defined constraints; these works do not consider recurrent events.

Temporal IE: Temporal Slot Filling (TSF) task in TAC-KBP [Surdeanu, 2013], extracts/infers temporal constraints for relations in knowledge base using evidence from a large text corpus. The two main difference between TSF and our work are: (1) we extract precise dates, while TSF extracts a range for a date and, (2) while both use weak supervision for annotating training data, TSF is less prone to noise as their supervision requires a pair of entities in the relation, whereas we use just a single entity (event instance). We automatically generate training data using ideas similar to the state of the art in distant supervision for temporal extraction [Ji *et al.*, 2014].

[6]Available at *https://github.com/dair-iitd/trine*

Slot Filling (SF): SF subtask [Surdeanu and Ji, 2014] in the Knowledge Base Population (KBP) track at TAC [Surdeanu *et al.*, 2011; Adel *et al.*, 2016] is quite close to our subtask of extracting instance occurrence dates. SF fills incomplete slots (such as born-in, employed-by) in a knowledge base by reading a large text corpus. SF extracts only one slot at a time, and only if it is mentioned in the text corpus. However, TRINE jointly infers all the missing slots (occurrence dates) of the recurrent event instances. This helps in not only correcting a few extraction mistakes, but also potentially filling slots that are not even mentioned in the text corpus.

Event Extraction: Foley et al. [2015] propose a distant supervision based approach using *Schema.org* event annotations to extract non-recurrent events from web pages. The greedy approach used for grouping event attributes results in low precision. We leverage the fact that event instances might follow a near-regular schedule to combat this low precision.

Kunneman and Van den Bosch [2015] propose an approach to automatically extract recurrent events from tweets. There are clear differences between their work and ours. Firstly, they do not use a KB or any descriptive text about recurrent events as input. So, they need not build a schedule extractor, and cannot build any component analogous to TRINE for joint inference. Secondly, while the use of tweets generally makes extraction harder due to text normalization issues, it also likely makes temporal extraction for events more accurate due to the 140 char limit.

Previous works on event extraction [Ritter *et al.*, 2012; Zhang *et al.*, 2015; Bethard and Martin, 2006; Patwardhan and Riloff, 2009] widely vary based on the techniques for event mention discovery. As most recurrent events are named, our event discovery is relatively straightforward. We focus on leveraging the near-periodic nature of recurrent event instances to better aggregate the associated temporal attributes.

3 Problem Definition

Let e be a recurrent event with a set of instances $\mathcal{I}^e = \{I_1^e, I_2^e, ..., I_{N_e}^e\}$. For example, 'Super Bowl' is a recurrent event and {'Super Bowl 2017', 'Super Bowl 2016', 'Super Bowl 2015'} are some of its instances. Each event instance I_i^e has an occurrence date d_i^e associated with it.

We are provided $\mathcal{K}$, a knowledge base of recurrent events, where each recurrent event has a set of known instances and their occurrence dates. We assume all facts in $\mathcal{K}$ are correct, but it can be incomplete. For instance, in the KB of Figure 1, '2015 Independence Bowl' is missing its date, and LA Marathon has no known instances.

Additionally, we are provided $\mathcal{C}$, a corpus of unstructured text that contains information about recurrent events and instances. This combines documents such as news articles, encyclopedia where information about recurrent events and their specific instances may be mentioned.

We define the task of inferring temporal knowledge for near-periodic recurrent events as follows: given an incomplete knowledge base $\mathcal{K}$ and a corpus of unstructured text $\mathcal{C}$, build a model to simultaneously (1) infer schedule s for each recurrent event e in $\mathcal{K}$, (2) populate missing information in $\mathcal{K}$, and (3) predict the occurrence date for future instances of

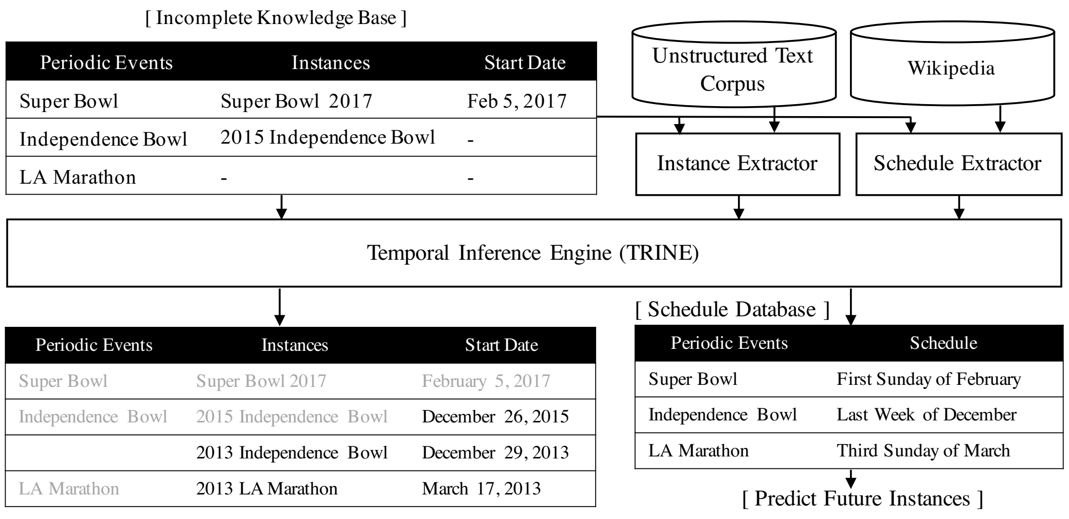

[Incomplete Knowledge Base]
Periodic Events | Instances | Start Date
Super Bowl | Super Bowl 2017 | Feb 5, 2017
Independence Bowl | 2015 Independence Bowl | -
LA Marathon | - | -
Unstructured Text Corpus
Wikipedia
Instance Extractor
Schedule Extractor
Temporal Inference Engine (TRINE)
Periodic Events | Instances | Start Date
Super Bowl | Super Bowl 2017 | February 5, 2017
Independence Bowl | 2015 Independence Bowl | December 26, 2015
2013 Independence Bowl | December 29, 2013
LA Marathon | 2013 LA Marathon | March 17, 2013
[Schedule Database]
Periodic Events | Schedule
Super Bowl | First Sunday of February
Independence Bowl | Last Week of December
LA Marathon | Third Sunday of March
[Predict Future Instances]

Figure 1: System overview for jointly inferring schedules and missing occurrence dates for recurrent events

Month Season (*MS*)	Date (*DT*)	Day/Week (*DW*)	Month/Season Modifier (MS_{mod})	Day/Week Modifier (DW_{mod})
JAN	1	MON	EARLY	1st
···	2	TUE	MID	2nd
DEC	3	···	LATE	3rd
SUMMER	4	SAT		4th
···	···	SUN		LAST
WINTER	31	WEEK		

Table 1: Atomic Elements for Schedule Representation

$$S \rightarrow C \mid (C \text{ or } S)$$

$$C \rightarrow MS \qquad C \rightarrow MS\ DT$$

$$C \rightarrow MS_{mod}\ MS \qquad C \rightarrow DW_{mod}\ DW \text{ of } MS$$

Figure 2: CFG Rules for Schedule Space Representation

the event. Our system consists of three major components as shown in Figure 1. Our key technical contribution is TRINE.

4 Temporal Extraction and Inference

We now discuss the technical details of our system. We first define a context free grammar for representing schedules, followed by descriptions of each individual component.

4.1 Schedule Space

Our definition of schedule space is inspired by the temporal type SET defined in [Pustejovsky *et al.*, 2003]. SET represents temporal expressions that suggest repetitive events (e.g. “every July”, “every 25th December”).

We introduce a representation for recurrent event schedules. The full grammar for schedule space is described in Table 1 and Figure 2. The non-terminal C constructs schedule expressions, with two rules on left representing month schedules, while the other two capture day and week schedules. We find that our CFG can represent almost all annual schedules. However, it cannot handle sub-annual (once in 4 years) or super-annual (e.g., weekly, fortnightly) events.

4.2 Schedule Extractor

Schedule extractor takes as input a list of known recurrent events from $\mathcal{K}$ and extracts their schedules from text in $\mathcal{C}$. Given a paragraph that contains a mention of a recurrent event e, schedule extractor returns its schedule s^e, if present.

Schedule extractor is a two-step rule-based system built on top of SUTIME [Chang and Manning, 2012]. The first step identifies temporal mentions in text and selects the ones labeled as temporal types DATE or SET with a month reference. Ideally, we should have used only SET annotations to extract periodicity, but we found them to have very low recall. We included DATE as sometimes a DATE label also suggest a schedule. For example, ‘January’ is labeled as DATE in “the event happens in January”. To improve precision, we also add a few additional rules to SUTIME to include temporal modifiers such as early, mid and late.

The second step normalizes the identified temporal mentions to map them to the schedule space representation. Temporal mentions such as “end January’ and “January end” are normalized to *LATE JAN* using TIMEX3 tags returned by SUTIME.

Empirically we find that the first schedule mentioned in text is almost always the one describing the event. Hence, the schedule extractor returns the first schedule mentioned in text, and discards other schedules, if any. On a handle-labeled test set, the schedule extractor has a precision of 0.83 and a recall of 0.79.

4.3 Instance Extractor

Instance extractor starts with a list of known recurrent events (and their instance names) from $\mathcal{K}$. Its goal is to detect instance mentions in text corpus $\mathcal{C}$ and extract occurrence dates for each mention.

It runs a three-step procedure. In the first step, the extractor identifies instance mentions and disambiguates it to a known event instance I^e, when possible. Second, it identifies candidate occurrence date d to extract (I^e, d) pairs. Finally, a binary classifier decides whether the candidate occurrence date is in fact an occurrence date for that instance or not.

Detecting mentions of recurrent events (and its instances) is easier than typical event extraction since most recurrent events are *named events*, and are often referred by their names or a handful of aliases, e.g., 'Super Bowl 2017' or 'Super Bowl LI'. If only the recurrent event is mentioned, then we disambiguate it to the year that is closest to the mention. For example, the mention in "... the 2011 edition of Independence Bowl is scheduled on 28th December" will be disambiguated to 'Independence Bowl 2011'.

In the second step, the extractor detects all date mentions in the paragraph mentioning the instance name using SUTIME. If there are multiple dates $\{d_1, d_2, ..., d_K\}$ mentioned, then the following pairs are generated: $\{(I^e, d_1), (I^e, d_2), ..., (I^e, d_K)\}$.

The final step is to run the instance-date pairs (I^e, d) through a classifier so that only plausible pairs are passed on to the inference engine. This step is similar to the Task-15A of SemEval 2007 [Verhagen *et al.*, 2007], which labels temporal relations holding between time and event expressions that occur within the same sentence. The knowledge base $\mathcal{K}$ is used as a source of distant supervision to automatically construct training data (similar to [Ji *et al.*, 2014]). We use standard NLP features like words, POS and NER tags of words around the date d, and dependency parse features. All features are generated using the Stanford CoreNLP pipeline [Manning *et al.*, 2014]. We train a max-entropy classifier with L1 regularization. Each extracted date is associated with a confidence. For multiple extractions of the same date, the confidences are aggregated using a Noisy-Or.

Previous work [Ji *et al.*, 2014] has separated temporal classification into four categories: START, END, HOLD (between START and END) and NULL (unrelated to event). However, often textual cues are insufficient to identify the correct temporal category. For example, in "It premiered at the YIFF on February 26", Feb 26 could be START, END or HOLD date. Our preliminary experiments on four-way classification returned a large number of errors.

As the output of instance extractor is aggregated by a downstream inference engine, we prefer a higher recall in the candidate pairs. As a consequence, we simply construct training set to separate NULL from {START, END, HOLD}. All positive labels are sent to TRINE. The date classifier has an estimated precision of 0.67 and a recall of 0.35 on 10 percent of the distant supervised data kept aside for test. These numbers are comparable to the slot filling accuracies [Surdeanu and Ji, 2014], where the slots are of the type DATE.

4.4 TRINE: Temporal Inference Engine

TRINE jointly infers missing occurrence dates in $\mathcal{K}$ and the schedules of recurrent events using the extractions mentioned above. The model must find the best schedule, even if the extracted schedule is too coarse. It must also be robust to off-schedule instances as well as to extraction noise.

We model the problem as inference in an undirected graphical model, shown in the Figure 3(a). Let S^e represent the random variable for the schedule of recurrent event e and D_i^e, the random variable for occurrence date of the instance I_i^e. For simplicity, we drop the superscript ^{e}, as we model only one recurrent event e at a time. The schedule S can take any value from the schedule space $\mathcal{S}$ and the occurrence date D_i can take any value from the day of year. Day of year is a number between 1 and 365/366 (e.g: "January 1" is day 1).

The joint model has three potentials: a unary potential, $\phi^{sch}(S)$, associated with the schedule S based on the schedules read. A unary potential, $\phi^{occ}(D_i)$, for the occurrence date of each instance based on the occurrence dates extracted and the knowledge base. Finally a pairwise potential, $\psi^{pair}(S, D_i)$, that represents how strongly the occurrence date D_i follows the schedule S. The factorization of the joint probability for this graphical model is as follows:

$$P(s, d_1, \cdots, d_N) = \frac{1}{Z}\phi^{sch}(s)\prod_{i=1}^{N}\psi^{pair}(s, d_i)\phi^{occ}(d_i)$$

where Z is the normalization factor and N is the total number of instances for e. Performing a joint MAP inference returns the best schedule and best occurrence date for each instance. In our experiments, we wish to return a ranked list of occurrence dates for each instance. We perform marginal inference to compute the posterior probability of each occurrence date for each instance, and rank them using the probabilities. Since there are only 365 candidate dates, and candidate schedules also make a small, enumerable set (size $\approx$1000), this is easy to compute.

We explain the potentials with the help of 'Super Bowl' as a running example. The three tables in Figure 3(c) represent the three knowledge sources. The first table shows the schedule extracted (*FEB*). The second shows the list of instances extracted for 'Super Bowl 2016' and 'Super Bowl 2015' with their extraction confidences. 'Super Bowl 2014' has no extractions but has its occurrence date in Freebase (shown in the third table). We now describe the three potentials.

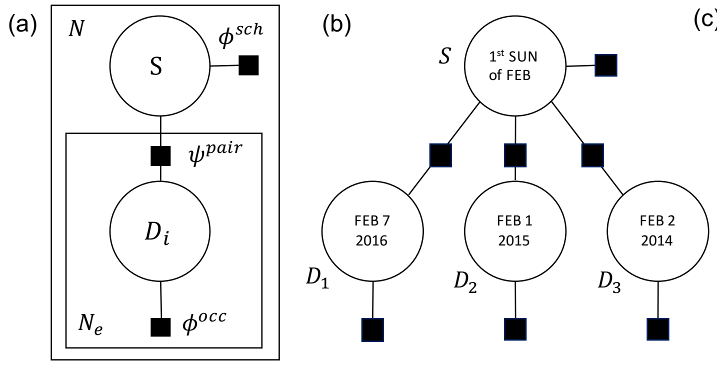
Schedule Potential: $\phi^{sch}(S)$, is defined using the extracted schedule s^{ex}. As it is common for schedules to be described loosely (e.g *FEB*) rather than precisely (e.g *1st SUN of FEB*), we define a subsumption relation between schedules, to use them in the potential. Let $sub(s^{ex})$ be the set of all schedules subsumed by s^{ex}. For example, if s^{ex} is *FEB*, then $sub(s^{ex})$ will contain {*FEB 1*, 1^{st} *SUN of FEB*, *EARLY FEB*, *etc*}. Thus the potential $\phi^{sch}(s)$ is given by

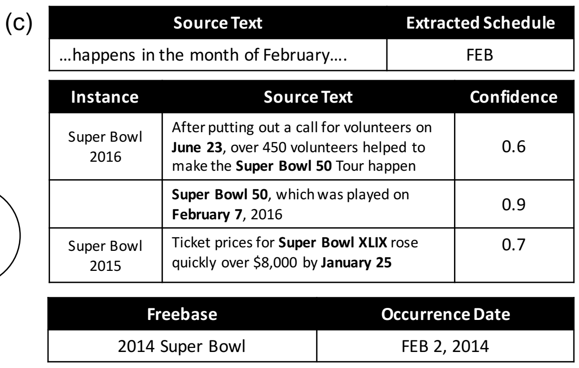
$$\phi^{sch}(s) = \begin{cases} \lambda, & \text{if } s = s^{ex} \\ \alpha \cdot \lambda, & \text{if } s \in sub(s^{ex}) \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

where λ and α are hyper parameters tuned using a dev set, with $\lambda > 1$ and $\alpha \in (\frac{1}{\lambda}, 1)$. Intuitively, this assigns the highest weight to the read schedules and reasonable weight to schedules subsumed by the read schedule.

For our running example, the schedule *FEB* is assigned the highest value λ in the schedule potential, and $\alpha \cdot \lambda$ is assigned to schedules subsumed by the extracted schedule such as *FEB 1*, 1^{st} *SUN of FEB*, *EARLY FEB*, *etc*.

Pairwise Potential: $\psi^{pair}(s, d_i)$ denotes the affinity between event's schedule s and an instance i's occurrence date d_i. We model each schedule s as a Normal distribution $\mathcal{N}(\mu_s, \sigma_s^2)$ over the days of the year. The mean of the schedule μ_s is set to the day corresponding to the mid point of the




(c)

Source Text	Extracted Schedule
…happens in the month of February….	FEB

Instance	Source Text	Confidence
Super Bowl 2016	After putting out a call for volunteers on **June 23**, over 450 volunteers helped to make the **Super Bowl 50** Tour happen	0.6
	Super Bowl 50, which was played on **February 7**, 2016	0.9
Super Bowl 2015	Ticket prices for **Super Bowl XLIX** rose quickly over $8,000 by **January 25**	0.7

Freebase	Occurrence Date
2014 Super Bowl	FEB 2, 2014

Figure 3: (a) Plate notation of TRINE (b) an example instantiation for Super Bowl with (c) knowledge from the three sources

schedule range. For example, the mean of a month schedule, is set to the day corresponding to the 15^{th} of the month (14^{th} for February). The standard deviation of the schedule σ_s is set such that the start/end of a schedule is $3\sigma_s$ away from the mean. The pairwise potential is given by the likelihood of the occurrence date d_i in the schedule distribution.

Occurrence Potential: The occurrence potential assigns highest weight for known dates in the knowledge base, and lower weight to the extracted dates. Let $\mathbb{D}_i = \{d_{ik}\}$ be a set of candidate occurrence dates extracted by the instance extractor for instance I_i. Let c_{ik} be the confidence associated with extraction d_{ik}. Let d_i^* be the occurrence date of I_i, if it is present in the knowledge base. The confidence associated with d_i^* is set to 1.0. We compute $\phi^{occ}(d_i)$ as

$$\phi^{occ}(d_i) = \begin{cases} c_{ik}, & \text{if } d_i = d_{ik}, d_{ik} \in \mathbb{D}_i \\ 1, & \text{if } d_i = d_i^* \\ \rho, & \text{otherwise} \end{cases} \tag{2}$$

where ρ is the noise parameter. When $\rho = 0$, one of the extractions will necessarily get picked for each instance. This works if each instance has its correct date in the candidate set. This is rarely true in practice. $\rho > 0$ adds flexibility to the model by allowing it to ignore *all* extracted candidates from the instance extractor.

For the Super Bowl running example, the occurrence potential of ‘Super Bowl 2016’ has all days of the year in 2016 (1-366). Day 175 (June 23, 2016) is assigned 0.6, day 38 (February 7, 2016) is assigned 0.9 and all other (unextracted) days of the year are assigned ρ. For 2014, all days are assigned ρ except day 33 (Feb 2, 2014) which is assigned 1.0, since it is present in the knowledge base.

The noise parameter has a crucial advantage. Consider that the model is initialized with only start dates from FreeBase and the recurrent event has an instance that deviated from its typical schedule. Such aberrations are common in recurrent events. For example, let 3 out of 4 instances of an event happened on *1st SAT of FEB* and one on *1st THU of FEB*. Here, the model with $\rho = 0$ would favor *1st WEEK of FEB* (a broader schedule) rather than the most likely narrower schedule: *1st SAT of FEB*. We make the model robust by allowing it to ignore a few instances in favor of a narrower pattern. ρ represents the penalty to pay for ignoring an instance.

Let $s \in S$ be a schedule with mean μ_s and standard deviation σ_s, d be the day with highest value for $\phi^{occ}(d)\psi^{pair}(s, d)$, and c be the confidence value associated with d and d_μ be the day corresponding to μ_s. Since d_μ has the highest value for ψ^{pair}, the model chooses between the day with the highest confidence and the mean of the schedule, based on ρ. To be more specific, the mean is chosen if

$$\begin{gathered} \phi^{occ}(d_\mu)\psi^{pair}(s, d_\mu) > \phi^{occ}(d)\psi^{pair}(s, d) \\ |d - \mu_s| > \sqrt{2 \log \tfrac{c}{\rho}}\ \sigma_s \end{gathered} \tag{3}$$

We tune ρ using a dev set and set it to 0.1. This provides a performance boost of 75% in our experiments (Table 4).

Figure 3(b) shows the instantiation of TRINE for ‘Super Bowl’ along with knowledge from the three sources. For simplicity we show only three instances, ‘Super Bowl 2016’, ‘Super Bowl 2015’ and ‘Super Bowl 2014’, with their corresponding occurrence date represented by D_1, D_2 and D_3.

The values estimated using MAP inference over the network are shown inside the circles in Figure 3(b). TRINE predicts the schedule as 1^{st} *SUN of FEB* – precise and fine grained, compared to that extracted from text. TRINE is able to ignore *JUN 23* and picks *FEB 7* for D_1. Due to the noise parameter, the extraction *JAN 25* is completely ignored and D_2 is assigned an occurrence date based on the schedule.

5 Experiments

Our experiments answer the following research questions. (1) How dependable is TRINE in predicting future events compared to humans and natural baselines? (2) How accurate is TRINE in performing knowledge base completion? In addition to this, we also perform an ablation study to showcase the incremental contribution of each knowledge source to the overall performance of the system.

Data: We require a knowledge base $\mathcal{K}$ of recurrent events and a corpus $\mathcal{C}$ of unstructured text as input. We choose Freebase as the knowledge base, as it had more recurrent events compared to DBPedia or YAGO. It lists 15,770 recurrent events, but only 4,350 of these have at least one instance with its occurrence date. We divide these 4,350 recurrent events into two equal sets: train and test. The instances and their occurrence dates in the train set are used as a source of distant supervision to train the instance extractor. A small fraction of the events from the train set are used as dev set for feature selection and hyper parameters tuning. The test set is used

	MRR	Acc@1	Acc@3	Acc@5
Mode Baseline	0.096	0.019	0.094	0.264
Mean Baseline	0.147	0.075	0.17	0.32
AMT	0.308	0.241	0.366	0.41
TRINE	**0.388**	**0.292**	**0.434**	**0.575**

Table 2: Future occurrence date prediction performance for 2016 instances of 106 recurrent events

for the ablation study. In addition to this, we also manually collected the occurrence dates for 500 instances missing in Freebase, to evaluate TRINE's performance.

Each Freebase entity has a description field and a link to the its English Wikipedia page. Each recurrent event's description field and the first paragraph of its Wikipedia page are fed as input to the schedule extractor. We use the New York Times Corpus [Sandhaus, 2008] and a part of *ClueWeb12*[7] as the input text corpora for the instance extractor.

Evaluation Metric: Each algorithm generates a ranked list of predictions for missing occurrence dates. Hence we use standard ranking metrics like Mean Reciprocal Rank and Accuracy@k as the evaluation metrics.

5.1 Future Instance Prediction

The objective of this experiment is to see how TRINE compares with humans in the task of predicting the schedule and inferring occurrence dates of future instances. This helps in testing the schedule space formulation and TRINE's ability to infer schedules. We made both TRINE and Amazon Mechanical Turk workers predict the occurrence date of 2016 event instances. We anonymized the recurrent events to prevent AMT workers from looking up external resources rather than inferring them. For fairness, we provide TRINE the same knowledge we provide AMT workers i.e., no instance or schedule extractions, as they are extracted using the event/instance names.

Given the occurrence date of the previous historical instances, AMT workers were requested to predict the 2016 occurrence date. They were also provided the day of the week for each occurrence date and were advised to use a calendar. Each worker made five guesses ordered by confidence. Three sets of annotations were collected for each recurrent event.

We also compare TRINE against two baselines: *Mean Baseline* and *Mode Baseline*. To compute these we first represent each occurrence date by the day of the year. For example, 25^{th} Dec, 2015 will be represented as 359. The mean baseline uses the mean of historical occurrence dates, whereas the mode baseline uses the mode. To generate a ranked list, the mode baseline simply orders each historical date by frequency in decreasing order. The mean baseline orders by deviation from the mean, with the two days on either side of the mean randomly ranked 2nd and 3rd, and so on.

For this experiment, we randomly sampled 106 recurrent events from Freebase with at least 3 historical instances each. As 2016 instances were not present in Freebase, we manually collected their occurrence dates from the Web. Table 2 shows the performance of AMT workers, TRINE and other baselines. TRINE considerably outperforms the baselines and is also much superior to that of the AMT workers. In 57% of the cases, one of the top five predictions is correct for TRINE, whereas this number is only about 41% for AMT workers.

[7]http://lemurproject.org/clueweb12/

	Instances Filled	MRR	Acc@1	Acc@5
Instance Extractions + Freebase	1580	0.508	0.425	0.653
Instance Extractions + No Freebase	2893	0.432	0.349	0.514
No Instance Extractions + Freebase	25573	0.257	0.172	0.363

Table 3: Performance of TRINE on Freebase Completion

The problem is inherently difficult because most of the recurrent events do not follow a perfect schedule. For example, the past instances of '24 Hours of Le Mans' occurred on the following dates: 2015-06-13, 2014-06-14 2013-06-22 and 2012-06-16. The 2015, 2014 instances took place on the *2nd SAT of JUN*, the 2013 on *4th SAT of JUN* and the 2012 on *3rd SAT of JUN*. Both TRINE and AMT users had *2nd*, *3rd*, and *4th SAT* in the top 3, with *2nd* at the top. The occurrence date represented as day of the year were: 164, 165, 173 and 168. The mean baseline's top prediction was day 167 in 2016 (Wednesday). As all the occurrence dates are different from each other, the mode would randomly rank them. This showcases the importance of explicitly modeling schedules.

The reason behind overall low MRRs for AMT workers and TRINE is the lack of a dominant schedule. In the previous example, even though the instances didn't follow a perfect schedule, there is a dominant schedule, which most of the instances followed. For such recurrent events, it is hard for both TRINE and AMT workers to infer a schedule and predict the future instance.

5.2 FreeBase Completion

To analyze the performance of TRINE for filling missing instances in FreeBase, we split the missing instances in FreeBase based on the knowledge sources available for predicting the occurrence: instance extractions and FreeBase.

Has Instance Extractions: These are instances for which the instance extractor has at least one extraction.

Has FreeBase Knowledge: All instances of events that have at least one instance with an occurrence date in Freebase.

The second column in Table 3 corresponds to the number of missing freebase instances filled in each category. To evaluate the performance, we sample 100 random instances from each category and manually collect the true occurrence dates.

When instance extractions are present, we find that TRINE's top prediction is correct about 42% of the time. Provenance highlighting of the text where it was extracted is a straightforward extension to our system. We believe that having a human ontologist vet the predictions will significantly improve the coverage of Freebase. For other cases, we believe that identifying larger corpora where these events will be mentioned and running our system through those will produce similar results.

For cases where instance extraction is not present, it is hard to ascertain the exact occurrence date. In such cases, our system provides a reasonable guess, which can definitely be used for any downstream task.

	MRR	Acc@1	Acc@5
TRINE	**0.415**	**0.308**	**0.536**
TRINE w/o schedules	0.41	0.307	0.534
TRINE w/o instances	0.361	0.261	0.471
TRINE w/o Freebase	0.188	0.133	0.233
TRINE with $\rho = 0$	0.237	0.159	0.301

Table 4: Contribution of each knowledge source to TRINE

5.3 Contribution of Knowledge Sources

To understand the contribution of each knowledge source, we measure TRINE's performance by dropping one source at a time. We refer to these models as *TRINE w/o knowledge source*.

We perform this experiment on the 2,175 recurrent events in the test set. We randomly remove one instance from each test event. Out of 2,175 events, 1,566 had at least one knowledge source available for predicting the held out instance.

The MRR and precision numbers for various settings of TRINE are shown in the Table 4. It is evident that TRINE with all knowledge sources combined performs the best. Comparatively speaking, schedules are less valuable than other sources, as they are seldom mentioned precisely in the text. For example, even though 'Chennai Open' follows the schedule *1st MON of JAN*, Wikipedia expresses it as "It is held annually in January in Chennai, Tamil Nadu, India."

We also note the dramatic decrease in performance when we force ρ, the noise parameter, to zero. This underscores the importance of providing the model with flexibility to correct errors from other components.

The major difficulty for TRINE is the lack of a dominant schedule (as described in Section 5.1). Other issues include (1) often the actual event happens a few days before or after the precise schedule, and (2) shift in the schedule over the years. For example, the Miami International Film Festival was held in *FEB* before 2006, but since then has been shifted to *EARLY MAR*.

6 Conclusions

Near-periodic recurrent events are an important class of events that have not received much attention in the NLP literature. We present the first system for predicting occurrence dates of event instances using information extracted from text, as well as information about known historical instances of the same event present in a knowledge-base. A key component is a joint inference module that uses inference over an undirected probabilistic graphical model to estimate the best schedule for the event as well as occurrence date for each event instance.

In an experiment over AMT, we find that TRINE beats humans in guessing the future occurrence date of an event. We also show that in cases when textual extractions are available, we can fill in missing dates in FreeBase at an acceptable MRR of over 0.5. However, performance degrades when less information about an event is available. Our work can easily create a system that can be put within the loop with a human expert, who can verify system predictions and add important information to the knowledge-base. We release our source code, datasets, and system predictions for further research at *https://github.com/dair-iitd/trine*.

Acknowledgments

This work is supported by Google language understanding and knowledge discovery focused research grants, a Bloomberg award, an IBM SUR award, and a Visvesvaraya faculty award by Govt. of India to the third author. We thank Microsoft Azure sponsorships, and IIT Delhi HPC facility for computational resources. The idea of this work came together through discussions with Biplav Srivastava about tourism-related events. We thank Abhishek Bansal for his early work on the project. We also acknowledge the IBM Research India PhD program, which enables the first author to pursue a PhD at IIT Delhi.

References

[Adel *et al.*, 2016] Heike Adel, Benjamin Roth, and Hinrich Schutze. Comparing convolutional neural networks to traditional models for slot filling. In *HLT-NAACL*, 2016.

[Bethard and Martin, 2006] Steven Bethard and James H. Martin. Identification of event mentions and their semantic class. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 146–154, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[Chang and Manning, 2012] Angel X. Chang and Christopher D. Manning. Sutime: A library for recognizing and normalizing time expressions. In *In LREC*, 2012.

[Craswell, 2009] Nick Craswell. Mean reciprocal rank. In *Encyclopedia of Database Systems*, 2009.

[Do *et al.*, 2012] Quang Xuan Do, Wei Lu, and Dan Roth. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 677–687. Association for Computational Linguistics, 2012.

[Foley *et al.*, 2015] John Foley, Michael Bendersky, and Vanja Josifovski. Learning to extract local events from the web. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 423–432, New York, NY, USA, 2015. ACM.

[Intxaurrondo *et al.*, 2015] Ander Intxaurrondo, Eneko Agirre, Oier Lopez de Lacalle, and Mihai Surdeanu. Diamonds in the rough: Event extraction from imperfect microblog data. In *HLT-NAACL*, 2015.

[Ji *et al.*, 2014] Heng Ji, Taylor Cassidy, Qi Li, and Suzanne Tamang. Tackling representation, annotation and classification challenges for temporal knowledge base population. *Knowl. Inf. Syst.*, 41(3):611–646, December 2014.

[Konovalov *et al.*, 2017] Alexander Konovalov, Benjamin Strauss, Alan Ritter, and Brendan O'Connor. Learning to extract events from knowledge base revisions. In *Proceedings of the 26th International Conference on World Wide*

Web, WWW '17, pages 1007–1014, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[Kunneman and Van den Bosch, 2015] Florian Kunneman and Antal Van den Bosch. Automatically identifying periodic social events from twitter. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 320–328, 2015.

[Kuzey *et al.*, 2014] Erdal Kuzey, Jilles Vreeken, and Gerhard Weikum. A fresh look on knowledge bases: Distilling named events from news. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1689–1698, New York, NY, USA, 2014. ACM.

[Manning *et al.*, 2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[Patwardhan and Riloff, 2009] Siddharth Patwardhan and Ellen Riloff. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 151–160, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Pustejovsky *et al.*, 2003] James Pustejovsky, José M. Castaño, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. Timeml: Robust specification of event and temporal expressions in text. In Mark T. Maybury, editor, *New Directions in Question Answering*, pages 28–34. AAAI Press, 2003.

[Reschke *et al.*, 2014] Kevin Reschke, Martin Jankowiak, Mihai Surdeanu, Christopher Manning, and Daniel Jurafsky. Event extraction using distant supervision. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), may 2014.

[Ritter *et al.*, 2012] Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1104–1112, New York, NY, USA, 2012. ACM.

[Sandhaus, 2008] E. Sandhaus. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12), 2008.

[Surdeanu and Ji, 2014] Mihai Surdeanu and Heng Ji. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. 2014.

[Surdeanu *et al.*, 2011] Mihai Surdeanu, Sonal Gupta, John Bauer, David McClosky, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. Stanford's distantly-supervised slot-filling system. In *TAC*, 2011.

[Surdeanu, 2013] Mihai Surdeanu. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*, 2013.

[Talukdar *et al.*, 2012] Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Coupled temporal scoping of relational facts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 73–82. ACM, 2012.

[Verhagen *et al.*, 2007] Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 75–80, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[Zhang *et al.*, 2015] Congle Zhang, Stephen Soderland, and Daniel Weld. Exploiting parallel news streams for unsupervised event extraction. *Transactions of the Association for Computational Linguistics*, 3:117–129, 2015.