Acquiring Temporal Constraints between Relations

Partha Pratim Talukdar*
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
ppt@cs.cmu.edu

Derry Wijaya*
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
dwijaya@cs.cmu.edu

Tom Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
tom.mitchell@cs.cmu.edu

ABSTRACT

We consider the problem of automatically acquiring knowledge about the typical temporal orderings among relations (e.g., *actedIn*(person, film) typically occurs *before wonPrize* (film, award)), given only a database of known facts (relation instances) without time information, and a large document collection. Our approach is based on the conjecture that the narrative order of verb mentions within documents correlates with the temporal order of the relations they represent. We propose a family of algorithms based on this conjecture, utilizing a corpus of 890m dependency parsed sentences to obtain verbs that represent relations of interest, and utilizing Wikipedia documents to gather statistics on narrative order of verb mentions. Our proposed algorithm, GraphOrder, is a novel and scalable graph-based label propagation algorithm that takes transitivity of temporal order into account, as well as these statistics on narrative order of verb mentions. This algorithm achieves as high as 38.4% absolute improvement in F1 over a random baseline. Finally, we demonstrate the utility of this learned general knowledge about typical temporal orderings among relations, by showing that these temporal constraints can be successfully used by a joint inference framework to assign specific temporal scopes to individual facts.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; I.2.6 [**Artificial Intelligence**]: Learning; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms, Experimentation

*Contributed equally.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

Keywords

Knowledge Bases, Temporal Ordering, Temporal Scoping, Narrative Ordering, Graph-based Semi-Supervised Learning, Label Propagation

1. INTRODUCTION

Harvesting temporal knowledge from Web sources is an important research challenge [23]. Web search and Question Answering (QA) systems can benefit from having knowledge about entities, their relationships, and the time at which the relationships hold (i.e., their time scopes) [1]. A closely related task to time scoping is temporal ordering of relations. Instead of finding the temporal scopes of relation instances (or simply, facts), temporal ordering finds order (e.g., *before, simultaneous*, etc.) between facts. For example, while temporal scoping aims to infer that *presidentOf(Bill Clinton, USA)* was true during 1993 - 2001, temporal ordering at the relation level aims to infer that, for the same person, *wasBornIn(person, location)* happens *before presidentOf(person, country)*. Knowledge of temporal order can be useful for temporal scoping. For example, if we know that Al Gore's vice presidency was *simultaneous* with Bill Clinton's presidency, and if we know the time scope of Clinton's presidency, then we can infer the time scope of Gore's vice presidency.

We address the problem of inferring temporal ordering of relations at the macro (corpus) level: a novel, important, and unexplored problem. Previous research has mostly focused on temporal ordering at the micro level (i.e., sentence or intra-document level), and that too over verbs (or single-verb events). Instead of taking a verb-centric view, we aim to temporally order relations, where each relation may be expressed by different verbs in different sentences. For example, the $actedIn(person, film)$ relation can be expressed by the verbs *acted*, *starred*, and many others.

This paper explores the feasibility of inducing *typical* temporal orderings between relations based on the narrative order of inferred mentions of these relations, averaged across many documents. We define the narrative order of relation mentions in a document as the textual order of sentences that contain verbs expressing these relations. We explore the conjecture that narrative order is correlated with temporal order frequently enough (especially when aggregated across a large number of documents) for it to serve as a probabilistic training signal for learning the temporal order. Such conjecture is appropriate especially in documents that

are each about an entity and the relations incident on the entity. In such document, relations are frequently mentioned around the entity in chronological order. Wikipedia is an example of a large and broad coverage corpus containing such documents.

Our temporal order of relations is related to structured sequences of participants and events in documents that have been called scripts [18] or narrative event chains [9], which are applied to or inferred from the narrative in the document. Since scripts are inherently temporal, e.g. restaurant script, employment script, or kidnapping script; we conjecture that narrative can also be useful for temporal ordering.

Our Contributions: We consider the problem of learning *typical* temporal ordering between *relations* (e.g., *actedIn*(person, film) happens *before wonPrize*(film, award)), given a database of macro-read facts (*relation instances*) such as Yago2 [12] without time information. We present GraphOrder, a graph-based label propagation algorithm for this task and demonstrate in our experiments that the narrative order of *mentions* of these relation instances (i.e., verbs expressing the relations), aggregated over a large number of documents, can be exploited as probabilistic training signals to infer the temporal ordering of the relations. We also mine syntactic information in the form of subject-verb-object triples from a large corpus of 890m dependency parsed sentences[1] and show how such information can be effectively used for finding these *relation mentions*. Through experiments on several domains of real-world datasets, we show the effectiveness of our method. We also propose $\text{CoTS}_{\text{Soft}}$, which extends a recent collective inference framework for temporal scoping [20], to handle soft temporal ordering constraints generated by GraphOrder. We incorporate our automatically acquired constraints into this framework and show the effectiveness of our learned temporal order constraints for temporal scoping of relation instances.

2. PROBLEM DEFINITION

Here we introduce our terminology and define the problem considered in this paper. We use the term *relation* to refer to any general typed binary relation r (e.g., *directed(person, film)* whose two arguments are of type *person* and *film*), and the term *relation instance* (or simply, *fact*) to refer to any instance of that relation (e.g., directed(George Lucas, Star Wars)). We say that a verb v expresses a relation r if mentions of v in documents frequently represent facts of relation r, with $\text{RelVerb}(r, v)$ measuring the strength of this association. $\text{Default}(r)$ is a function which returns a single default (or representative) verb (extracted from the *relation name*) which represents relation r. For example, Default(actedIn) = "act".

We use the term *narrative order* between verbs v_1 and v_2 within a document collection to refer to statistics about the sequence in which v_1 and v_2 are mentioned within the documents: we use $\text{NBefore}(v_1, v_2)$ to measure how often v_1 occurs before v_2 in a text narrative, and use $\text{NSimultaneous}(v_1, v_2)$ to measure how often v_1 occurs in the same sentence (i.e., simultaneously) as v_2 in the text narrative. We use the term *temporal order* between relations r_1 and r_2 to refer to the temporal sequence in which these relations occur in the real world. Specifically, we use $\text{TBefore}(\text{r}_1, \text{r}_2)$ to represent that facts of relation r_1 *typically* occur temporally before facts of r_2 involving at least one shared argument. Similarly, we use $\text{TSimultaneous}(\text{r}_1, \text{r}_2)$ to denote that facts of these two relations *typically* occur at the same time (i.e., there is a significant overlap between their temporal spans). We note that while TSimultaneous is a symmetric relation, TBefore is not. Our temporal orders are partial and uncertain (or soft). Given that many relations have only probabilistic temporal precedence, the non-negative values of $\text{TBefore}(\text{r}_1, \text{r}_2)$ and $\text{TSimultaneous}(\text{r}_1, \text{r}_2)$ are intended to capture a degree of confidence that increases monotonically with this probability. In this paper, we consider the following problem:

Problem Statement: Given (1) a set of typed binary relations R with one common argument type, which we shall refer to as the *domain*; (2) for each relation $r \in R$, a set of facts Q_r; and (3) an unlabeled text corpus C, where each document in the subset $D \subset C$ describes one instance from the *domain*; we would like to estimate the temporal relationship for each pair of relations in R, i.e., estimate values for $\text{TBefore}(\text{r}_\text{i}, \text{r}_\text{j})$, $\text{TSimultaneous}(\text{r}_\text{i}, \text{r}_\text{j}) \in \mathbb{R}_+$, $\forall r_i, r_j \in R$, where higher values represent higher confidence that the corresponding temporal relationship holds.[2]

Example: Let $R = \{actedIn(person, film)$, $directed(person, film)$, $wonPrize(film, award)\}$ be the set of typed binary relations, with common argument type *film*. Hence, *film* will also be the the domain in this case. Let the set of facts for each relation be as follows: $Q_{actedIn} = \{$(Tom Hanks, Forrest Gump)$\}$, $Q_{directed} = \{$(Robert Zemeckis, Forrest Gump)$\}$, and $Q_{wonPrize} = \{$(Forrest Gump, Academy Awards)$\}$. Let D be a singleton set consisting of the Wikipedia page on the film "Forrest Gump", and let C be its superset consisting of additional documents sampled from a large text corpus such as [7]. Given these as input, we would like to determine the temporal relationship (i.e., estimate values for TBefore and TSimultaneous) for each pair of relations in R.

Please note that since our inputs do not consist of any labeled data with temporal orderings annotated (as in [17]), our proposed method is unsupervised with respect to this task.

3. METHOD

3.1 Overview

The central thesis behind the method introduced here is that: "*Statistics aggregated from a large number of documents about narrative orders of verbs representing two relations can serve as probabilistic training signals for learning the temporal order between the two relations.*"

Let us consider two relations incident on *film* instances: *directed* and *wonPrize*, whose default verbs are "directed" and "won," respectively. If in documents discussing *film* instances, we predominantly observe the narrative order *directed* $\rightarrow$ *won*, i.e., $\text{NBefore}(directed, won) \gg \text{NBefore}(won, directed)$, this should increase our estimate of $\text{TBefore}(directed, wonPrize)$. In other words, $\text{NBefore}(directed, won)$, which can be estimated directly from free text, can be exploited to estimate the unobserved $\text{TBefore}(\text{directed}, \text{wonPrize})$ (and similarly for TSimultaneous). Based on the thesis mentioned above, our proposed approach consists of three phases, which are outlined in Figure 1. We next describe each of these three phases.

[1]To the best of our knowledge, this is one of the largest parsed corpus ever used for temporal extraction task.

[2]$\mathbb{R}_+$ is the set of non-negative real numbers

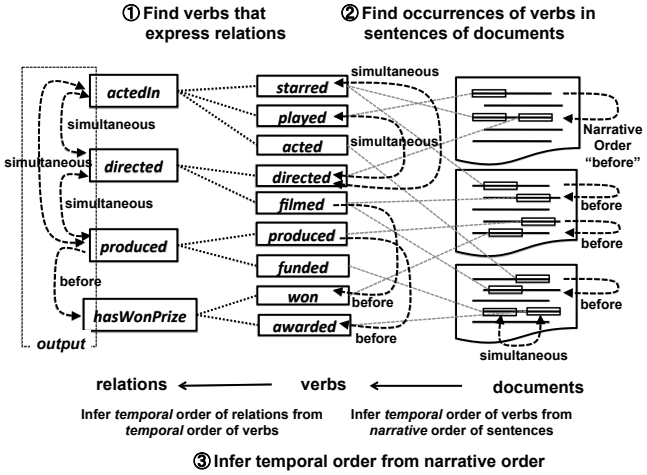


Figure 1: Outline of our proposed method which consists of three phases (see Section 3.1 for an overview). In the first phase, each relation is represented by a set of verbs (Section 3.2); in the second phase, the narrative order over verb mentions are estimated (Section 3.3); and in the final phase, the temporal order for each relation pair is estimated (Section 3.4).

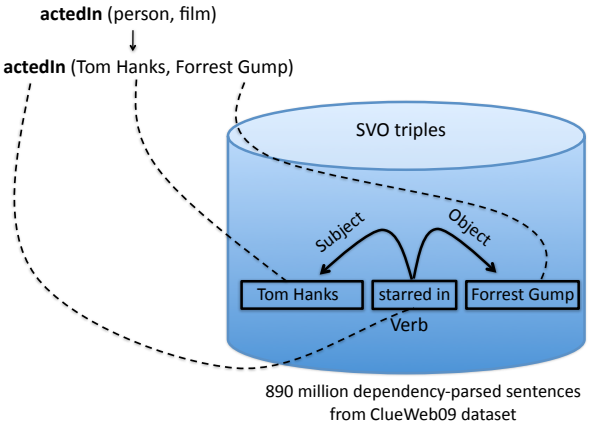


Figure 2: Finding verbs that express relations by mining a collection of subject-verb-object (SVO) triples extracted from 890m dependency parsed sentences (Section 3.2).

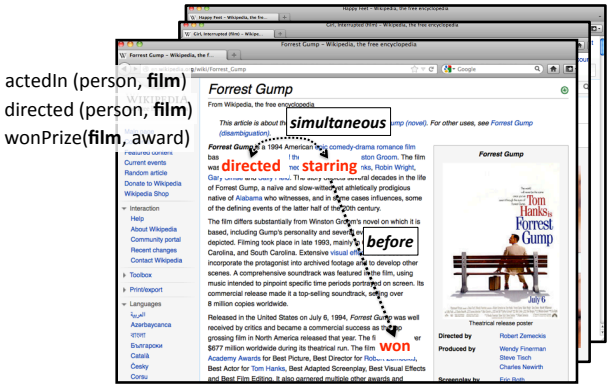


Figure 3: Narrative orders of verbs (e.g., *directed, starring, won*, etc.) in documents from the *film* domain (Section 3.3). Each document describes one instance (e.g., "Forrest Gump") from the domain (*film* in this case). These narrative orders provide guidance for learning temporal order of relations (e.g., *actedIn(person, film)*, *directed(person, film)*, *wonPrize(film, award)*) (Section 3.4).

3.2 Find Verbs that Express Relations

To identify verbs that express a particular relation r, we consider the given set of input facts Q_r representing known instances of relation r. Each fact $f \in Q_r$ is a triple, $\langle f_{a1}, f_{a2}, f_{rel} \rangle$ containing the values of the first argument, second argument, and the name of relation (in this case r), respectively. An example of a fact is ⟨Tom Hanks, Forrest Gump, actedIn⟩[3]. We draw on work on distributional similarity between verbs [10] and other means of grouping verbs that share the same lexical items in subject/object positions [6]. In particular, for each relation r we identify verbs whose mentions occur with subject/object pairs that match the known instances of relation r in Q_r.

[3]Alternatively, we also write it as actedIn(Tom Hanks, Forrest Gump)

More specifically, to identify the verbs associated with relation r (as illustrated in Figure 2), we use a large collection of Subject-Verb-Object (SVO) triples which can be compared to the facts Q_r about relation r. We construct this dataset of SVO triples by first parsing 50m Web documents (890m sentences, 16B tokens) using the MALT dependency parser [16], and then extracting SVO triples from these parsed sentences[4]. We aggregate all the SVO triples and count their frequency using Hadoop. This yields in a dataset with 114m triples, which we shall refer to as S.

[4]Details on the SVO triple extraction will appear in a longer version of this paper.

Fields within each tuple $s \in S$ can be accessed using the following: s_{sbj}, s_{obj}, s_{verb}, and s_{cnt}, which return the subject, object, verb, and the count of the tuple, respectively. To the best of our knowledge, this is one of the largest dependency parsed corpora that has ever been used in published research[5].

Let $S_v = \{s \mid s_{verb} = v,\ s \in S\}$ be the set of tuples in S with v as its verb. Also, let $U_S = \{u \mid |S_u| > 0\}$ be the set of unique verbs in S, and let U be the overall set of unique verbs under consideration. We have,

$$U = U_S \cup \{\text{Default}(r) \mid r \in R\} \tag{1}$$

We now compute $\text{RelVerb}(r, v)$ to measure whether verb $v \in U$ expresses relation r as follows:

$$\text{RelVerb}(r, v) = \frac{I_1(r, v) + \sum_{f \in Q_r, s \in S_v} I_2(f, s) \times s_{cnt}}{\sum_{u \in U} \left[I_1(r, u) + \sum_{f \in Q_r, s \in S_u} I_2(f, s) \times s_{cnt} \right]} \tag{2}$$

where $I_1(r, u) = 1$ iff $\exists r \in R$ such that $u = \text{Default}(r)$, and 0 otherwise; and $I_2(f, s)$ is another indicator function which returns 1 whenever arguments of f and s match, i.e., $(f_{a1}, f_{a2}) = (s_{sbj}, s_{obj})$ or $(f_{a1}, f_{a2}) = (s_{obj}, s_{sbj})$, and 0 otherwise. We now define F_r, the set of verbs used to express relation r as follows,

$$F_{\delta,r} = \{v \in U \mid \text{RelVerb}(r, v) > \delta\}$$

where $\delta \in \mathbb{R}$ is a threshold, which is set to 0 for the experiments in this paper. Examples of the top-5 elements from the set $F_{\delta,actedIn}$ relation, are as follows: stars, features, 's, costars, starring. This suggests that this method is able to automatically discover meaningful verbs to express a given relation.

3.3 Estimating Narrative Order of Verbs

Once we find the set of verbs that express relations, we find mentions of these verbs in documents in D, as illustrated in Figure 3. Since each document in D is about a domain instance (Section 2), we make a simplifying assumption that verbs appearing in the same document are sharing the domain instance as subject/object argument. In the future, a dependency parse of sentences in the document can be used to find verbs sharing the same domain instance as subject/object.

We shall represent a document $d \in D$ as a sequence of sentences sorted as per the narrative order, with $|d|$ denoting the total number of sentences in the document, and $d_k (1 \leq k \leq |d|)$ returning the k^{th} sentence in document d.

Now, for each sentence d_k, we compute the sentence score (SS) of each (r, v) pair mentioned in the sentence.

$$\text{SS}(d_k, r, v) = \frac{\text{VerbCount}(d_k, r, v)}{\sum_{r \in R, v \in U} \text{VerbCount}(d_k, r, v)} \tag{3}$$

where $\text{VerbCount}(d_k, r, v)$ is the number of times verb v is used to express relation r in sentence d_k. This can be estimated by matching the arguments of v in d_k against Q_r, the set of facts from relation r. However, we shall make a simplifying assumption and increment verb count by 1 whenever we come across v in d_k, subject to the condition that $v \in F_{\delta,r}$ (increasing δ may improve precision of verbs expressing a relation, but may affect recall). The scoring in Equation 3 will discount scores for verbs which express a large number of relations.

We then define the narrative order scores for two verbs v_1 and v_2 in a document d as the sum of scores (SS) of v_1 and v_2 in all pairs of sentences for which v_1 is before v_2, and the sum of scores of v_1 and v_2 in all sentences where v_1 and v_2 both occur. We then aggregate these per-document narrative orders across all documents containing v_1 and v_2 by taking the average of these per-document scores as the final narrative order scores $\text{NBefore}(v_1, v_2)$ and $\text{NSimultaneous}(v_1, v_2)$ for the verbs v_1 and v_2.

We note that Rel-grams [2], a recently proposed semantic resource, may also be used to estimate NBefore, although it is less clear how this resource can be used to estimate NSimultaneous.

3.4 Estimating Temporal Order of Relations

Once we find narrative order of verbs as described in Section 3.3, we infer temporal order of relations expressed by these verbs using two methods. The first, Pairwise (Section 3.4.1), is a non-transitive method that determines the order for two relations r_i and r_j by making pairwise decisions. The other method, **GraphOrder** (Section 3.4.2), uses a novel graph-based label propagation algorithm to temporally order relations in a joint optimization, and also while taking transitivity into account.

Because we are the first to address the problem of temporally ordering relations, there are no existing techniques that we could compare to. In our experiments, we compare the resulting temporal order of relations to a random baseline inspired by [9], which addresses a related but different problem. We also use our non-transitive method, Pairwise, as baseline to compare to GraphOrder, our proposed transitive graph-based method.

3.4.1 Pairwise Temporal Ordering of Relations

We now define scores for the the temporal order relations TBefore and TSimultaneous as follows:

$$\text{TBefore}(\text{r}_\text{i}, \text{r}_\text{j}) = \frac{1}{Z} \sum_{u,v \in U} \text{RelVerb}(r_i, u) \times \text{NBefore}(u, v) \times \text{RelVerb}(r_j, v) \tag{4}$$

$$\text{TSimultaneous}(\text{r}_\text{i}, \text{r}_\text{j}) = \frac{1}{Z} \sum_{u,v \in U} \text{RelVerb}(r_i, u) \times \text{NSimultaneous}(u, v) \times \text{RelVerb}(r_j, v) \tag{5}$$

where $\text{TSimultaneous}(\text{r}_\text{i}, \text{r}_\text{j}) = \text{TSimultaneous}(\text{r}_\text{j}, \text{r}_\text{i})$ and $Z = \text{TBefore}(\text{r}_\text{i}, \text{r}_\text{j}) + \text{TBefore}(\text{r}_\text{j}, \text{r}_\text{i}) + \text{TSimultaneous}(\text{r}_\text{i}, \text{r}_\text{j})$ is the normalizing factor. As is evident from the equations above, the temporal ordering scores for a pair of relations are obtained by marginalizing over all the verb pairs used to express the two relations. We call this method Pairwise.

3.4.2 GraphOrder: Collective Temporal Ordering of Relations

The temporal ordering method described in previous section makes pairwise decision considering two relations at a time. However, it is conceivable that instead of ordering two relations at a time, a collective framework which looks at all relations at once could be useful. Transitivity is one such benefit that is possible in this collective ordering framework. For example, the pairwise method might infer that

[5]The SVO triples dataset is available from the authors on request.

TBefore(r_1, r_2) and TBefore(r_2, r_3) are true, but it might fail to identify that TBefore(r_1, r_3) is also true. However, if we considered all three relations at the same time, then it might be easier to establish the temporal relationship between r_1 and r_3. Although simple transitive closure would work well in case of hard orderings, it is not clear how such a closure could be computed with soft ordering constraints, as is the case in our setting, where there is inherent uncertainty in the inferred temporal orders. We note that the methods for inducing global ordering over soft orderings [5, 8] are not applicable in the current setting, as instead of a single global ordering, we are interested in finding the transitive closure over the soft orderings.

In order to overcome these challenges, we propose a novel graph-based label propagation algorithm GraphOrder, which is both parallelizable and scalable. We first create a graph $G = (X, E, W)$ whose vertices $X = R \cup U$ (see Equation 1), i.e., each vertex is either a relation or a verb. Edges in this graph consist of the following mix of directed and undirected edges[6]: (1) undirected edge between relation r and verb v with edge weight $W_{r,v} = \text{RelVerb}(r, v)$; (2) directed edge between verb v_i and v_j with $W_{v_i,v_j} = \text{NBefore}(v_i, v_j)$; (3) undirected edge between verb v_i and v_j with $W_{v_i,v_j} = \text{NSimultaneous}(v_i, v_j)$; (4) directed temporal order edge between relations r_i and r_j with $W_{r_i,r_j} = \text{TBefore}(\text{ri,rj})$; and (5) undirected temporal order edge between r_i and r_j with $W_{r_i,r_j} = \text{TSimultaneous}(\text{r}_\text{i}, \text{r}_\text{j})$. This graph is very similar to the one shown in Figure 1, except that there are no document nodes here. We have $E = E_D \cup E_U$ where E_D is the set of directed edges while E_U is the set of undirected edges.

We shall now inject each relation node $u \in R \subseteq X$ with a unique node specific label l_u. This seed information is stored in the initial label matrix Y with $Y_{u,l_u} = 1$. Each such relation specific label l_u has another temporally-after variant $\vec{l_u}$, which can be interpreted as follows: if another relation node $v \in R$ is assigned the label $\vec{l_u}$ with high score, then we should infer that there is a *before*(u, v) relationship between the two relations. Otherwise, a high l_u score on v should lead us to infer that there is a *simultaneous*(u, v). Following [19], we shall also have a none-of-the-above label, $\top$, which we shall use as an automatic threshold on the label scores. Let L be the total set of labels with total $|L| = 2 \times n + 1$ labels in it, where $|R| = n$ is the total number of relations.

In order to perform collective node classification and thereby infer temporal ordering among relations, our proposed algorithm, GraphOrder, optimizes the following objective function (Equation 6),

$$\begin{aligned}\min_{\{\hat{Y}_{u,l}\}} \quad & \mu_1 \sum_{u \in R} (Y_{u,l_u} - \hat{Y}_{u,l_u})^2 \\ & + \mu_2 \sum_{(u,v) \in E_D, l \in L} W_{u,v} \times (\hat{Y}_{u,l} - \hat{Y}_{v,\vec{l}})^2 \\ & + \mu_2 \sum_{(u,v) \in E_U, l \in L} W_{u,v} \times (\hat{Y}_{u,l} - \hat{Y}_{v,l})^2 \\ & + \mu_3 \sum_{x \in X, l \in L} (\hat{Y}_{u,l} - M_{u,l})^2 \end{aligned} \tag{6}$$

where μ_1, μ_2, μ_3 are hyperparameters, and M is a regularization matrix, and $\hat{Y}_{u,l} \in \mathbb{R}$ is the output variable which measure the score of label l assigned to node u. The objective above is convex[7] and is similar in spirit to the QC criteria [3] and MAD [19] that tries to minimize difference between predicted and actual labels, with the critical distinction that the second term in this objective involves directed edges with transformed labels across the edge (l vs $\vec{l}$). We develop a Jacobi iteration-based iterative algorithm to solve this objective[7]. However, in order to handle the directed edges, whenever a label l is to be spread over the directed edge $(u, v) \in E_D$, instead of updating the score for label l on v, we transform the label and update the score for $\vec{l}$ on v. We iterate this process for a fixed number of iterations[8], after which the assigned labels are used to infer the temporal relations as described earlier in this section. The objective function and solution to the optimization problem are detailed in the Appendix (Section A and Section B).

The algorithm we propose above is novel as no previous label propagation algorithm has considered the situation when there is a need to transform labels during propagation. Also, this algorithm is essentially self-supervised as no external hand labeled data is necessary. As we will see in the experimental results in Figure 4, the proposed algorithm effectively combines these strategies for the task of temporal order acquisition.

4. EXPERIMENTS

4.1 Experimental Setup

We select entities in Yago2 [12] as our domain instances. We select relations in Yago2 that are incident on these entities to temporally *order* and their relation instances (facts) to temporally *scope*. Yago2 is a knowledge base containing entities and relations automatically extracted from the entities' Wikipedia infoboxes. For this reason, we choose Wikipedia as the source of our unlabeled corpus D. Each document in D is thus about an entity (i.e., contains the entity in its infobox). Although Wikipedia pages, being general purpose, may seem more chronological and thus better suit the assumption that narrative follows temporal order, we show in experiments that the choice of Wikipedia pages does not in fact trivialize the problem. There is enough variation in the narrative sequence of verb mentions in Wikipedia that we benefit from aggregation. We use Wikipedia pages without any special processing, omitting infoboxes, using only sentences in the page sorted by their narrative order in the page.

The relations we want to temporally order and whose instances we want to time scope are chosen from the domains: *albums*, *cricketers*, *films*, *footballers*, *novels*, *operas*, *plays*, *politicians*, and *songs*. For each domain, we select the top 25 relations that have at least one argument belonging to the domain, to temporally order (based on the number of relation instances/facts). We also select as *domain instances*, whose Wikipedia pages we use as our corpus D, the top 1000 Yago2 entities that participate the most in these relations. By doing so, we ensure that D contains a lot of information on the relations we want to temporally order. We define an argument of a relation as belonging to the domain $films$ if it belongs to any Wikipedia category that ends with the word "$\_films$": e.g. *wikicategory_1930s_crime_films*, etc.

[6]By directed, we refer to edges corresponding to NBefore and TBefore; and by undirected to edges corresponding to NSimultaneous and TSimultaneous orders.

[7]Please see appendix for details.

[8]Set to 10 iterations for the experiments in this paper

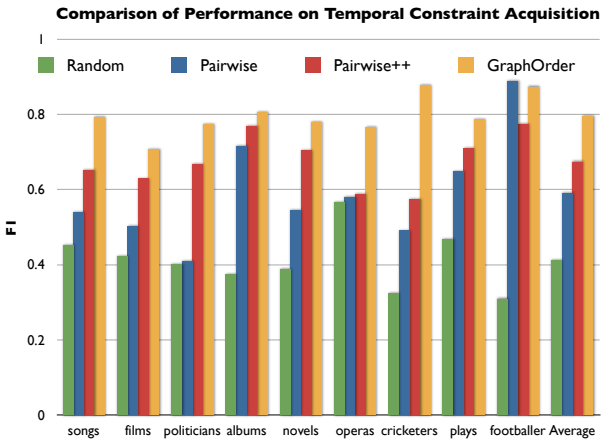


Figure 4: Comparison of performance of the temporal constraint acquisition methods discussed in the paper. We observe that the graph-based inference method GraphOrder (Section 3.4.2) (i.e., the rightmost bar) achieves the best performance on average.

For example, the top 5 relations selected for the domain *films* are *actedIn*, *wasCreatedOnDate*, *directed*, *created*, and *produced*. A lot of relations in Yago2 are biographical (birth, death) due to the nature of the infoboxes. We also notice that some entities are categorized incorrectly in Wikipedia, e.g., a *person* maybe categorized as $films$[9]. These may lead to irrelevant relations selected, e.g., $wasBornIn$ for $films$.

4.2 Temporal Ordering Experiments

Following [9], we use a **random** baseline which generated 300 random sets of pairwise orderings for each domain. In each set, we generate the same number of orderings as that learned by GraphOrder (described below), and the performance is averaged across these 300 sets. As there are no existing methods that output temporal orders at the relation level, we use the methods that are based on Pairwise Temporal Ordering (Section 3.4.1): using default verbs only (**Pairwise**) or using also verbs derived from the SVO dataset (**Pairwise++**) as additional baselines. We compare these baselines to our proposed method **GraphOrder** (Section 3.4.2), which performs collective temporal ordering using label propagation-based inference over a graph consisting of constraints from the Pairwise++ system. We observe that GraphOrder converges quite fast (10 iterations are usually sufficient).

For each domain and each method, we obtain temporal orderings of relations and the scores of the orderings. Each ordering is between a pair of relations. For evaluation, for each domain and each method we create a directed acyclic graph (DAG) of the learned orderings, traversing over the orderings from the highest score to the lowest score, and adding an ordering to the DAG only if it does not conflict with previously added orderings.

Similar to temporal evaluation in [9], we hand identify a *gold standard* set of temporal orderings for each domain. All attempts were made to include every *before* and *simultaneous* relation that can exist between the relations for the domain or that could be deduced through transitivity rules. We ignore temporal orders of relations that are ambiguous to humans. For example, $isMarriedTo$ and $actedIn$, one can be before or after the other at the relation-level. We evaluate each temporal ordering (i.e., each edge) in the DAG for correctness according to the gold standard. Precision is computed as the number of edges marked correct over the total number of edges marked unambiguous in the DAG. Since it is difficult to determine upfront the full set of valid temporal orderings, we decide to estimate the recall with respect to the largest number of correct edges returned by any one of the compared methods. Using precision and the estimated recall, we compute F1 (Figure 4).

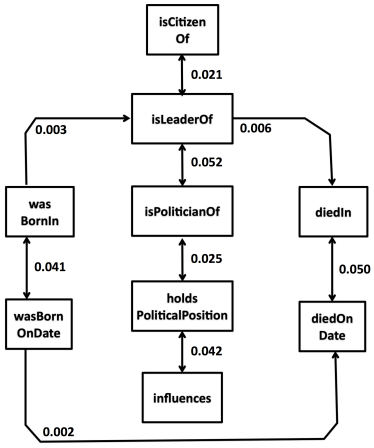
Experimental results comparing these four systems are shown in Figure 4. We see that GraphOrder performs better than all three baselines on average and across all domains, showing the merit of our graph-based, unsupervised, macro learning of temporal orderings. On average, we observe a 38.4% improvement in F1 score over the random baseline for GraphOrder.

In Figure 4, we see that performance (F1) of Pairwise++ is higher than Pairwise on average and across all domains except in $footballers$. This indicates that adding verbs from SVO typically improves temporal ordering and demonstrates the benefits of having a more flexible representation of relations that involves multiple verbs. We see that the highest increase in performance is in *politicians*, in which we find the largest number of SVO verbs (118 verbs) to represent the relations. In contrast, we only obtain one verb from SVO: "had visited" that represents the $influences$ relation in $footballers$. In this case, adding verbs from SVO does not help performance and may even degrade it if the verb found does not really express the relation, as in this case.

In Figure 4, we observe that performance of GraphOrder is higher than the non-transitive baselines (Pairwise++ and Pairwise). In terms of precision and estimated recall, we find that GraphOrder increases recall without hurting precision. On average, GraphOrder increases the number of correctly learned temporal orderings by 48.7% from that learned by Pairwise++ and this improves the F1 score by 12.2%. This shows how collective reasoning using label propagation can improve temporal ordering.

A few examples of temporal orders learned by GraphOrder in *politics* and in *films* are shown in Figure 5. As can be seen from the figure, in case of *politics*, the method is able to learn orderings pretty accurately: all relations happen after *wasBornIn*. In addition, a person must be a *citizen* of some country if he happens to *hold a political position*, and/or is *a leader* of something. We also notice that a *politician* or a *leader* must have an *influence* at least before he *dies*. In case of *films*, the method learns correctly that the process of *creating* a movie: *producing*, *directing*, *acting* must happen in the same period of time and all before the movie can *win* a prize. However, there are still questionable ordering such as $diedIn$ before $hasWonPrize$ (possibly winning an award posthumously), and mistake such as $wasBornIn$ simultaneous *produced*. We note that all these orderings have scores attached to them which can be taken into account in any subsequent consumption of these constraints, as we do in temporal scoping (Section 4.4).

[9]http://en.wikipedia.org/wiki/Joseph_Kessel is a person that has a Wikipedia category: "Novels adapted into films"



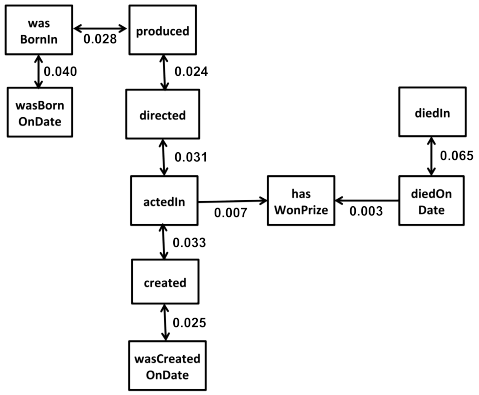


Figure 5: Examples of temporal orderings of relations in *politics* (left) and *films* (right) which were automatically learned by GraphOrder. Directed arrow indicates a *before* relation while each bi-directed arrow indicates a *simultaneous* relation. The score on each edge indicates the confidence of the temporal order.

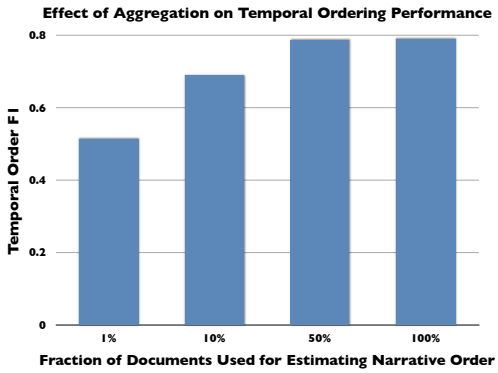


Figure 6: Performance of GraphOrder as the number of Wikipedia documents from which the narrative order of verbs are estimated is increased. We observe that the overall temporal ordering performance improves as increasing number of documents are used to estimate narrative order. All results are averaged over the 9 domains.

Domain	Constraint Source	Precision	Recall	F1
Films	None	55.6 (5.7)	49.9 (5.5)	52.6 (5.6)
	GraphOrder	71.0 (3.8)	62.3 (4.1)	66.4 (4.0)
	Manual	89.1 (5.6)	70.6 (3.3)	**78.8** (4.0)
Novels	None	53.4 (7.5)	37.5 (9.2)	43.8 (8.5)
	GraphOrder	67.2 (12.1)	44.4 (13.1)	**53.0** (12.7)
	Manual	60.8 (7.1)	41.4 (12.3)	48.8 (11.0)
Songs	None	31.0 (4.9)	16.5 (2.6)	21.5 (3.4)
	GraphOrder	60.0 (5.4)	26.9 (1.6)	**37.1** (2.4)
	Manual	43.5 (4.3)	20.8 (2.4)	28.1 (2.9)

Table 1: Results of temporal scoping experiment using the $\text{CoTS}_{\text{Soft}}$ framework, with different sources of inter-relation temporal ordering constraints: None (i.e., no such constraints used), constraints inferred by GraphOrder (Section 3.4.2), and manually encoded constraints. All results are averaged over five evaluation sets, with standard deviation indicated in brackets. For each domain, the setting with highest F1 score is marked in bold. See Section 4.4 for details.

4.3 Effect of Aggregation on Temporal Ordering

The goal in this section is to measure what effect aggregation of narrative order of verbs from a large number of Wikipedia documents has on final temporal ordering of relations. In Figure 6, we report the temporal ordering performance of GraphOrder, the best performing system from Section 4.2, as the fraction of Wikipedia documents from which narrative order of verbs are estimated is increased. All results are averaged across 9 domains. From Figure 6, we observe that performance improves sharply as the fraction of documents used is increased. This demonstrates the value of aggregation for temporal ordering. The results in Figure 6 suggest that a small number of documents (e.g., 1%) are not sufficient for optimal temporal ordering performance. This also suggests that the use of Wikipedia doesn't trivialize the problem as otherwise a small number of documents would have been sufficient for reliable estimation of narrative orders and thereby temporal order of relations. Instead, these results suggest that document specific inconsistencies can be overcome through aggregation across a large set of documents, resulting in improved temporal ordering performance.

4.4 Temporal Scoping Experiment

The experiments described above establish that GraphOrder, our proposed approach, can acquire knowledge of the *typical* temporal ordering between different relations. In this section we explore how useful these learned temporal constraints are, for the task of assigning specific temporal scope to individual facts (instances of these relations). In particular, we use an extended version of the CoTS system [20] to temporally scope facts from the Yago KB. CoTS is a joint inference ILP framework for temporal scoping that combines the rising and falling of counts of occurrences of facts in documents over time (as a signal to indicate the period of validity of the facts), with constraints about the facts (*si-*

multaneous, before or after temporal orders) to infer the temporal scope of facts (i.e., the interval of time when a fact is true). The original CoTS system [20] can only handle hard and unweighted constraints, which is not adequate in our case as the temporal ordering constraints learned by GraphOrder are soft and weighted in nature. In order to overcome these limitations, we present an extension of the CoTS system, which we shall refer to as $\text{CoTS}_{\text{Soft}}$.

$\text{CoTS}_{\text{Soft}}$ System: Let $x_{r,s,t} \in \{0,1\}$ be a binary variable with $x_{r,s,t} = 1$ whenever fact r from relation s is true at time t, and 0 otherwise. Let $z^b_{r,s,t} \in \{0,1\}$ be a binary variable with $z^b_{r,s,t} = 1$ indicating that fact s started to be true at time t. Similarly, $z^e_{r,s,t} = 1$ indicating that fact s ceased to be true at time t. Let $b(r,s,t)$ be our initial belief that fact s from relation r is true at time t. Now, given a temporal order TBefore(a, c) with score $q_{a,c}$, and two facts u and v from the relations a and c, respectively, $\text{CoTS}_{\text{Soft}}$ introduces the following constraint,

$$\sum_t t * z^e_{a,u,t} - \sum_t t * z^b_{c,v,t} - \xi_{a,u,c,v} \leq 0$$

where $\xi_{a,u,c,v} \in \{0,1\}$ is a binary *slack* variable which allows this constraint to be violated, but not without suffering a penalty of $q_{a,c}$ in the objective as we shall see shortly. Please note that $\text{CoTS}_{\text{Soft}}$ enforces this soft-constraint iff facts u and v have a shared argument entity. $\text{CoTS}_{\text{Soft}}$ also uses the CONSISTENCY constraints from [20] to make sure the z^b, z^e and x variables are consistent with one another at any given time. $\text{CoTS}_{\text{Soft}}$ uses the following set of inequalities to encode a TSimultaneous(a, c) ordering between the two facts u and v from above at any given time t.

$$\begin{aligned} x_{a,u,t} &\leq x_{c,v,t} + \xi_{a,u,c,v,t} \\ x_{c,v,t} &\leq x_{a,u,t} + \xi_{c,v,a,u,t} \end{aligned}$$

where $\xi_{a,u,c,v,t}$ and $\xi_{c,v,a,u,t}$ are binary slack variables. Let Δ be the set of all slack variables, with q_ξ representing the penalty associated with slack variable ξ. We now present the objective optimized by $\text{CoTS}_{\text{Soft}}$ subject to the constraints presented above.[10]

$$\max_{\{x_{r,s,t}\}} \sum_{r,s,t} b(r,s,t) * x_{r,s,t} - \gamma \times \sum_{\xi \in \Delta} q_\xi \times \xi$$

where γ is a hyperparamater which we set to 1 for all the experiments in this section. We use the Mosek[11] optimization package to solve this Integer Linear Program (ILP).

We now describe the experimental setup used to evaluate the usefulness of temporal orderings learned by GraphOrder. In this section, we experiment with three domains: *films*, *novels* and *songs*. For each domain, we obtain facts from the Yago2 KB as in previous sections, a subset of which is already temporally scoped. We split this subset into two parts, with the first set consisting 80% of the facts. We keep the temporal scopes of the first set fixed as prior knowledge, and use the $\text{CoTS}_{\text{Soft}}$ system to predict temporal scopes for the other set. We then compare predicted temporal scope with truth to compute precision, recall and F1. We repeat this process five times for each domain, generating a separate split each time. All results are averaged over these five runs. For each split, we evaluate temporal scoping performance of the $\text{CoTS}_{\text{Soft}}$ system when injected with inter-relation temporal ordering constraints obtained from three sources: (1) None (i.e., when no temporal ordering constraints are used); (2) temporal orders learned by GraphOrder in Section 4.2; and (3) manually crafted ordering rules. For all experimental runs, we required $\text{CoTS}_{\text{Soft}}$ to make point predictions (i.e., unit length temporal scopes). For each fact $r(a,b)$ we query the Gigaword corpus[12] with "*a AND b*" or "*a AND Default(r)*" if b is a date, to estimate $b(r,s,t)$. We perform all evaluations at the year granularity, and since Gigaword's coverage is limited to the period 1994-2008, we only consider facts which are true during this period.

Experimental results comparing these three ordering generators are presented in Table 1. For all three domains, we observe that GraphOrder leads to improvement in performance (F1) compared to when no constraints are used. Interestingly, constraints learned by GraphOrder outperforms manually crafted constraints in two out of the three domains. This might be due to the incomprehensiveness of the manual rule writing process leading to omission of certain useful orders, which were included in the orderings learned by GraphOrder.

Next, we evaluate whether the temporal orders learned by GraphOrder, along with $\text{CoTS}_{\text{Soft}}$, can be used to increase Yago2's temporal coverage, i.e., temporally scope Yago2 facts which are currently not scoped. We experiment with facts from the *films* domain (total 1831 facts), and fix all facts that are already temporally scoped (such as instances of *produced*). We then use the $\text{CoTS}_{\text{Soft}}$ system to temporally scope facts from the *actedin* relation, which are currently not scoped. Using random sampling-based manual evaluation of 100 facts, we observe a temporal scoping precision of 69.8%. Thus, we are able to increase Yago2's temporal coverage at a modest precision, showing the merit of learning temporal constraints by GraphOrder and using them for temporal scoping. A few examples of correct scoping are: 2001 for *actedIn*(Jason Biggs, American Pie 2); 1996 for *actedIn*(John Neville, High School High). Examples of an incorrect scoping is: 1998 for *actedIn*(Judy Davis, Deconstructing Harry) (correct: 1997).

5. RELATED WORK

Schank and Abelson [18] first proposed hand-coded scripts to describe frequently recurring social situations such as a visit to a restaurant. Our work is similar in spirit to scripts. However, instead of hand coding scripts, we automatically identify verbs that can express a relation, find sequence of these verbs in a document, and aggregate over similar sequences in a large number of documents to infer temporal order of relations expressed by the verbs.

We also draw from Chambers and Jurafsky's [9] automatic induction of script-like structure called narrative events chain. We use a related notion of protagonist (domain) overlap to relate the set of relations we temporally order. But we differ in that we use narrative order to serve as probabilistic training signals for learning temporal order; and use unsupervised method of temporal ordering instead of a supervised classifier. Also, their focus is to output the narrative chain of *verbs* in a document while ours is to output the

[10]Please note that even though we discuss only two types of constraints here, slack variable-based extensions of all the constraints in the CoTS system have been implemented inside $\text{CoTS}_{\text{Soft}}$. We refer the reader to [20] for a complete listing of these constraints.

[11]http://www.mosek.com/

[12]English Gigaword: http://bit.ly/KJMTYr

typical temporal order of *relations* over a large number of documents.

Fujiki et al. [11] investigated script acquisition by arranging first paragraphs of news articles based on their dates of issue. In contrast, we do not use such date of issue times to order paragraphs in documents, thus our method is applicable more widely even to documents without creation time. We also temporally order relations instead of just verbs.

Timely YAGO [22] harvests temporal facts using regular expressions in Wikipedia infoboxes, but is not applicable to arbitrary text. PRAVDA [21] harvests temporal facts using a combination of textual patterns and graph-based re-ranking techniques to extract facts and their temporal scopes at the same time. In contrast, our goal is to infer *typical* temporal order of *relations*, and not temporally scope individual facts. However, our experiments demonstrate that knowledge of typical temporal order can be effective in improving temporal scoping performance.

There are several algorithms for classifying temporal relations between events [15, 8, 24]; many of them trained on the TimeBank [17] dataset. Unlike these methods, we propose a self-supervised method for temporal ordering that does not require labeled data. While these previous works focused on temporal ordering of events with one verb per event, we temporally order relations that are represented by a group of verbs. To the best of our knowledge, there is no other work on temporal ordering of relations. However, these previous works are complementary in that their output can be used as additional verb ordering signals to augment our narrative order statistics.

Finally, CoTS [20] is a recently proposed system for temporally scoping relational facts which so far used manually edited temporal order constraints. While manual ordering is appealing, unfortunately it is not very scalable as in practice hundreds of relations, whose numbers are ever increasing, may need to be ordered. As demonstrated in Section 4, our automatically acquired, scalable temporal orderings can be a replacement for the current manual rules in CoTS. We also extend CoTS framework to $\text{CoTS}_{\text{Soft}}$ which can handle weighted constraints through the use of slack variables.

Learning to model ranked data from partial [13] and pairwise preferences [14] has been recently studied. While the pairwise orderings in [14] are unweighted, the orderings in GraphOrder are weighted. Also, these previous studies considered only one type of ordering relationship between two items (viz., whether one item is preferred over the other), while GraphOrder uses two ordering relationships, TBefore and TSimultaneous, to compare two items (relations in our case). More importantly, in contrast to these previous approaches, the goal in GraphOrder is *not* to learn a ranking model, but to expand an initial set of weighted pairwise temporal orderings by taking transitivity into account.

6. CONCLUSION

In this paper, we address the problem of learning typical temporal orderings over relations (e.g., learning that *directed(person, film)* typically occurs before *wonPrize(film, award)*). This problem is important because this type of knowledge can play a key role in assigning temporal scope to specific instances of these relations (e.g., to determine when the relation instance *wonPrize(Gone With The Wind, Academy Award)* occurred). We present both a method for acquiring this type of general ordering knowledge from distant supervision, and a method for using this knowledge to infer temporal scope for specific instances of these relations. To acquire this knowledge, we introduce GraphOrder, a novel graph-based collective label propagation method. Our approach is based on the assumption that relation mentions in sentences are often expressed by verbs, the assumption that narrative order of verbs correlates with the temporal order of the relations they represent, and the fact that temporal orderings are transitive. Our approach mines a corpus of 890m dependency parsed sentences to discover verbs that express relations of interest, and collects statistics about mentions of these verbs over a large number of documents. We show the effectiveness of our proposed method for learning temporal orders of relations, compared to non-transitive and random baselines. We also show the utility of this learned knowledge about temporal ordering, for the problem of assigning temporal scope to specific instances of these relations. In future work, we would like to explore alternative approaches of estimating narrative order. We would also like to extend the proposed algorithms to learning orderings when some relation arguments are instantiated, and believe the knowledge our current system learns can serve as useful priors for acquiring this information.

Acknowledgments

We thank Justin Betteridge (CMU) for help with parsing the corpus. We are thankful to Mosek[13] for their generous academic licensing; and to the anonymous reviewers for their constructive comments. This research has been supported in part by DARPA (under contract number FA8750-09-C-0179), Google, and Fulbright. Any opinions, findings, conclusions and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsors.

7. REFERENCES

[1] O. Alonso, M. Gertz, and R. Baeza-Yates. On the value of temporal information in information retrieval. In *ACM SIGIR Forum*, volume 41, pages 35–41. ACM, 2007.

[2] N. Balasubramanian, S. Soderland, Mausam, and O. Etzioni. Rel-grams: A probabilistic model of relations in text. In *Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX 2012)*, 2012.

[3] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. *Semi-supervised learning*, pages 193–216, 2006.

[4] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[5] P. Bramsen, P. Deshpande, Y. Lee, and R. Barzilay. Inducing temporal graphs. In *Proceedings of EMNLP*, 2006.

[6] S. Brody. Clustering clauses for high-level relation detection: An information-theoretic approach. In *ACL*, 2007.

[7] J. Callan, M. Hoy, C. Yoo, and L. Zhao. Clueweb09 data set. *boston.lti.cs.cmu.edu*, 2009.

[8] N. Chambers and D. Jurafsky. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP*, 2008.

[9] N. Chambers and D. Jurafsky. Unsupervised learning of narrative event chains. *ACL*, 2008.

[13] Mosek ApS: http://mosek.com/

[10] T. Chklovski and P. Pantel. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP*, volume 4, pages 33–40, 2004.
[11] T. Fujiki, H. Nanba, and M. Okumura. Automatic acquisition of script knowledge from a text collection. In *Proceedings of EACL*, 2003.
[12] J. Hoffart, F. Suchanek, K. Berberich, E. Lewis-Kelham, G. De Melo, and G. Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of WWW*. ACM, 2011.
[13] G. Lebanon and Y. Mao. Non-parametric modeling of partially ranked data. *Journal of Machine Learning Research*, 9:2401–2429, 2008.
[14] T. Lu and C. Boutilier. Learning mallows models with pairwise preferences. *ICML*, 2011.
[15] I. Mani, M. Verhagen, B. Wellner, C. Lee, and J. Pustejovsky. Machine learning of temporal relations. In *Proceedings of ACL*, 2006.
[16] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007.
[17] J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, et al. The timebank corpus. In *Corpus Linguistics*, volume 2003, page 40, 2003.
[18] R. Schank, R. Abelson, et al. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*, volume 2. Lawrence Erlbaum Associates Hillsdale, NJ, 1977.
[19] P. P. Talukdar and K. Crammer. New regularized algorithms for transductive learning. *ECML*, 2009.
[20] P. P. Talukdar, D. Wijaya, and T. Mitchell. Coupled temporal scoping of relational facts. In *Proceedings of WSDM*, pages 73–82. ACM, 2012.
[21] Y. Wang, B. Yang, L. Qu, M. Spaniol, and G. Weikum. Harvesting facts from textual web sources by constrained label propagation. In *Proceedings of CIKM*, 2011.
[22] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In *Proceedings of EDBT*, 2010.
[23] G. Weikum, S. Bedathur, and R. Schenkel. Temporal knowledge for timely intelligence. *Enabling Real-Time Business Intelligence*, pages 1–6, 2011.
[24] K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto. Jointly identifying temporal relations with markov logic. In *Proceedings of ACL*, 2009.

APPENDIX

A. GraphOrder OBJECTIVE IS CONVEX

We first reproduce the objective function minimized by the label propagation-based GraphOrder algorithm presented in Sec 3.2.2 (Equation 6), and write it as a sum of two functions as follows.

$$J(\{\hat{Y}_{ul}\}) = J_1(\{\hat{Y}_{ul}\}) + \mu_2 \times J_2(\{\hat{Y}_{ul}\})$$

where

$$\begin{aligned} J_1(\{\hat{Y}_{ul}\}) &= \mu_1 \sum_{u \in R} (Y_{u,l_u} - \hat{Y}_{u,l_u})^2 \\ &+ \mu_2 \sum_{(u,v) \in E_U, l \in L} W_{u,v} \times (\hat{Y}_{u,l} - \hat{Y}_{v,l})^2 \\ &+ \mu_3 \sum_{x \in X, l \in L} (\hat{Y}_{u,l} - M_{u,l})^2 \end{aligned}$$

where $\mu_1, \mu_2, \mu_3 \geq 0$ are hyperparameters and

$$J_2(\{\hat{Y}_{ul}\}) = \sum_{(u,v) \in E_D, l \in L} W_{u,v} \times (\hat{Y}_{u,l} - \hat{Y}_{v,\vec{l}})^2$$

From the construction of the graph, we have $W_{u,v} \geq 0$, i.e., all edge weights are non-negative. From the analysis in [19], we know that J_1 is a convex function. We also know that the non-negative weighted sum of two convex functions is also a convex function [4]. Putting all of these together, we shall be able to prove that J is convex if we can show that J_2 is a convex function.

We rewrite J_2 as the non-negative weighted sum of even smaller functions:

$$J_2(\{\hat{Y}_{ul}\}) = \sum_{(u,v) \in E_D, l \in L} W_{u,v} \times J_2^{'}(\hat{Y}_{ul}, \hat{Y}_{v\vec{l}})$$

where $J_2^{'}(\hat{Y}_{ul}, \hat{Y}_{v\vec{l}}) = (\hat{Y}_{u,l} - \hat{Y}_{v,\vec{l}})^2$.

Now, reusing the non-negative weighted sum property of convex functions once again, we can show that J_2 is convex if each $J_2^{'}(\hat{Y}_{ul}, \hat{Y}_{v\vec{l}})$ is instead convex (please note that $W_{u,v} \geq 0,\ \forall u, v$). We shall prove that $J_2^{'}$ is indeed a convex function by showing that its Hessian is a Positive Semi-Definite (PSD) matrix.

After some algebra, we can show that the Hessian matrix of $J_2^{'}(\hat{Y}_{ul}, \hat{Y}_{v\vec{l}})$ is $H_{2\times 2} = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$ and that it is PSD as for any $x = [x_1 x_2]^T$, we have $x^T H x = 2(x_1 - x_2)^2 \geq 0$.

From this we conclude that $J_2^{'}(\hat{Y}_{ul}, \hat{Y}_{v\vec{l}})$ is convex, which implies that J_2 is convex, and which in turn implies that J, i.e., the objective function function optimized by the GraphOrder is indeed convex.

B. SOLVING GraphOrder OBJECTIVE

In the previous section, we proved that the objective J minimized by GraphOrder is convex. Hence, the optimal solution is obtained by setting $\frac{\delta J}{\delta \hat{Y}_{u,l}} = 0,\ \forall u \in X, l \in L$. Following the analysis in [3], we develop a Jacobi iteration method and end up with the following iterative update, which updates the score for label l at node u at time instant $t+1$ as follows:

$$\hat{Y}_{u,l}^{(t+1)} = \frac{B_{u,l}^{(t)}}{B_{u,u}^{(t)}}$$

$$\begin{aligned} B_{u,l}^{(t)} &= \mu_1 \times S_{uu} \times I(l_u, l) \times Y_{u,l} \ + \\ &\quad \mu_2 \times \sum_{v:(u,v) \in E_D} W_{u,v} \times \hat{Y}_{v,\vec{l}}^{(t)} \ + \\ &\quad \mu_2 \times \sum_{v:(u,v) \in E_U} W_{u,v} \times \hat{Y}_{v,l}^{(t)} \ + \ \mu_3 \times M_{u,l} \end{aligned}$$

$$\begin{aligned} B_{u,u}^{(t)} &= \mu_1 \times S_{uu} \times I(l_u, l) \ + \ \mu_2 \times \\ &\quad \sum_{v:(u,v) \in E_D} W_{u,v} + \mu_2 \times \sum_{v:(u,v) \in E_U} W_{u,v} + \mu_3 \end{aligned}$$

Above, S is a diagonal matrix with $S_{u,u} = p_u^{inj}$ when $u \in R$ (the set of relation nodes) and 0 otherwise, with p_u^{inj} set using the equations in Sec 2.2 of [19]; $I(l_u, l) = 1$ iff $u \in R$ and l is the node specific label for relation node u. While computing $\hat{Y}_{u,\vec{l}}^{(t+1)}$, we consider $l = \vec{l}$. This iterative process is updated until convergence or until a fixed number of iterations. We have found that the iterations usually converge with a small number of iterations. We also note that the iterative updates above can be easily parallelized in the MapReduce framework using Hadoop, and hence the optimization can be applied to large graphs.