Harvesting Facts from Textual Web Sources by Constrained Label Propagation

Yafang Wang, Bin Yang, Lizhen Qu, Marc Spaniol and Gerhard Weikum
Max-Planck-Institut für Informatik
Saarbrücken, Germany
{ywang|byang|lqu|mspaniol|weikum}@mpi-inf.mpg.de

ABSTRACT

There have been major advances on automatically constructing large knowledge bases by extracting relational facts from Web and text sources. However, the world is dynamic: periodic events like sports competitions need to be interpreted with their respective timepoints, and facts such as coaching a sports team, holding political or business positions, and even marriages do not hold forever and should be augmented by their respective timespans. This paper addresses the problem of automatically harvesting temporal facts with such extended time-awareness. We employ pattern-based gathering techniques for fact candidates and construct a weighted pattern-candidate graph. Our key contribution is a system called PRAVDA based on a new kind of label propagation algorithm with a judiciously designed loss function, which iteratively processes the graph to label good temporal facts for a given set of target relations. Our experiments with online news and Wikipedia articles demonstrate the accuracy of this method.

Categories and Subject Descriptors

H.1.0 [**Information Systems**]: Models and Principles—*General*

General Terms

Algorithms, Design, Experimentation

Keywords

Knowledge Harvesting, Label Propagation, Temporal Facts

1. INTRODUCTION

1.1 Motivation

Who were the teammates of the legendary Argentinian player Diego Maradona? How often has the pop singer Madonna been married, and to whom? Questions like these often arise from the information needs of Internet users, but are not supported by keyword-based search engines. Instead, answering them requires an understanding of entities and relationships embedded in Web contents, or even better, a knowledge base that contains facts about people, organizations, locations, etc. In the last few years, such knowledge bases have indeed been built and made publicly accessible, for example: *DBpedia* [1], *YAGO* [17], *TextRunner* [9], *ReadTheWeb* [6], the commercial knowledge portals *freebase.com* and *trueknowledge.com*, the computational knowledge engine *wolframalpha.com*, the entity search engine *entitycube.research.microsoft.com*, and others. Most of these have been automatically compiled, either by integrating structured sources (e.g., geospecies.org, geonames.org, world factbook, etc.), by harvesting semi-structured sources such as Wikipedia info boxes, lists, and categories, or by information extraction from Web pages. Some of them involve a substantial amount of manual curation for quality assurance. Moreover, there is a strong momentum towards semantically inter-linking many knowledge sources into the *Linked Data* cloud [10], see *linkeddata.org*.

However, the world is highly dynamic and nothing lasts forever! Knowledge evolves over time, and many facts are fairly ephemeral, e.g., winners of sports competitions, and occasionally even CEOs and spouses. In addition, many information needs by advanced users require *temporal knowledge*. For example, consider the following variations of the above example questions: Who were the team mates of Diego Maradona during the 1990 FIFA World Cup? When did Madonna get married, when did she get divorced? None of these questions are supported by existing knowledge bases. The problem that we tackle in this paper is to automatically distil, from news articles and biography-style texts such as Wikipedia, *temporal facts* for a given set of relations. By this we mean instances of the relations with additional time annotations that denote the validity point or span of a relational fact. For example, for the *winsAward* relation between people and awards, we want to augment facts with the time points of the respective events; and for the *worksForClub* relation between athletes and sports clubs, we would add the timespan during which the fact holds. This can be seen as a specific task of extracting ternary relations, which is much harder than the usual information extraction issues considered in prior work.

1.2 Contribution

Our system called PRAVDA (label PRopagated fAct extraction on Very large DAta) gathers fact candidates and distills facts with their temporal extent based on a new form of *label propagation (LP)*. This is a family of graph-based semi-supervised learning methods, applied to (in our setting) a similarity graph of fact candidates and textual patterns. LP algorithms start with a small number of manually labelled seeds, correct facts in our case, and spread labels to neighbors based on a *graph regularized objective function* which we aim to minimize. The outcome is an assignment of labels to nodes which can be interpreted as a per-node probability distribution over labels. In our scenario, the labels denote relations to which the fact in a correspondingly labelled node belongs.

We adopt the specific algorithm of [18], coined MAD (Modified Adsorption), with an objective function that combines the quadratic loss between initial labels (from seeds) and estimated labels of vertices with a data-induced graph regularizer and an L2 regularizer.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.
Copyright 2011 ACM 978-1-4503-0717-8/11/10 ...$10.00.

The graph regularizer is also known as the un-normalized graph Laplacian, which penalizes changes of labels between vertices that are close. We develop substantial extensions, and show how to judiciously construct a suitable graph structure and objective function. Notably, we consider inclusion constraints between different relation labels for the same node in the graph. For example, we may exploit that a relation like *joinsClub* (with time points) is a sub-relation of *worksForClub* (with time spans).

Research on temporal fact extraction is still in its infancy: prior work is scarce and has major limitations. [23] showed how to extract temporal facts from Wikipedia infoboxes and lists, but can handle only semi-structured input and does not carry over to textual sources. [27] and [11] used an elaborated learning machinery to tap into temporal statements in text, but are computationally expensive and reliant on extensive training data. In contrast, our method is much more efficient and requires only minimal supervision. NLP research (e.g., [21, 5]) has also looked into temporal phrases, but has focused on detecting events (e.g., "her birthday"), relative expressions (e.g., "last Sunday"), and time-order relations (before, after, during, etc.). These basic tasks are still far from actual fact extraction. LP algorithms have been applied to knowledge-harvesting problems, but only to the simpler task of acquiring taxonomic relations between entities and classes (instanceOf) and among classes (subclassOf), see most notably the work of [19]. For harvesting arbitrary types of binary relations and especially for ternary relations with time points or time spans, we need a very different graph construction and loss function, and we have extended the MAD algorithm to consider also inclusion constraints (i.e., subsumptions) among relations.

In summary, this paper makes the following contributions:

- a new model for distilling temporal facts from text sources like news or biographies,
- an algorithm for extended label propagation with consideration of inclusion constraints,
- experimental studies, comparing the extraction of base facts (without time) against a state-of-the-art baseline [13], and demonstrating the effectiveness of harvesting temporal facts for a variety of relations from the domains of soccer and celebrities.

2. FRAMEWORK & SYSTEM OVERVIEW

This section gives a brief overview on the relation types we harvest and the system architecture. In particular, we introduce the distinction between *base* (without temporality) and *temporal relations* (including temporality).

2.1 Relation Types

We distinguish between *base* and *temporal relations*. *Base relations* indicate the relationships between two entities without any temporal information. A base relation R_b is commonly described with a type signature, $sig(R_b)$ = (*TYPE*$_1$, *TYPE*$_2$), meaning that the relation R_b only holds its semantic meaning for two entities whose types are *TYPE*$_1$ and *TYPE*$_2$, respectively. A base relation contains a set of base facts. Each base fact is in the form of (e_i, e_j), where e_i and e_j indicate two entities whose types are consistent with the corresponding base relation's type signature. For example, *worksForClub* is a base relation with signature (*PERSON*, *CLUB*). The base fact `(Lionel_Messi, FC_Barcelona)` holds for the *worksForClub* relation, where `Lionel_Messi` is a *PERSON* entity and `FC_Barcelona` is a *CLUB* entity.

Temporal relations indicate the relationships between two entities at specific time points or during particular time spans. Similar to base relations, a temporal relation is always associated with a type signature indicating the pertinent entity types. A temporal relation contains a set of temporal facts in the form of (e_i, e_j)@t_k, where e_i and e_j are two entities with pertinent types, and t_k indicates a temporal mention (either a time point or a time span). For example, *joinsClubTemp* is a temporal relation with type signature (*PERSON*, *CLUB*). The temporal fact `(David_Beckham, Real_Madrid)@2005` holds for the *joinsClubTemp* relation, indicating that `David_Beckham` joined `Real_Madrid` in 2005.

2.2 System Architecture

Figure 1 gives an overview of our system for fact harvesting from Web sources. The system consists of four components: *candidate gathering*, *pattern analysis*, *graph construction*, and *label propagation*. The components are invoked in four phases.

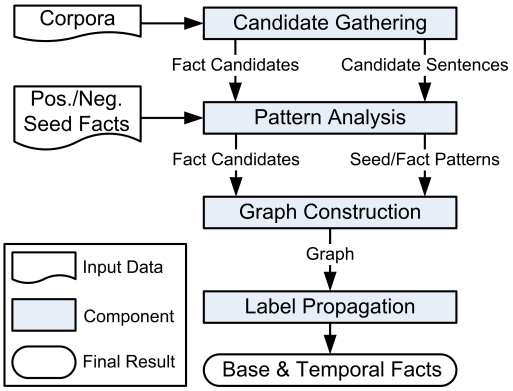

Figure 1: System Overview

1. *Candidate Gathering*: This phase serves to collect potentially relevant sentences. A sentence is interesting if it contains entities in relevant types for a target relation of interest. We employ a test for the potential presence of a base fact in a sentence, by checking for the existence of two noun phrases (denoting two entities) in the same sentence. In addition, we test for the existence of a temporal expression (currently only explicit date) in the sentence, thus producing raw input also for temporal fact candidates.
2. *Pattern Analysis*: Based on a (small, manually crafted) set of seed facts for a particular relation (either base or temporal), seed patterns in the form of sets of word-level n-grams are extracted from the interesting sentences. For each target relation that we aim to extract, a statistical prior is then computed based on how strongly fact candidates have evidence in the form of seed patterns. This will be discussed in Section 3.2.
3. *Graph Construction*: To facilitate the knowledge harvesting procedure, the fact candidates and pattern candidates are represented as vertices of a graph. Edges represent the evidence connection between a pattern and a fact candidate, as well as the similarity between two patterns. The graph construction is presented in Section 3.3.
4. *Label Propagation*: Finally, a graph-based semi-supervised learning method is employed to harvest both base facts and temporal facts. To this end, we have developed an extended label propagation algorithm (cf. Section 4 for details).

3. PATTERNS & GRAPH MODEL

In this section, we first describe our candidate gathering approach, raising surface (text) strings to entities. Then, we introduce our pattern analysis method, which is based on "basic", "seed" and

"fact" patterns. Basic patterns are (entity) type-aware word-level n-grams of surface strings. Based on statistical analytics, those patterns yielding in high quality results for a certain relation become seed patterns. Fact patterns are finally derived from sentences where the similarity of an observed basic pattern and seed patterns exceeds a pre-defined threshold. Finally, we will explain the underlying (pattern-candidate) graph model between patterns and candidate sentences. To facilitate the following discussions, we use bold capital letters to indicate matrices, and normal lower-case letters to indicate scalar values.

3.1 Candidate Gathering

Our approach is driven by target relations for which we would like to gather instances. The set of relations of interest is denoted as $\mathcal{R}$={R_1, R_2, ..., R_m}. Each is associated with a type signature and is either a base relation or a temporal relation. $Type(\mathcal{R})$ contains all possible entity types in the type signatures of the relations in $\mathcal{R}$.

We assume that there exists a knowledge base with typed entities which contains the individual entities whose types are in $Type(\mathcal{R})$. In this paper, we consider relations of interest from the domains of soccer and celebrities, and use YAGO [17] as knowledge base. The YAGO knowledge base contains almost all the pertinent entities harvested from Wikipedia, such as persons (both sports people and celebrities), clubs, locations, awards, universities, and even a large fraction of people's spouses. YAGO also provides us with a fairly complete dictionary of surface names for entities, including abbreviations (e.g., "ManU" for the entity `Manchester_United`). Thus, detecting mentions of entities in an input text is straightforward.

We consider a textual corpus $\mathcal{C}$, e.g., a set of Wikipedia articles, news articles, or Web pages, as input for candidate gathering. The corpus is pre-processed to generate meaningful segments. In this paper, we consider sentence-level segments: the corpus is decomposed into a set of sentences where each sentence is a sequence of tokens.

Given a sentence, an entity recognition and disambiguation module first checks whether at least two entities (and a temporal mention for temporal relations) are mentioned in the same sentence. This is primarily based on YAGO's "means" relation, where a string means an entity (e.g., "ManU" meaning `Manchester_United`). The entity recognition engine works in four stages:

1. Based on a sliding window technique "interesting" strings are extracted. A text string becomes interesting when it matches the subject of a YAGO fact with the means relation. That means, the string can be mapped to at least one YAGO entity.
2. Entities that can be mapped unambiguously (by plain string matching between the textual input and the means relation of YAGO) are identified. These entities serve as "ground-truth entities" for the following disambiguation steps.
3. The "ground-truth entities" and Wikipedia anchor links are used to build a graph for disambiguation. The graph consists of edges between those (entity-) nodes where mutual anchor links among pages exist. For each "interesting string" it is now checked if an entity exists that is directly linked with a single "ground-truth entity". If more than two links to "ground-truth entities" exist, the analyzed entity becomes - due to its strong relation to this context a "ground-truth" entity, too.
4. Disambiguation of each "interesting" string to a single (highest ranked) entity based on a distance function applied to the before constructed graph.

We then check whether the entity types are compatible with the type signature of some target relation R_m. If so, a candidate fact is generated, and the sentence is added to the set of interesting sentences. For example, the sentence "`David_Beckham` played for `Real_Madrid` and `LA_Galaxy`" will create the candidate facts (`David_Beckham`, `Real_Madrid`) and (`David_Beckham`, `LA_Galaxy`), but not (`Real_Madrid`, `LA_Galaxy`) as there is no relation of interest between two entities of type *CLUB*.

3.2 Pattern Analysis

3.2.1 Basic Patterns

Our notion of patterns builds upon the prior work of [13], where sets of word-level n-grams from sentences are used as patterns. We extend this approach by exploiting the type signatures of the target relations for which we perform fact harvesting.

Consider a fact candidate (e_i, e_j) and its corresponding candidate sentence, written in the form $x\ e_i\ y\ e_j\ z$ where x, y and z are surface string surrounding the entities. We consider the *context surface string* y as an (initial) indicator of the relationship with fact (e_i, e_j). For example, we observe a fact candidate (`David_Beckham`, `LA_Galaxy`) in the sentence s = "`David_Beckham` finally moved from Real_Madrid before his recent joining `LA_Galaxy` in 2007.", the context surface string "finally moved from Real_Madrid before his recent joining" is the evidence for the *joinsClubTemp* relationship between `David_Beckham` and `LA_Galaxy`.

Context surface strings are usually long and over specific. It is extremely rare to observe other entity pairs sharing exactly the same context surface strings. To overcome this problem, we generalize the pattern representation. One possible solution is to adopt word level n-grams to represent patterns. Before generating the n-grams, we first do compression and lifting on context surface strings.

In order to avoid getting misguided by problems induced from natural language processing, we convert each surface string into a *compressed surface string*. These contain only - preserving their original order - nouns, verbs, and prepositions after stop words removal based on Wikipedia's "List of English Stop Words"[1]. Nouns, verbs and prepositions are identified by LingPipe Part-of-Speech tagger[2]. All verbs are transformed into present tense based on the verb dictionary provided by Assert (Automatic Statistical SEmantic Role Tagger)[3]. Nouns are stemmed with PlingStemmer[4].

The compressed surface strings are further generalized by considering the set of types that constitute the type signatures of our target relations. This is done by replacing entities by their types (e.g. *PERSON*, *CITY*, *CLUB*, and *DATE*). Thus, the corresponding *lifted surface string* of sentence s is {"move from *CLUB* before join"}. For a lifted surface string, we generate word-level n-grams as its *basic pattern* (with n typically set to 2). For example, the basic pattern of s w.r.t. the entity pair (`David_Beckham`, `LA_Galaxy`) is denoted as $BP(s,$ `David_Beckham`, `LA_Galaxy`$)$ = ("move from", "from *CLUB*", "*CLUB* before", "before join").

3.2.2 Seed Patterns

Given a set of positive and negative samples for the facts in a specific relation, we are able to measure how good a pattern is for a particular relation. The best patterns are then regarded as *seed patterns* for this relation. The overall procedure of identifying seed patterns for a specific relation is similar to [13].

A positive seed fact (e_i, e_j) (or $(e_i, e_j)@t_k$) for a base relation R_b (or a temporal relation R_t), is a valid fact of the relation R_b (or R_t). A negative seed fact (e_i, e_j) (or $(e_i, e_j)@t_k$) for R_b (or R_t) is an entity pair (with temporal mention) that is not a fact of R_b (or

[1] *http://en.wikipedia.org/wiki/Stop_words*
[2] *http://alias-i.com/lingpipe/web/demo-pos.html*
[3] *http://cemantix.org/assert.html*
[4] *http://www.mpi-inf.mpg.de/yago-naga/javatools/*

R_t). $PF(R_i)$ and $NF(R_i)$ denote the manually crafted positive and negative seed facts for R_i.

For each base relation R_b we identify the sentences that contain two entities from a positive seed in $PF(R_i)$ and the sentences that contain two entities from a negative seed in $NF(R_i)$. These sentences are collected into the sets $PS(R_b)$ and $NS(R_b)$, respectively. Analogously, for each temporal relation R_t, the positive sentences are those that contain both entities and the temporal expression (date) of a positive seed, and the negative sentences are those that contain a negative seed. The sets of positive and negative sentences are denoted as $PS(R_t)$ and $NS(R_t)$, respectively.

Given a relation $R_i \in \mathcal{R}$, a candidate sentence s with lifted surface string $L(s)$ containing a basic pattern p, support and confidence are defined as follows:

$$supp(p, R_i) = |\{s \in PS(R_i) | p \subseteq L(s)\}|$$

$$conf(p, R_i) = \frac{|\{s \in PS(R_i) | p \subseteq L(s)\}|}{|\{s \in PS(R_i) \cup NS(R_i) | p \subseteq L(s)\}|}$$

The support value captures the frequency of a pattern in conjunction with positive seed facts, whereas the confidence value denotes the ratio of a pattern co-occurring with positive seed facts versus negative seed facts. If $supp(p, R_i)$ is above predefined thresholds, the basic pattern p is considered as a *seed pattern* of R_i. $SP(R_i)$ denotes the set of all seed patterns for R_i. In our experiment, we vary the threshold of support in different settings.

3.2.3 Fact Patterns

Given a candidate sentence $s = x\ e_i\ y\ e_j\ z$, the *fact pattern* of the sentence w.r.t. the entity pair (e_i, e_j) is defined as:

$$(TYPE_1, TYPE_2, BP(s, e_i, e_j)) \quad (1)$$

TYPE$_1$ and *TYPE*$_2$ are the types of e_i and e_j, respectively. $BP(s, e_i, e_j)$ is a basic pattern of sentence s w.r.t. the entity pair (e_i, e_j). In addition, a fact pattern is associated with a weight vector that contains the weights of the fact patterns with regard to the target relations of interest: $(w(BP(s, e_i, e_j), R_1), w(BP(s, e_i, e_j), R_2), \ldots, w(BP(s, e_i, e_j), R_m))$.

Given a relation R_k, if its type signature is the same as the fact pattern's type signature, the weight is defined by Equation (2):

$$weight(BP(s, e_i, e_j), R_k) = \max_{p \in SP(R_k)} (sim(BP(s, e_i, e_j), p) \times conf(p, R_k)) \quad (2)$$

where the similarity between the basic pattern $BP(s)$ and a pattern p in the seed patterns of R_i is defined based on the Jaccard coefficient with regard to n-grams q:

$$sim(BP(s, e_i, e_j), p) = \frac{|\{q \in BP(s, e_i, e_j) \cap q \in p\}|}{|\{q \in BP(s, e_i, e_j) \cup q \in p\}|}$$

3.3 Graph Model

A weighted undirected graph $G(V, E, \mathbf{W})$ is employed to facilitate the knowledge harvesting, where V is the set of vertices, E is the set of edges, and $\mathbf{W}$ is the weight matrix (i.e., an entry $\mathbf{W}_{ij}$ indicating the similarity of the corresponding vertices v_i and v_j).

We divide the set of vertices into two subsets: *fact candidate vertices* (V_F) and *fact pattern vertices* (V_P), where $V_F \cap V_P = \emptyset$ and $V = V_F \cup V_P$. A vertex in V_F corresponds to a fact candidate, i.e., either a base fact (e_i, e_j) or a temporal fact $(e_i, e_j)@t_k$. A vertex in V_P corresponds to a fact pattern in the form of $(TYPE_1, TYPE_2, p)$, as defined in formula (1), where p indicates a basic pattern.

Edges between a fact candidate vertex and a fact pattern vertex: An edge between a fact candidate vertex v_f and a fact pattern vertex v_p is generated, if there is a candidate sentence containing both. Intuitively, more candidate sentences provide higher support for the fact candidate v_f held in the relation v_p. Let $S(v_f, v_p)$ be the set of candidate sentences, then the weight of the edge (v_f, v_p) is defined and normalized as $1 - e^{(-1)*\beta*|S(v_f, v_p)|}$, where $\beta \in (0, 1]$. The parameter β smoothens similarity values by reducing the impact of outliers with undesirable high cardinality such as spam.

Edges between two fact pattern vertices: If two fact patterns are similar, their corresponding vertices are connected by an edge with a similarity-based weight. The intuition is that similar patterns often indicate the same relation. Two fact pattern vertices v_{p_i} and v_{p_j} are considered dissimilar (similarity is zero) if their corresponding type signatures are inconsistent (i.e., $v_{p_i}.TYPE_1 \neq v_{p_j}.TYPE_1$ or v.v.). If their type signatures are consistent, the edgeweight is defined by the similarity of the basic patterns $v_{p_i}.p$ and $v_{p_j}.p$ as follows:

1. If the two patterns share the same verb and preposition, the similarity between them is the distance-weighted Jaccard similarity. If the two patterns differ in their verbs or prepositions, then the similarity is set to zero.
2. If one pattern contains a verb and the other one does not, the similarity between them is zero.
3. If neither of them contains a verb, the similarity is the distance-weighted Jaccard similarity.

The distance-weight of an n-gram in a basic pattern is defined based on its position in the sentence. The closer the n-gram appears to the target entity, the higher the weight. This way, we can effectively deal with relatively long patterns. For instance in the example graph in Figure 2, although the two patterns V_{P2} and V_{P3} share the n-gram "move from", its weight in V_{P3} is very low, thus resulting in a very low similarity between the two patterns.

4. LABEL PROPAGATION ALGORITHM

After constructing the graph of fact candidates and fact patterns, a semi-supervised label propagation algorithm is applied to extract the relations that hold between fact candidates. The idea of label propagation is that the labeling of vertices (here the possible relations) is propagated to nearby vertices via weighted edges until convergence. In this process, the vertices linked by highly weighted edges are more likely to have the same labels, because the weights of edges represent the vertex similarities. All vertices are treated uniformly, regardless of whether they represent fact candidates or fact patterns.

Suppose the graph contains n vertices, consisting of fc fact candidate and fp fact pattern vertices, where $|V| = n$, $|V_F| = fc$, $|V_P| = fp$, and $n = fc + fp$. In addition, there are $m + 1$ labels. The first m labels correspond to the m relations in the set $\mathcal{R}$ of target relations, plus a "none" label (denoting no or an unknown relation). In the following, we will use labels and relations interchangeably.

The matrix $\mathbf{Y} \in \mathbb{R}_+^{n \times (m+1)}$ denotes the graph's initial label assignment. The vector $\mathbf{Y}_{i*}$ denotes the i^{th} row of matrix $\mathbf{Y}$, with each element $\mathbf{Y}_{ik} \in [0, 1]$ indicating vertex v_i's confidence score for label k, where $k \in \{1, 2, ..., m + 1\}$. The ℓ^{th} column of matrix $\mathbf{Y}$ is denoted by vector $\mathbf{Y}_{*\ell}$.

When a fact candidate vertex v_i corresponds to a positive seed fact of relation R_k, the vertex is labelled as R_k with confidence 1, i.e., $\mathbf{Y}_{ik} = 1$. A fact pattern vertex v_j is labelled as R_k with confidence 1 (namely $\mathbf{Y}_{jk} = 1$), if its weight for a seed pattern of relation R_k (evaluated by Equation 2) is greater than a pre-specified threshold value γ. Note that if v_j has a strong weight (greater than γ) with the seed patterns of more than one relation, all the corresponding relations are labelled. All other entries in $\mathbf{Y}$ are initially set to 0.

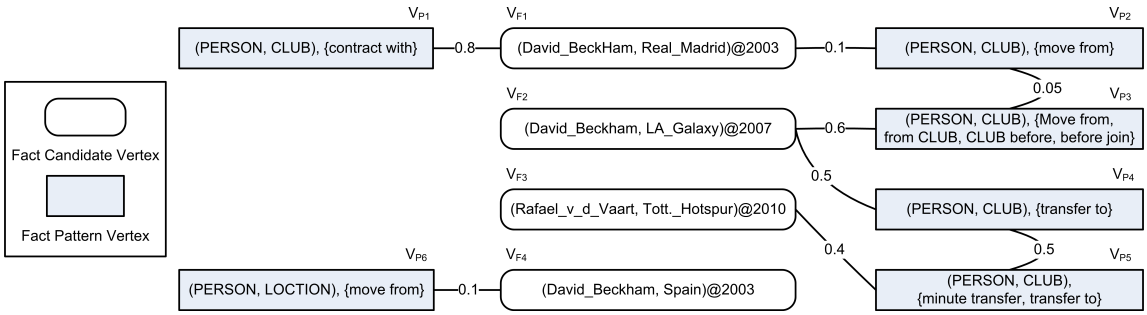

Figure 2: An Example Graph

When the label propagation process finishes, each vertex is associated with a vector indicating the estimated labels. We use matrix $\widehat{\mathbf{Y}} \in \mathbb{R}_+^{n\times(m+1)}$ to indicate the estimated labels for all vertices. Note that the matrix $\widehat{\mathbf{Y}}$ has exactly the same ordering of vertices and labels as $\mathbf{Y}$. Finally, for each vertex, the labels with sufficiently high confidence scores are selected.

4.1 Basic Objective Function

As shown in [18], the process of label propagation is realized by minimizing the following objective function:

$$\mathcal{L}(\widehat{\mathbf{Y}}) = \sum_{\ell} \Big[(\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell})^T \mathbf{S}^{\ell} (\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell}) + \mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell} + \mu_2 \| \widehat{\mathbf{Y}}_{*\ell} - \mathbf{R}_{*\ell} \|_2 \Big] \quad (3)$$

where $\mathbf{Y}_{*\ell}$ and $\widehat{\mathbf{Y}}_{*\ell}$ denote the ℓ^{th} column vector of the initial assignment matrix $\mathbf{Y}$ and estimated label matrix $\widehat{\mathbf{Y}}$, respectively.

The intuition of keeping estimated labels centred around initial labels is realized by the first term of the objective function (3), which is the quadratic loss that measures the discrepancies between the initial labels and the estimated labels. In the original MAD algorithm of [18], the diagonal matrix $\mathbf{S}^{\ell}$ is identical for every label, with p_i^{inj} on the main diagonal for labeled vertices and 0 for unlabeled ones, where p_i^{inj} is set to be proportional to vertex v_i's degree. Since the initially labelled fact pattern vertices may be noisy, the corresponding elements on the main diagonal of $\mathbf{S}^{\ell}$ should also be able to control the confidences on the label. Assuming that we are considering label ℓ and v_i is a fact pattern vertex, we represent the confidences on the label ℓ by multiplying p_i^{inj} with the vertex v_i's fact-pattern weight on label ℓ (evaluated by Equation 2).

The label spreading effect is achieved by the unnormalized graph Laplacian matrix $\mathbf{L}$, which smoothes label similarities between linked vertices based on edge weights. This property becomes clearer when $\mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell}$ is rewritten as

$$\mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell} = \mu_1 \frac{1}{2} \sum_{i,j} \mathbf{Z}_{ij} (\widehat{\mathbf{Y}}_{i\ell} - \widehat{\mathbf{Y}}_{j\ell})^2 \quad (4)$$

where $\mathbf{Z}_{ij} = \mathbf{W}_{ij} \times p_i^{cont} + \mathbf{W}_{ji} \times p_j^{cont}$ and $\mathbf{W}_{ij}$ is the edge weight between vertices v_i and v_j. Following [18], we set p_i^{cont} to be inversely proportional to vertex v_i's degree. The hyper-parameter μ_1 controls the influence of the graph Laplacian. In order to minimize the function, similarities of labels between vertex pairs linked with high edge-weight are enforced, while different labels are tolerated for those vertex pairs with low edge-weights.

The last term in the objective function (3) can be regarded as a combination of an L2 regularizer for the m relations of interest and a loss function for the "none" label. $\mathbf{R}_{*\ell}$ is the ℓth column of the abandon matrix $\mathbf{R} \in \mathbb{R}_+^{n\times(m+1)}$. Specifically, $\mathbf{R}_{*\ell}$ is a zero-valued column vector for every $\ell \in \{1, 2, ..., m\}$, which works exactly as an L2 regularizer for the estimated labels of the m relations of interest to avoid over-fitting to their seed labels. However, the vector of the last label $(m+1)$ is $\mathbf{R}_{*(m+1)} = (p_1^{abnd}, p_2^{abnd}, ... p_n^{abnd})^T$, where p_i^{abnd} is set to 1-p_i^{inj}-p_i^{cont} like in the original MAD algorithm [18]. The hyper-parameter μ_2 adjusts the degree of regularization and the trust in the "none" relation.

Since we only change the diagonal matrices $\mathbf{S}_{\ell}$, the objective function can still be solved by the optimization algorithm of MAD [18].

4.2 Incorporating Inclusion Constraints

Sometimes, a relation R_i implies another relation R_j, meaning that a valid fact of R_i should also be a valid fact of R_j. In such a case, there is an inclusion constraint (subsumption) that R_i implies R_j. For example, the relation *isCapitalOf* between cities and countries implies the *isCityOf* relation. Inclusion dependencies arise in specific forms for temporal relations, that is, between events and their implied lasting relations, and also between base relations and their temporal counterparts. For example, the event relation *joinsClubTemp* implies *worksForClubTemp* and the base relation *worksForClub*. The temporal fact *joinsClubTemp* (`David_Beckham`, `Real_Madrid`) @2003 also implies the temporal fact *worksForClubTemp* (`David_Beckham`, `Real_Madrid`) @2003, which in turn implies the base fact *worksForClub* (`David_Beckham`, `Real_Madrid`).

Inclusion constraints provide an additional asset: it can generate (or reinforce) the evidence of a fact by observing another fact of a different relation. Thus, if relation ℓ implies relation ℓ', it suggests for a fact candidate v_m having label ℓ, that the estimated confidence value $\widehat{\mathbf{Y}}_{m\ell'}$ should be greater or equal than the estimated confidence value $\widehat{\mathbf{Y}}_{m\ell}$. For instance, if a fact candidate v_m is estimated as *joinsClubTemp*, its estimated labels should satisfy the following constraint $\widehat{\mathbf{Y}}_{m(worksForClubTemp)} \geq \widehat{\mathbf{Y}}_{m(joinsClubTemp)}$. In the following, we will show how to incorporate inclusion constraints into an extended label propagation framework.

4.2.1 Objective Function with Inclusion Constraints

We first introduce some notations. The matrix $\mathbf{C} \in \mathbb{R}_+^{m\times m}$ records the inclusion dependencies between different relations of interest. Specifically, if relation ℓ implies relation ℓ', we set $\mathbf{C}_{\ell'\ell} = 1$. Given a label ℓ, $imply(\ell)$ denotes the labels that imply ℓ and $implied(\ell)$ those labels that are implied by ℓ.

We introduce a new objective function, described in equation (5), to incorporate inclusion constraints between different relations:

$$\mathcal{L}(\widehat{\mathbf{Y}}) = \sum_{\ell} \Big[(\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell})^T \mathbf{S}_{\ell} (\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell}) + \mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell}$$

$$+ \mu_2 \|\widehat{\mathbf{Y}}_{*\ell} - \mathbf{R}_{*\ell}\|_2 + \mu_3 \sum_v \sum_{k \neq \ell} \mathbf{C}_{\ell k} \mathbf{Y}_{vk} (\widehat{\mathbf{Y}}_{v\ell} - \widehat{\mathbf{Y}}_{vk})^2 \Big]$$

$$s.t.\ \widehat{\mathbf{Y}}_{v\ell'} \geq \widehat{\mathbf{Y}}_{v\ell},\ \ v \in V,\ \ell' \in implied(\ell) \tag{5}$$

The last term in Equation (5) serves to smoothen the estimated labels which satisfy the inclusion constraints. Note that only vertices containing initial labels are smoothed. In contrast to the MADDL algorithm [18], which assumes all correlations among classes to be symmetric, our approach does not assume symmetry of correlations. For example, assume that *joinsClubTemp* implies *worksForClubTemp* and the fact candidate `(David_Beckham, Real_Madrid)` @2003 holds for *joinsClubTemp* with high probability (e.g. 0.9), but has low probability for the *worksForClubTemp* relation (e.g. 0.1). This is undesirable and should be smoothed, as *joinsClubTemp* implies *worksForClubTemp*. Therefore, the objective function (5) will pull the estimated value for *worksForClubTemp* closer to the value for *joinsClubTemp* by reducing the labels' differences. On the contrary, if the same fact candidate holds for *worksForClubTemp* with high probability (e.g., 0.9), but has low probability for the *joinsClubTemp* (e.g., 0.1), the objective function (5) will not pull the estimated value for the *joinsClubTemp*.

4.2.2 Optimization Problem Solution

In the Appendix, we proove that adding inclusion constraints does not change the *convexity* of the objective function in Equation (3). The convex optimization problem is solved by *cyclic coordinate descent* [3]. The algorithm iterates through each coordinate of $\widehat{\mathbf{Y}}$, and solves sequentially the following sub-problems until it converges:

$$\min_{\widehat{\mathbf{Y}}_{v\ell}} \mathcal{L}(\widehat{\mathbf{Y}})\ \ s.t.\ \ lb \leq \widehat{\mathbf{Y}}_{v\ell} \leq ub \tag{6}$$

where lb and ub are the lower and upper bounds of $\widehat{\mathbf{Y}}_{v\ell}$, respectively.

In each sub-problem, all variables of $\widehat{\mathbf{Y}}_{v\ell}$ except the one at the current coordinate are fixed. So this technique optimizes the objective function by solving a sequence of single-variable optimization problems. Because the problem is a weighted sum of quadratic functions, there is a closed-form solution for each sub-problem. For coordinate $\widehat{\mathbf{Y}}_{v\ell}$, we set the first partial derivative with respect to $\widehat{\mathbf{Y}}_{v\ell}$ to zero. Thus, the optimum in the absence of constraints is:

$$\widehat{\mathbf{Y}}_{v\ell} = \frac{\mathbf{S}_{\ell vv}\mathbf{Y}_{v\ell} + \mu_1 \sum_i \mathbf{Z}_{vi}\widehat{\mathbf{Y}}_{i\ell} + \mu_2 \mathbf{R}_{v\ell} + \mu_3 \sum_{k\neq\ell} \theta \widehat{\mathbf{Y}}_{vk}}{\mathbf{S}_{\ell vv} + \mu_1 \sum_i \mathbf{Z}_{vi} + \mu_2 + \mu_3 \sum_{k\neq\ell} \theta}$$
$$\theta = (\mathbf{C}_{\ell k} + \mathbf{C}_{k\ell})\mathbf{Y}_{vk} \tag{7}$$

From the inclusion constraints we can infer a lower (lb) and an upper bound (ub) for the solution. Specifically, if $imply(\ell)$ is not empty, $lb = \max_{\ell' \in imply(\ell)} \widehat{\mathbf{Y}}_{v\ell'}$, otherwise $lb = 0$; if $implied(\ell)$ is not empty, $ub = \min_{\ell' \in implied(\ell)} \widehat{\mathbf{Y}}_{v\ell'}$, otherwise $ub = 1$. For example, the lower bound of *worksForClubTemp* is the maximum of *joinsClubTemp* and *leavesClubTemp*, with the upper bound 1. Similarly, the upper bound of *joinsClubTemp/leavesClubTemp* is the value on *worksForClubTemp*, with the lower bound 0. The final optimum is computed by $\max(\min(\widehat{\mathbf{Y}}_{v\ell}, ub), lb)$ to ensure that the optimal solution satisfies the inclusion constraints.

We refer to this algorithm as *ICMAD* (Inclusion-Constraints-aware MAD). The complete method is shown in Algorithm 1. Compared to the original MAD method, the lower and upper bounds are checked after each label update.

Algorithm 1 ICMAD Algorithm.

Input: Graph: $G = (V, E)$,
Matrices $\mathbf{S}^\ell$ for each $\ell \in [1...m+1]$,
Matrix $\mathbf{Z}$ derived from the graph Laplacian,
Abandon matrix $\mathbf{R}$,
Initial labels matrix $\mathbf{Y}$,
Inclusion constraint weight matrix $\mathbf{C}$
Output: Estimated labels matrix $\widehat{\mathbf{Y}}$

```
Ŷ ← Y;
repeat
  for each v ∈ V do
    for each ℓ ∈ [1, ...m + 1] do
      // Evaluation based on Equation (7);
      numerator ← S^ℓ_vv Y_vℓ + μ1 Σ_i Z_vi Ŷ_iℓ + μ2 R_vℓ +
                  μ3 Σ_{k≠ℓ} (C_ℓk + C_kℓ) Y_vk Ŷ_vk;
      denominator ← S^ℓ_vv + μ1 Σ_i Z_vi + μ2 +
                    μ3 Σ_{k≠ℓ} (C_ℓk + C_kℓ) Y_vk;
      Ŷ_vℓ ← numerator / denominator;
      for each k ∈ [1, ...m] ∧ k ≠ ℓ do
        // Lower and upper bounds checking;
        if C_ℓk > 0 ∧ Ŷ_vℓ < Y_vk then Ŷ_vℓ ← Y_vk;
        if C_kℓ > 0 ∧ Ŷ_vℓ > Y_vk then Ŷ_vℓ ← Y_vk;
      end for
    end for
  end for
until converged
```

4.3 Combining Base and Temporal Graphs

The above *ICMAD* algorithm can be applied to either base or temporal facts and their patterns. So far, however, these would be two separate instantiations of the same algorithm. In this subsection, we discuss how to combine the two settings for enhanced output.

For base relations, seed patterns may be noisy, especially when positive seeds (entity pairs) hold for multiple relations. For example, a person may be born and may die in the same place, so both relations *isBornIn* and *hasDiedIn* have similar seed patterns, which is undesired. However, the corresponding temporal relations are better distinguishable, as - usually - people do not die on the same date (or same year) they are born. Thus, by combining base and temporal graph, the seed patterns from the temporal graph can help to counter ("correct") noisy patterns in the base graph.

The pattern nodes in the temporal graph are actually a subset of those in the base graph. Thus, we can unify both graphs by sharing all pattern nodes and conceptually connecting the fact candidate nodes from base and temporal graph with their respective patterns. A new objective function is described in Equation (8), where matrix $\mathbf{G} \in \mathbb{R}_+^{p_b \times p_t}$ records the relationship of the same pattern nodes in base and temporal graph. p_b and p_t are the number of pattern nodes in base and temporal graph. If a pattern node v_i in the base graph is exactly the same as the pattern node v_j in the temporal graph, then we assign $\mathbf{G}_{ij}$ to 1. Otherwise, $\mathbf{G}_{ij}$ is set to 0.

Suppose $\mathbf{Y} = \left(\begin{smallmatrix} \mathbf{Y_b} & 0 \\ 0 & \mathbf{Y_t} \end{smallmatrix}\right)$, where $\mathbf{Y_b}$ and $\mathbf{Y_t}$ indicate the initial label matrices of base and temporal graph, respectively. Likewise, $\widehat{\mathbf{Y}} = \left(\begin{smallmatrix} \widehat{\mathbf{Y_b}} & 0 \\ 0 & \widehat{\mathbf{Y_t}} \end{smallmatrix}\right)$, where $\widehat{\mathbf{Y_b}}$ and $\widehat{\mathbf{Y_t}}$ indicate the estimated label matrices of base and temporal graph, respectively. $\mathbf{S}^\ell$ and $\mathbf{L}$ can also be re-written analogously. On this basis, we define Equation (8) as the new objective function as follows:

$$\mathcal{L}(\widehat{\mathbf{Y}}) = \sum_\ell \Big[(\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell})^T \mathbf{S}^\ell (\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell}) + \mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell}$$
$$+ \mu_2 \|\widehat{\mathbf{Y}}_{*\ell} - \mathbf{R}_{*\ell}\|_2 + \mu_3 \sum_v \sum_{k\neq\ell} \mathbf{C}_{\ell k} \mathbf{Y}_{vk} (\widehat{\mathbf{Y}}_{v\ell} - \widehat{\mathbf{Y}}_{vk})^2$$
$$+ \mu_4 \sum_{v,u} \mathbf{G}_{vu} (\widehat{\mathbf{Y_b}}_{v\ell} - \widehat{\mathbf{Y_t}}_{u\ell''})^2 \Big]$$
$$s.t.\ \widehat{\mathbf{Y}}_{v\ell'} \geq \widehat{\mathbf{Y}}_{v\ell},\ \ v \in V,\ \ell' \in implied(\ell) \tag{8}$$

Base Relations	Temporal Relations	Type Signatures
isBornIn	*isBornInTemp*	(*PERSON*, *CITY*)
hasDiedIn	*hasDiedInTemp*	(*PERSON*, *CITY*)
worksForClub	*worksForClubTemp*	(*PERSON*, *CLUB*)
	joinsClubTemp	(*PERSON*, *CLUB*)
	leavesClubTemp	(*PERSON*, *CLUB*)
isMarriedTo	*isMarriedToTemp*	(*PERSON*, *PERSON*)
	getsMarriedWithTemp	(*PERSON*, *PERSON*)
	getsDivorcedFromTemp	(*PERSON*, *PERSON*)
	winsAwardTemp	(*PERSON*, *AWARD*)

Table 1: Relations of Interest

where label ℓ'' is the corresponding temporal relation of base label ℓ (e.g., ℓ for *worksForClub* with ℓ'' for *worksForClubTemp*).

The new objective function shown in Equation (8) keeps the same form as Equation (5). Therefore, it is still solvable by cyclic coordinate descent.

5. EXPERIMENTS

All methods used in this paper are implemented in Java using Sun JDK 1.6. We then detect and disambiguate entities based on the Stanford named entity recognizer[5] and YAGO's "means" relation as oulined in Section 3.1. Temporal mentions are identified by regular expressions (currently yearly granularity). The actual label propagation algorithm is based on the Junto library [18].

Experiments have been conducted on a workstation with two Intel Xeon 2.40GHz, 4-Core CPUs, and 40GB memory. Except parameters $\beta = 0.05$ and $\gamma = 0.5$, all the other hyper parameters used in our experiments are manually tuned on a separate dataset.

5.1 Experimental Setup

Data Sets: Our methods have been evaluated against two corpora from the soccer and celebrity domain. In the soccer domain, we selected more than 23.000 soccer players' Wikipedia articles. In addition, we retrieved around 110.000 on-line news articles (e.g., BBC, Yahoo! news, ESPN, etc.) by searching for players contained in the "FIFA 100 list"[6]. Likewise, the celebrity corpus has been populated with more than 88.000 news articles of persons mentioned in the "Forbes 100 list"[7] in addition to their Wikipedia articles.

Relations of Interest: In the current experiments, we are interested in four base relations and nine temporal relations. The relations and their type signatures are summarized in Table 1. It is worth mentioning that temporal relations (highlighted in grey) such as *isBornInTemp* or *hasDiedInTemp* are "extensions" of their corresponding base relations *isBornIn* resp. *hasDiedIn* by temporal information for the identified event (e.g. birth, death, etc.).

Seed Selection: For each relation positive and negative seed facts have been manually selected. Facts with prominent entities are chosen as seed facts due to their frequency. Thus, k co-occurring entities are chosen as seed facts for each relation type signature.

For example, when selecting seed facts for the *worksForClub* relation, we compute the co-occurrence statistics (eventually supported by keyword filtering) of *PERSON* and *CLUB* entities based on the same sentence. Starting with the highest ranked entity pair, we manually check in Wikipedia if the relation of interest is satisfied. If approved, it becomes a positive seed fact, otherwise a negative.

Performance Metrics: Baseline for our experiments is the state-of-the-art PROSPERA system [13]. We evaluate both approaches in terms of precision correctness of the extracted facts. Due to the size of the corpus, evaluation of recall is impossible. Precision estimates are based on sampling (50 facts per relation) with a human evaluation against ground truth information in Wikipedia.

Sample Output: For the relations of interest we generate patterns and (temporal) facts. Table 2 depicts randomly chosen results from both corpora. Results for temporal facts are marked in grey.

Relation	Pattern	Fact
joinsClubTemp	join *LEAGUE* side	(David_Beckham, Los_Angeles_Galaxy)@2007
leavesClubTemp	transfer from	(Thierry_Henry, Juventus_F.C.)@1999
getsMarriedWithTemp	marry	(Demi_Moore, Ashton_Kutcher)@2005
getsDivorcedFromTemp	divorce	(Jennifer_Aniston, Brad_Pitt)@2005
worksForClub	to join at	(Michael_Owen, Real_Madrid_C.F.)
isMarriedTo	husband	(Hillary_Rodham_Clinton, Bill_Clinton)

Table 2: Patterns and Facts extracted for Relations of Interest

5.2 Results on Base Fact Extraction

We first compare base fact extraction with PROSPERA and PRAVDA. The support threshold in pattern analysis is $supp(p, R_i) = 10$ with 100 positive and 10 negative seed facts for each relation.

	PRAVDA		PROSPERA	
Relation	**# Facts**	**Precision**	**# Facts**	**Precision**
worksForClub	12.724	88%	4.536	82%
isBornIn	3.629	88%	2.426	90%
hasDiedIn	183	90%	55	93%
isMarriedTo	428	74%	147	90%

Table 3: Base Fact Extraction (100 positive, 10 negative seeds)

Table 3 summarizes the results of this experiment. As one can see both approaches achieve comparable precision. However, PRAVDA clearly outperforms PROSPERA with respect to the number of extracted facts. The reason for this major gain can be explained with LP's propagation of the labels' confidence scores from one node to another. This results in confidence propagation among very similar, but less frequently occuring patterns ("be born in" and "born in"). Of course, this comes with the risk of inducing more noise, which becomes apparent for more complex relations such as *isMarriedTo*. The explanation for this "misbehavior" is simple: a prominent couple will be frequently mentioned in the media in completely different contexts (e.g. when dating, talking, visiting, etc.), thus, propagating high confidence to less useful labels, too.

In order to quantify the impact of the number of seed facts, we now evaluate the performance of two base relations (*worksForClub* and *isMarriedTo*) for varying numbers of positive seed facts while keeping the number of negative seeds fixed with 10. We vary them from 10, 20, 50 to 100 by simultaneously adopting the support $supp(p, R_i)$ used in pattern analysis to 4, 6, 8 and 10 respectively.

[5] *http://www-nlp.stanford.edu/software/CRF-NER.shtml*
[6] *http://en.wikipedia.org/wiki/FIFA_100*
[7] *http://www.forbes.com/lists/2010/53/celeb-100-10_The-Celebrity-100.html*

		PRAVDA		PROSPERA	
	# Pos. Seeds	# Facts	Precision	# Fact	Precision
worksForClub	10	4.882	84%	2.202	86%
	20	8.381	82%	3.358	86%
	50	10.826	82%	4.321	84%
	100	12.724	88%	4.536	82%
isMarriedTo	10	388	74%	16	0%
	20	391	64%	123	88%
	50	391	78%	159	82%
	100	428	74%	147	90%

Table 4: Impact of Varying the Number of Positive Seed Facts

Table 4 summarizes the results on the impact of varying the number of positive seed facts. PRAVDA extracts far more facts than PROSPERA, particularly if the number of positive seeds is low. Both approaches achieve similar precision quality throughout the experiments (except for PROSPERA for the *isMarriedTo* relation with 10 positive seeds given). It can again be recognized that PRAVDA is slightly prone to generate more noise, particularly when the relation of interest is more complex. As an exapmle of the *isMarriedTo* relation, please consider the following text from our corpus:

> "The Scottish News of the World has published photographs which it claims show how Sir Paul McCartney is secretly dating Hollywood actress Rosanna Arquette - the virtual double of his estranged wife Heather Mills."

Since we do not use a dependency parser for efficiency reasons, there is no evidence that the two entities `Rosanna Arquette` and `Heather Mills` are not directly connected. Consequently, the surface string pattern "wife" links `Rosanna Arquette` with `Heather Mills` and identifies them (incorrectly) as a couple.

5.3 Results on Temporal Fact Extraction with Inclusion Constraints

In the following we will present the results of our methods on temporal fact extraction with inclusion constraints (yearly granularity). Since PROSPERA is not geared for harvesting temporal facts, we only report the performance of our method. We evaluate inclusion constraints for two relations:

- *joinsClubTemp* and *leavesClubTemp* imply *worksForClubTemp*
- *getsMarriedWithTemp* and *getsDivorcedFromTemp* imply *isMarriedToTemp*

The results are depicted in Table 5 (10 positive and negative seed facts for the first seven relations) with a special highlighting in dark grey of the relations *worksForClubTemp* and *isMarriedToTemp*. Without giving any seed facts to the *worksForClubTemp* and *isMarriedToTemp* relations, inclusion constraint are successfully applied to extract facts for the two relations.

We see that results with respect to the number of extracted facts and precision are not so good as for base fact extraction. This is not surprising for at least three reasons. First, our extraction currently operates on the sentence level, meaning that entity pair and temporal expression have to be in the very same sentence. Second, we are currently able to detect only explicit temporal expressions. Third, associating temporal expressions to the appropriate relation and/or entity/ies is sometimes complex for human, too.

Extraction of facts for the *leavesClubTemp* and *joinsClubTemp* relations are particularly difficult. Usually media coverage starts a long time before the actual transfer actually occurs. Consequently, this induces a lot of noise to the *leavesClubTemp* relation. In addition, both before mentioned relations are - of course - affected by unsupported rumors, which are common to sport news coverage.

Relation	# P/N Seeds	# Facts	Precision
isBornInTemp	10/10	3.471	82%
hasDiedInTemp	10/10	65	88%
winsAwardTemp	10/10	243	84%
joinsClubTemp	10/10	1.871	80%
leavesClubTemp	10/10	235	71%
getsMarriedWithTemp	10/10	76	78%
getsDivorcedFromTemp	10/10	13	71%
worksForClubTemp	0/0	2.086	86%
isMarriedToTemp	0/0	88	80%

Table 5: Temporal Fact Extraction with Inclusion Constraints

5.4 Joint Base and Temporal Fact Extraction

In the following we now investigate the results of joint base and temporal fact extraction. In contrast to the experiments before, we now combine the base and temporal graphs and extract the facts simultaneously. Table 6 compares the joint approach with the separate (for base facts resp. temporal facts) ones. For the sake of clarity, results of temporal fact extraction have been highlighted in grey. Experiments have been conducted with 10 positive/negative seed facts per base relation and 5 positive/negative seed facts per temporal relation (except *worksForClubTemp* and *isMarriedToTemp* in order to evaluate the impact of inclusion constraints).

As we can see from Table 6, particularly base fact extraction benefits from the joint apporach, while temporal fact extraction remains more or less unchanged. It is worth mentioning that the *worksForClub* relation increases a lot in terms of number of extracted facts as well as slightly with respect to precision. Even more, the number of extracted facts as well as the precision for the *worksForClub* relation is higher than its separate "counterpart", which is the 20-seed baseline compared with 10 seeds for *worksForClub* + 5 seeds for *joinsClubTemp* + 5 seeds for *leavesClubTemp* in the joint approach.

Finally, we study the impact of inclusions constraints on the joint approach. From the results in Table 7 we can see that inclusion constraints play an important role in the joint extraction approach. Both base relations (*worksForClub* and *isMarriedTo*) benefit from their inclusion, while precision slightly decreases for *worksForClub*. Accuracy for temporal relations (colored in grey) remains almost unchanged. Please note that no facts have been discovered for *worksForClubTemp* and *isMarriedToTemp* (indicated in dark grey in Table 7) as they have been extracted via inclusion constraints, only.

6. RELATED WORK

Relation extraction aims at detecting and classifying semantic relationships between entities typically from text. Many machine learning methods have been exploited to address this task. Supervised learning methods, such as [8, 29] suffer from a large amount of manually labelled training data. Semi-supervised methods (e.g. [28]) require less labelled data, and particularly [7] first attempted to apply label propagation for relation extraction. Recently distant supervision for relation extraction(e.g. [12, 26]) became popular. Instead of using a labelled corpus, the input of distant supervision

	Joint		Separate	
Relation	**# Facts**	**Precision**	**# Facts**	**Precision**
worksForClub	10.467	84%	4.882	84%
isBornIn	3.139	92%	368	96%
worksForClubTemp	2.761	84%	2.032	88%
isBornInTemp	3.184	84%	3.297	85%

Table 6: Comparison of Joint and Separate Fact Extraction

	W/ Constraints		W/O Constraints	
Relation	**# Facts**	**Precision**	**# Facts**	**Precision**
worksForClub	10.467	84%	5.657	90%
isMarriedTo	413	64%	480	64%
worksForClubTemp	2.761	84%	0	NA
isMarriedToTemp	98	82%	0	NA
joinsClubTemp	1.788	82%	1.840	82%
leavesClubTemp	249	64%	273	66%
getsMarriedWithTemp	73	77%	73	77%
getsDivorcedFromTemp	13	78%	13	73%

Table 7: Joint Extraction With and Without Constraints

only requires an existing knowledge base. These advances in information extraction and the success of knowledge communities like Wikipedia have enabled the largely automated construction of fairly comprehensive knowledge bases such as *DBpedia* [1], *YAGO* [17], *freebase.com*, *trueknowledge.com*, the TextRunner project [9, 24], the StatSnowball project [30], and others. None of these knowledge bases has a temporal dimension. They are all designed for time-invariant knowledge and built as snapshots in time. For example, while several of them know that people can have multiple spouses, none of them can capture the time intervals for the various marriages. Birth and death dates of people are available, though, as they can be extracted at the level of base-facts such as `Diego_Maradona bornOn 30-October-1960`. This is much simpler than the kind of temporal facts that tackled in this paper.

Temporal knowledge as a first-class citizen in knowledge bases has been addressed solely by a few prior papers: [27], [23], and [11]. [27] focuses on extracting business-related temporal facts such as terms of CEOs. It uses an NLP machinery, with deep parsing of every sentence, and machine-learning methods for classifiers for specifically interesting relations. It workes reasonably well, but is computationally expensive, requires extensive training, and cannot be easily generalized to other relations. [23] focuses on extracting relevant timepoints and intervals from semistructured data in Wikipedia: dates in category names, lists, tables, infoboxes. However, there is no support for processing free text. [11] uses training data with fine-grained annotations to learn an inference model based on Markov Logic. This involves using consistency constraints on the relative ordering of events. This machinery is computationally expensive and cannot easily be scaled up.

The NLP community has event extraction tasks in its SemEval workshop series [20], using representations such as TimeML and reference corpora such as Timebank [5]. The TARSQI toolkit [21] provides a suite of techniques for detecting events and temporal expressions, based on a dictionary of time-related adverbial phrases and regular-expression matching. [25] use a small set of seeds to learn pattern rules for various complex relations, such as joining (leaving) a job. The algorithm works on parsed input data, which requires preprocessing by a dependency parser. Moreover, temporal mentions associated with these relations are not considered.

Temporal facts or temporal information are also considered in other fields. For instance, [22] focuses on logics-based querying over uncertain temporal facts, but does not address the extraction and temporal fact harvesting process. There is also recent awareness of temporal IR: ranking of search results for keyword queries with temporal phrases [4, 14].

Label propagation belongs to a family of graph-based semi-supervised learning methods. The first LP method [31] assumed that initial labels are fully correct. This assumption is inappropriate for our setting, as the seed patterns are not fully correct, but only associated with a confidence value. The Adsorption algorithm [2] copes with noisy initial labels, and is successfully used in video recommendation. The MAD algorithm [18] optimizes the Adsorption algorithm. The MADDL algorithm [18] incorporates possible dependencies on different labels into the optimization function. However, these dependencies must be symmetric, in contrast to the asymetric inclusion constraints in our work. In a recent survey [19], MAD is regarded as the most effective label propagation algorithm across various data sets and experimental conditions. The nature of label propagation, which can be iteratively approximated by the Jacobi method, makes it nicely scalable [2]. For example, [15] describes a scalable implementation of label propagation on MapReduce to deal with Web-scale graphs.

7. CONCLUSIONS & OUTLOOK

This paper introduced a unified framework for harvesting base facts and temporal facts from textual Web sources. Our experimental results with news articles and Wikipedia pages demonstrate the viability and high accuracy of our approach. Moreover, for the base-facts case, we have shown much higher recall over one of the best state-of-the-art baselines, while achieving similar precision. We believe that extended forms of constrained label propagation, as developed in this paper, are a very promising alternative to prior methods like Max-Sat reasoning or sampling over Markov-Logic models or factor graphs.

Our extended LP method is nicely geared for scale-out on distributed platforms. So far, however, our experiments were at a scale where we did not need to utilize this property. Our future work aims at experiments with Web-scale datasets (e.g., the TREC ClueWeb corpus), and will explore the scalability behavior of our method.

Acknowledgements

This work is supported by the 7^{th} Framework IST programme of the European Union through the focused research project (STREP) on Longitudinal Analytics of Web Archive data (LAWA) under contract no. 258105.

8. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, pages 722–735, 2007.

[2] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW*, pages 895–904, 2008.

[3] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. Nonlinear programming: theory and algorithms. 1993.

[4] K. Berberich, S. J. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *ECIR*, pages 13–25, 2010.

[5] B. Boguraev, J. Pustejovsky, R. K. Ando, and M. Verhagen. Timebank evolution as a community resource for timeml parsing. *Language Resources and Evaluation*, pages 91–115, 2007.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313, 2010.

[7] J. Chen, D.-H. Ji, C. L. Tan, and Z.-Y. Niu. Relation extraction using label propagation based semi-supervised learning. In *ACL*, 2006.

[8] A. Culotta and J. S. Sorensen. Dependency tree kernels for relation extraction. In *ACL*, pages 423–429, 2004.

[9] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, 2008.

[10] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

[11] X. Ling and D. S. Weld. Temporal information extraction. In *AAAI*, 2010.

[12] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL/AFNLP*, pages 1003–1011, 2009.

[13] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, 2011.

[14] M. Pasca. Towards temporal web search. In *SAC*, pages 1117–1121, 2008.

[15] D. Rao and D. Yarowsky. Ranking and semi-supervised classification on large scale graphs using map-reduce. In *Graph-based Methods for Natural Language Processing*, pages 58–65, 2009.

[16] C. E. Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[17] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706. ACM, 2007.

[18] P. P. Talukdar and K. Crammer. New regularized algorithms for transductive learning. In *ECML/PKDD (2)*, pages 442–457, 2009.

[19] P. P. Talukdar and F. Pereira. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *ACL*, pages 1473–1481, 2010.

[20] M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, J. Moszkowicz, and J. Pustejovsky. The tempeval challenge: identifying temporal relations in text. *Language Resources and Evaluation*, 43:161–179, 2009.

[21] M. Verhagen, I. Mani, R. Sauri, J. Littman, R. Knippen, S. B. Jang, A. Rumshisky, J. Phillips, and J. Pustejovsky. Automating temporal annotation with tarsqi. In *ACL*, 2005.

[22] Y. Wang, M. Yahya, and M. Theobald. Time-aware reasoning in uncertain knowledge bases. In *MUD*, pages 51–65, 2010.

[23] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In *EDBT*, pages 697–700, 2010.

[24] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *WWW*, pages 635–644, 2008.

[25] F. Xu, H. Uszkoreit, and H. Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *ACL*, 2007.

[26] L. Yao, S. Riedel, and A. McCallum. Collective cross-document relation extraction without labelled data. In *EMNLP*, pages 1013–1023, 2010.

[27] Q. Zhang, F. Suchanek, and G. Weikum. TOB: Timely ontologies for business relations. In *WebDB*.

[28] Z. Zhang. Weakly-supervised relation classification for information extraction. In *CIKM*, pages 581–588, 2004.

[29] G. Zhou, M. Zhang, D.-H. Ji, and Q. Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *EMNLP-CoNLL*, pages 728–736, 2007.

[30] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*, pages 101–110, 2009.

[31] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

Appendix

Convexity Proof

We show that the function 5 is convex, following the convention of [16], we rewrite the estimated labels of $\widehat{\mathbf{Y}}$ into a vector $\widehat{\mathbf{y}}$ with length $n \times (m+1)$.

$$\widehat{\mathbf{y}} = (\widehat{\mathbf{Y}}_{11}, ..., \widehat{\mathbf{Y}}_{n1}, \widehat{\mathbf{Y}}_{12}, ..., \widehat{\mathbf{Y}}_{n2}, ..., \widehat{\mathbf{Y}}_{1(m+1)}, ..., \widehat{\mathbf{Y}}_{n(m+1)})^T \tag{9}$$

In the same way, the initial label assignment matrix $\mathbf{Y}$ can be converted into a vector $\mathbf{y}$. Following the same idea, the graph Laplacian $\mathbf{L}$ is block diagonal with the matrices $\mathbf{L_1}, ..., \mathbf{L_{m+1}}$. In our case, $\mathbf{L_i} = \mathbf{L}$.

$$\begin{bmatrix} \mathbf{L_1} & & & \\ & \mathbf{L_2} & & \\ & & ... & \\ & & & \mathbf{L_{m+1}} \end{bmatrix}. \tag{10}$$

And the diagonal matrix $\mathbf{S}$ is represented analogously.

$$\begin{bmatrix} \mathbf{S_1} & & & \\ & \mathbf{S_2} & & \\ & & ... & \\ & & & \mathbf{S_{m+1}} \end{bmatrix}. \tag{11}$$

Then the matrix $\hat{\mathbf{C}}$ implied by inclusion constraints has the same form as the unnormalized graph Laplacian, which is $\hat{\mathbf{C}} = \mathbf{D}^{\hat{\mathbf{C}}} - \mathbf{W}^{\hat{\mathbf{C}}}$. The elements $\mathbf{W}^{\hat{\mathbf{C}}}{}_{v^\ell v^{\ell'}}$ and $\mathbf{W}^{\hat{\mathbf{C}}}{}_{v^{\ell'} v^\ell}$ of affinity matrix $\mathbf{W}^{\hat{\mathbf{C}}}$ have the same positive weights $\mathbf{C}_{\ell\ell'} \cdot \mathbf{Y}_{v\ell'}$, if an inclusion constraint between labels ℓ and $\ell^{'}$ is included. The remaining entries of $\mathbf{W}^{\hat{\mathbf{C}}}$ are zeros. Then the degree matrix $\mathbf{D}^{\hat{\mathbf{C}}}$ is a diagonal matrix with $\mathbf{D}^{\hat{\mathbf{C}}}{}_{ii} = \sum_j \mathbf{W}^{\hat{\mathbf{C}}}{}_{ij}$.

With the new formulation of the problem, the optimization problem including inclusion constraints takes the form

$$\min_{\widehat{\mathbf{y}}} \Big[(\widehat{\mathbf{y}} - \mathbf{y})^T \mathbf{S} (\widehat{\mathbf{y}} - \mathbf{y}) + \mu_1 \widehat{\mathbf{y}}^T \mathbf{L} \widehat{\mathbf{y}} + \mu_2 (\widehat{\mathbf{y}} - \mathbf{r})^T \mathbf{I} (\widehat{\mathbf{y}} - \mathbf{r}) + \mu_3 \widehat{\mathbf{y}}^T \hat{\mathbf{C}} \widehat{\mathbf{y}} \Big] \quad s.t.\ \widehat{y}_v^l \geq \widehat{y}_v^{l'}$$

The Hessian matrix of the objective function with respect to $\widehat{\mathbf{y}}$ is

$$\mathbf{S} + \mu_1 \mathbf{L} + \mu_2 \mathbf{I} + \mu_3 \hat{\mathbf{C}} \tag{12}$$

which is positive definite if $\mu_2 > 0$ ($\hat{\mathbf{C}}$ is positive semi-definite following the same reason as for the unnormalized graph Laplacian). Finally, as the feasible region is defined by linear constraints, the optimization problem is convex.