PRAVDA-live: Interactive Knowledge Harvesting

Yafang Wang, Maximilian Dylla, Zhaochun Ren, Marc Spaniol and Gerhard Weikum
Max-Planck-Institut für Informatik
Saarbrücken, Germany
{ywang|mdylla|zren|mspaniol|weikum}@mpi-inf.mpg.de

ABSTRACT

Acquiring high-quality (temporal) facts for knowledge bases is a labor-intensive process. Although there has been recent progress in the area of semi-supervised fact extraction, these approaches still have limitations, including a restricted corpus, a fixed set of relations to be extracted or a lack of assessment capabilities. In this paper we introduce PRAVDA-live, a framework that overcomes these limitations and supports the entire pipeline of interactive knowledge harvesting. To this end, our demo exhibits fact extraction from ad-hoc corpus creation, via relation specification, labeling and assessment all the way to ready-to-use RDF exports.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation

Keywords

Interactive Knowledge Harvesting, Label Propagation

1. INTRODUCTION

In recent years, automated fact extraction from Web contents has seen significant progress with the emergence of freely available knowledge bases, such as DBpedia [1], YAGO [10], TextRunner [5], or ReadTheWeb [3]. These knowledge bases are constantly growing and contain currently (by example of DBpedia) several million entities and half a billion facts about them.

To expand these knowledge bases even further, extraction of factual knowledge from free text is an essential ingredient, since most knowledge is not available in (semi-)structured formats. Systems like NELL [4] or Prospera [8] are tackling this task. Additionally, most knowledge bases come with a certain thematic focus and relations specific to that, i.e. YAGO's most populated relations are about persons. This calls for the creation of even more knowledge bases in previously uncovered domains or contexts.

However developing a fact extraction system from scratch is a cumbersome endeavor, because even subproblems such as entity disambiguation or pattern mining are active research topics on their own, requiring experts to be solved. Hence, to lower the bar of customized fact extraction from free text, we introduce an interactive knowledge harvesting system called PRAVDA-live[1]. Our system is an out-of-the-box solution to address this issue by covering all subproblems within a web-based interface. This allows user of all provenance (from greenhorns to experts) to perform fact extraction on their own.

Problem Setting. Let us consider the following sentence from Wikipedia: "*Albert Einstein was born in Ulm, in the Kingdom of Württemberg in the German Empire on 14 March 1879.*" We can nail down parts of its semantics by using the relation *bornIn*, which connects a subject of type person with an object of type location and maybe an additional temporal annotation. Given that, our system can be employed to automatically distill the fact $bornIn(Einstein, Ulm)@14.03.1879$ from this sentence.

State of the Art. DBpedia [1] and YAGO [10] harvest semi-structured data from Wikipedia, such as tables, infoboxes, or categories, not focusing on the extraction from free text. The projects NELL/ReadTheWeb [3] and TextRunner/ReVerb [5, 6] tackle free text, have an online query interface for their stored facts, but do not allow interactive fact extraction. Freebase [2] is very related to our work, since users are encouraged to add facts. Nevertheless, there is no support for extraction from text documents. Moreover, there are numerous crowd-sourcing activities for data annotation such as KiWi [9] or Semantic Wiki [7]. However, those efforts do not aim at fact extraction for knowledge bases in terms of clarity and scale.

Contributions. We present a system called PRAVDA-live, which supports fact extraction of user-defined relations from ad-hoc selected text documents and ready-to-use RDF exports. Further features include the support for temporally annotated relations, custom or mined extraction-patterns, and a constraint solver being able to clean extracted facts by inter-fact constraints.

2. FRAMEWORK AND ALGORITHMS

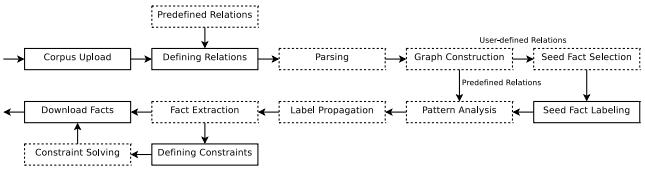

Figure 1: System Workflow

Framework. Our system is structured as depicted in Figure 1, where boxes with continuous lines require user-interaction. As a

[1] http://www.mpi-inf.mpg.de/yago-naga/pravda/

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

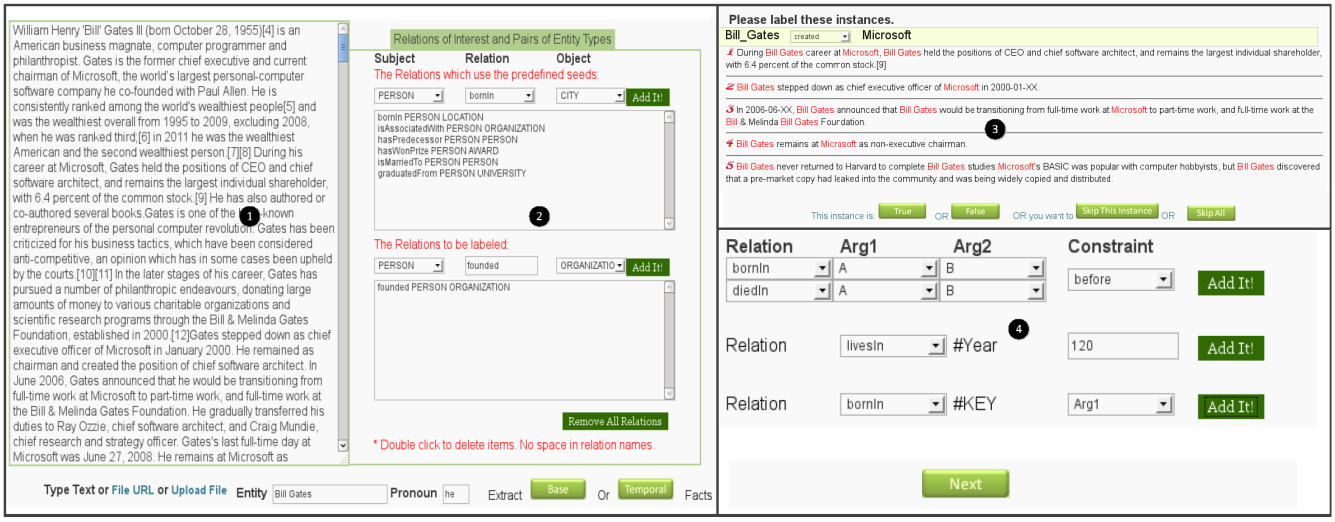

Figure 2: User Interface of PRAVDA-live

first step, the user uploads a corpus to be subject to the extraction process. Then, the user either selects predefined relations or defines customized relations or both. What follows are the parsing and graph construction stages performed in the backend. If customized relations are present, the next step is the seed fact selection followed by manual labeling of these. Afterwards we continue with pattern analysis and label propagation, eventually yielding extracted facts. Finally, there is the option to define and apply constraints to the extracted facts, or to download the facts immediately.

2.1 Algorithms

Parsing. Once the corpus is uploaded and the relations of interest are known, our system proceeds with the parsing stage. Following [13] entities are recognized and disambiguated by leveraging YAGO [10]. The surface string between two entities are lifted to patterns by considering n-grams of nouns, verbs converted to present tense, and prepositions. A detailed description of patterns is beyond the scope of this paper, however we refer the interested reader to [8]. Finally, we detect temporal expressions by regular expressions.

Graph Construction. As in [13] we construct an undirected graph $G = (V, E)$ comprising two types of vertices $V = V_e \dot{\cup} V_p$. The former set V_e contains one vertex per entity pair discovered in the corpus and the latter set V_p has one vertex per pattern. Edges between V_p and V_e are added, if an entity pair occurs in a pattern. Additional edges between vertices in V_p are derived from similarities among patterns. An example graph is shown in Figure 3, where oval vertices belong to V_e and box-shaped vertices are members of V_p.

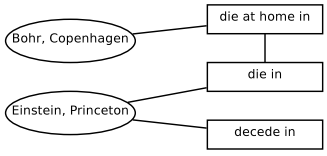

Figure 3: Example Graph

Seed Fact Selection. In previous works seed facts were chosen entirely manually, which requires an understanding of the label propagation procedure. To ease usability of our system, we develop a novel ranking algorithm returning the entity pairs to be labeled by the user. The algorithm is presented in Section 2.2. This stage is skipped for problem instances with only predefined relations, where labeled seed facts are available in our system.

Pattern Analysis. In this stage, the labeled seed facts are employed to compute an initial weighting of the patterns, being derived from frequency counts of both the patterns as well as the entity pairs. A more detailed description is available in [8].

Label Propagation. Building on [13] we utilize Label Propagation [11] to determine the relation expressed by each pattern. Here, the labeled seed facts and patterns serve as input. Label Propagation is a semi-supervised learning algorithm, where the supervision results from the labeled seed facts. It passes labels on the previously described graph (see Figure 3), where each label corresponds to a relation.

Fact Extraction. After Label Propagation has terminated, the entity pair vertices which hold a relation's label weighted above a threshold form a fact.

Constraint Solving. Given user-defined constraints and a set of extracted facts, we intend to select a maximal consistent subset of facts. The resulting optimization problem is encoded into an integer linear program as in [12].

2.2 Seed Fact Selection

Our seed fact selection procedure acts on the graph introduced in Section 2. By construction the graph consists of disconnected components corresponding to a different type pair each. Considering a single connected component, we cluster its vertices reflecting patterns in the following manner: If two patterns have an identical first verb and last preposition, they belong to the same cluster. For example, in the disconnected component of Figure 3, we cluster '*die in*' and '*died at home in*'. More formally, a disconnected component is defined as $C = (V_e \dot{\cup} (\dot{\bigcup}_i V_{p,i}), E)$, where V_e are the vertices standing for entity pairs, and each $V_{p,i}$ represents a cluster of vertices embodying patterns. With respect to Figure 3, we have $V_e = \{(Bohr, Copenhagen), (Einstein, Princeton)\}$ and $V_{p,0} = \{die\ at\ home\ in, die\ in\}$, $V_{p,1} = \{decede\ in\}$, for instance. To each disconnected component we apply Algorithm 1 implementing a greedy strategy. The loop in Line 3 repeatedly cycles through all clusters of pattern vertices $V_{p,i}$ beginning with the largest cluster. It stops when k seed facts have been determined. For

Algorithm 1 Seed Fact Selection

Require: Component $C = (V_e \dot{\cup} (\dot{\bigcup}_i V_{p,i}), E)$, number of seeds k
1: $S = \emptyset$ ▷ Seed facts to return
2: **while** $|S| < k$ **do**
3: **for** $V_{p,i} \in C$ by decreasing $|V_{p,i}|$ **do**
4: $v_e := \text{argmax}_{v_e \in V_e \setminus S, \exists v_p \in V_{p,i} : (v_e, v_p) \in E} degree(v_e)$
5: $S := S \cup \{v_e\}$
6: **if** $|S| \geq k$ **then**
7: **break** ▷ Break loop in Line 3
8: **return** S

each cluster we add an entity-pair-vertex v_e as seed, which has maximum degree and is connected to $V_{p,i}$ (Line 4). In Figure 3 the largest cluster is $V_{p,0} = \{die\ at\ home\ in, die\ in\}$ where the algorithm selects $(Einstein, Princeton)$ since its degree is maximal.

3. SYSTEM IMPLEMENTATION

PRAVDA-live is implemented in Java, where Apache Tomcat[2] acts as Webserver. While parsing text documents we employ OpenNLP [3] for part of speech tagging, converting verbs to present tense and stemming nouns. Furthermore, the integer linear program tackling the constraints utilizes Gurobi[4]. All data is managed by a PostgreSQL[5] database.

3.1 User Interface

Figure 2 shows screenshots of PRAVDA-live's user interface.

1. **Corpus Upload.** The user interface offers the opportunity for both pasting text into a text field and uploading larger text files. In Figure 2 the text field holds excerpts from Einstein's Wikipedia article.
2. **Defining Relations.** By investigating the text on the left, the user can define relations of interest. There are two lists of relations. The upper list holds exclusively predefined relations, which is *bornIn* typed by *person* and *location* in the screenshot. On the other hand, the lower list may contain both predefined and user-defined relations, since on these the seed fact selection process will be invoked.
3. **Seed Labeling.** During the labeling process text snippets are displayed, where recognized entities are marked in red. It is the user's task to select the correct relation (here '*created*') connecting both entities. Seed facts can be labeled as true or false, as indicated by the respective buttons.
4. **Defining Constraints.** As a final feature, PRAVDA-live allows the specification of constraints to be applied to the extracted facts. In the example screenshot, we require birthdays to precede dates of death and we disallow the time annotation of *livesIn* to exceed 120 years in length. As for non-temporal constraints, we enforce *bornIn* to be functional, such that persons can be born in at most one location.

4. DEMONSTRATION SCENARIOS

In order to showcase the entire pipeline of our interactive knowledge harvesting system PRAVDA-live, we have prepared two dedicated demo scenarios: **Ad-hoc Fact Extraction for YAGO** and **Fact Extraction on Customized Relations**. Users may freely interact with our system.

Ad-hoc Fact Extraction for YAGO. In the first scenario we will enable users to harvest (temporal) facts based on the relations supported by YAGO. To this end, users will be able to either upload a document collection or paste a text document in the user interface of PRAVDA-live for a subsequent fact extraction. After that, the facts can be evaluated via the assessment interface. Finally, the so created fact set can be exported as an RDF document, which is ready to be fed into YAGO.

Fact Extraction on Customized Relations. The second scenario allows users to harvest facts from customized relations. This use case is of particular interest for those, who want inject RDF exports from PRAVDA-live into a proprietary knowledge base. To this end, we demonstrate the specification of additional relations and the corresponding labeling of a small number seed facts. Further, we showcase the bulk processing feature of PRAVDA-live with a subsequent assessment. The demo concludes with a RDF export of the extracted facts.

[2] http://tomcat.apache.org/
[3] http://opennlp.apache.org/
[4] http://www.gurobi.com/
[5] http://www.postgresql.org/

Acknowledgments

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

5. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC*, pages 11–15, 2007.
[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.
[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313, 2010.
[4] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka, and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM 2010*, pages 101–110, 2010.
[5] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, 2008.
[6] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545, 2011.
[7] M. Krötzsch and D. Vrandecic. Semantic mediawiki. In *Foundations for the Web of Information and Services*, pages 311–326, 2011.
[8] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, 2011.
[9] S. Schaffert, J. Eder, S. Grünwald, T. Kurz, M. Radulescu, R. Sint, and S. Stroka. Kiwi - a platform for semantic social software. In *SemWiki*, 2009.
[10] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706. ACM, 2007.
[11] P. P. Talukdar and K. Crammer. New regularized algorithms for transductive learning. In *ECML/PKDD*, pages 442–457, 2009.
[12] Y. Wang, M. Dylla, M. Spaniol, and G. Weikum. Coupling Label Propagation and Constraints for Temporal Fact Extraction. In *ACL*, 233-237, 2012.
[13] Y. Wang, B. Yang, L. Qu, M. Spaniol, and G. Weikum. Harvesting facts from textual web sources by constrained label propagation. In *CIKM*, pages 837–846, 2011.