LUKe and MIKE: Learning from User Knowledge and Managing Interactive Knowledge Extraction

Steffen Metzger
Max-Planck-Institute for Informatics
Campus E1 4
Saarbrücken, Germany
smetzger@mpi-inf.mpg.de

Michael Stoll
Institute for Visualization and Interactive Systems (VIS),
University of Stuttgart
Stuttgart, Germany
michael.stoll@vis.uni-stuttgart.de

Katja Hose
Max-Planck-Institute for Informatics
Campus E1 4
Saarbrücken, Germany
hose@mpi-inf.mpg.de

Ralf Schenkel
Saarland University and MPI Informatics
Saarbrücken, Germany
schenkel@mmci.uni-saarland.de

ABSTRACT

Semantic recognition and annotation of unqiue enities and their relations is a key in understanding the essence contained in large text corpora. It typically requires a combination of efficient automatic methods and manual verification. Usually, both parts are seen as consecutive steps. In this demo we present MIKE, a user interface enabling the integration of user feedback into an iterative extraction process. We show how an extraction system can directly learn from such integrated user supervision. In general, this setup allows for stepwise training of the extraction system to a particular domain, while using user feedback early in the iterative extraction process improves extraction quality and reduces the overall human effort needed.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Evaluation/methodology, Graphical User interfaces, Natural language*; I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge Acquisition*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language Parsing and Understanding*

Keywords

Information Extraction, Knowledge Acquisition, Learning, GUI, User Feedback, Web Service

1. INTRODUCTION

With the growing amount of textual information digitally available, an efficient analysis of large amounts of natural language text to grasp the essential content relevant for a particular interest becomes increasingly important.

In several areas, e.g. the humanities, semantic information is traditionally attached to source documents by expert users, e.g. by annotation, in order to allow automatic evaluation and management of the data and the contained semantic information. However human processing cannot efficiently deal with the large amount of documents digitally available. There is a growing number of automatic information extraction systems [1, 3, 6], which can support humans in grasping the essential semantic information contained in natural language texts. Typical systems support named entity recognition (NER), e.g. identifying *"Barack Obama"* and *"US President"* as references to the same unique entity, and relation extraction, e.g. extracting an instance of an abstract *bornIn* relation from the text *"Barack Obama's birthplace Hawaii"*. Such relational information is often represented in RDF triple form, e.g. (`Obama`,`bornIn`,`Hawaii`). Information extraction systems often follow an iterative approach, expanding an initial knowledge base step by step. User involvement is usually required to provide the initial basic domain knowledge, e.g. by providing examples for entity and relation recognition. Although these extraction systems can reach relatively good precision values in general[6], high precision usually comes at the price of lower recall, such that many relationship instances are not found or at least not found in all documents where they occur. Additionally, for many use-cases there is a need for quality guarantees. This is typically achieved by an additional manual filtering step after the extraction is complete.

In this paper, we aim to close the gap between manual annotation and automatic extraction by integrating large portions of initial domain knowledge generation and manual quality control into the extraction process. We present *LUKe*, an extraction system based on [6] that can integrate user feedback on-the-fly, and *MIKE*, a user-interface to control the extraction system, present the extraction results to

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.
ACM 978-1-4503-1156-4/12/10.

a human user and allowing for extension and correction of recognized knowledge. The seemless integration of user feedback into the iterative extraction process allows the system to learn early on. Hence, quality achieved is optimized while human effort needed to ensure a given quality level is minimized.

2. RELATED WORK

There are several other approaches to visualize semantic information [1, 2, 4, 7, 8]. Traditional tools to model ontologies, e.g. WebProtégé [7], lack the capability to link information extracted from documents back to their sources[2]. On the other hand, there are general semantic frameworks like UIMA[4] and GATE[1] with especially the latter allowing user interaction by modeling the extraction process as a workflow where each step produces some sort of document representation, e.g. a token stream, and this representation can potentially be modified before being fed into the next step of an extraction pipeline. Semantic Wikis have a stronger focus on user driven data generation, yet they typically lack information extraction capabalities to support users in the annotation process. However, there has been work on integrating existing ontologies and reasoning capabilities as a background model into semantic wiki systems[8]. The most similar vision of an iterative learning process is probably shared by the KiWi project[1]. It envisions an integration of information extraction and knowledge management with Wiki technology[5]. While we share a similar architecture as envisioned in the project, our frontend aims at a user base that is not accommodated to Wikis. Still, since backend and frontend are independent, a Wiki-frontend could as well be attached to our extraction API.

3. FUNCTIONALITY

A typical user workflow consists of two main iteratively repeated steps: 1) Information Extraction 2) Evaluation and Modification of extracted information After each extraction phase, the user can give feedback, such that in the next extraction iteration the system can learn from user corrections and apply any new insights on the whole corpus.

However, initially a user first has to select source files by providing their URIs (e.g. web urls or WebDAV directories), which will be added to a list of project files to work on. Next, the user may configure the extraction run including the choice of domain knowledge used and which extraction steps (entity recognition, relation recognition) shall be performed. Once the extraction process is started *MIKE* displays status information on the extraction progress. When the extraction process has finished, results are retrieved from the extraction system and provided as a listed overview, which may be grouped and sorted by files, entities or statements (Figure 1, area 1). In case of grouping by files, the extracted entites and the extracted statements within each file are indicated. In both the other cases, for each entity/statement the files where it is mentioned are listed.

A click on a list item opens the corresponding document within an editor component. There, its textual content is displayed enriched with highlighted entity and statement mentions (Figure 1, area 3). This way, a user directly sees the extracted information in context allowing her to judge correctness and completeness of the extraction results. Both entity and statement mentions can be filtered by a confidence threshold (area 2). A click on an entity or statement mention provides further information, such as its author (the user or system name that found or last edited the mention), the recognition confidence and its context (area 4). In case of an entity mention also the referenced entity is shown, while in case of a statement mention all relations recognized between the included entities (at that mention) are listed. Entity mentions may be corrected by changing the referenced entity and/or by adjusting their bounds. For a statement mention a user may support/refute a given relation and/or add a new one. Both types of mention can be added by selecting a text part and providing an entity reference or at least one relation, respectively. After giving feedback a user may restart the extraction on all or a subset of documents and the system can re-evaluate all findings based on the given feedback.

Finally, the frontend allows for export of edited files in an annotated XML format or of all extracted information in a single dump.

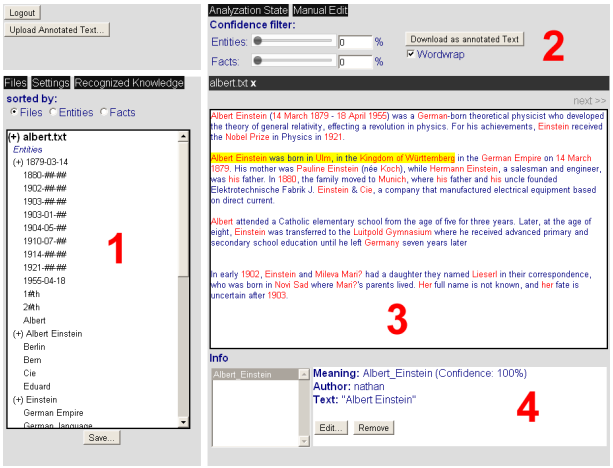

Figure 1: Overview with opened Document

LUKe Learning Effects.

In our setup we use a pattern based extraction system based on [6]. However, the system has been extended to learn from user feedback and integrate it into its own iterative extraction process. This way the extraction system learns from corrections of entity disambiguations to do it correctly at other occurrences of the same entity representation. For instance, if *"US President"* has been manually set to indicate the entity `BarackObama` then 1) in other cases where *"US President"* occurs the system will tend towards interpreting these as references to `BarackObama` and 2) all other entity references that are ambiguous, but might refer to Obama are more likely associated with the entity `BarackObama`. Additionally, if the system did not even consider `BarackObama` as the entity being adressed by *"US President"*, e.g. because its database was outdated, then it will consider this new possibility also in other texts. For the relations the system manages pattern–relation associations with confidence values, e.g. it might consider "X was born in Y" as an expression of the `bornIn` relation between two entities X and Y or "X 's newest master piece Y" as a (less reliable) pattern for a `directed` relation, but it might also indicate a `actedIn`,

[1]http://www.kiwi-project.eu/

`wrote` or `painted` relation. If a user adapts the interpretation of an ambiguous pattern, the system can modify its confidence in the corresponding pattern–relation association and thus learn about the typcial interpretation within the current domain. Additionally, as relations are typed, choosing correct entities, e.g. by fixing disambiguation decisions or by setting the relation, allows the system to also adapt the entity's type information. For instance, if the system only knows about Quentin Tarantino being a director, a manually added instance of an `actedIn` instance can lead to him being also known as an actor.

4. ARCHITECTURE

The architecture is divided into three main layers.

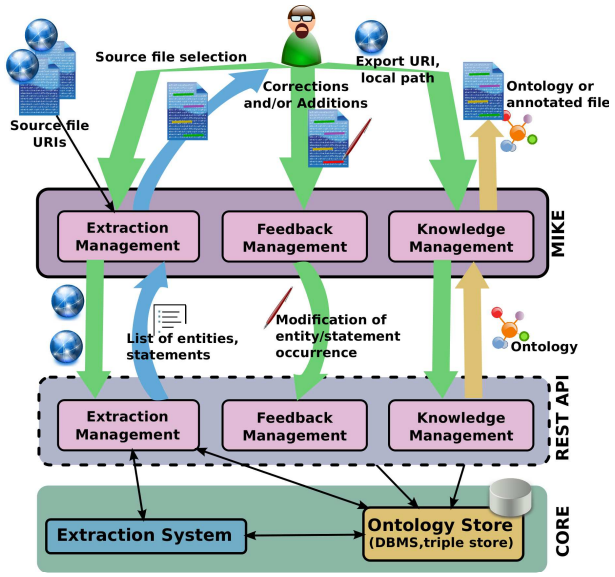

Figure 2: Architecture

The *CORE* layer consists of an adapted extraction system and an efficient ontology store, such as a database or a triple store. On top of these core components a *REST Web service API* provides access to extraction, feedback and knowledge management. That is, extraction jobs can be started via the interface, results can be obtained, feedback be provided and snapshots of all information extracted so far can be exported in an ontological format. Information is exchanged either in HTML, XML or JSON format. Extraction results mainly consist of entity and statement occurrences. Each entity occurrence's main components are 1) the position in the text, 2) the original text and 3) the entity reference candidates along with confidence values. Similarly, statement occurrences mainly consist of 1) the position in the text, 2) the two entities involved and 3) the possible interpretations.

Finally, *MIKE* provides a web-based user-interface. It wraps the functionality provided by the REST API in an easily understandable graphical frontend.

The architecture is modular in nature, e.g. any extraction system supporting the REST-API interface could be used and similarly another visualisation frontend could access the interface. In our demo setup we use *LUKe*, a modified version of [6], and a PostgreSQL database as core components. While in the demo a single system hosts all components (potentially except the database), the extraction could also be run in a distributed way. Both the extraction system in our setup as well as the REST-API are written in Java with the latter being run on a Tomcat server. *MIKE* is implemented in PHP using cURL and openSSL for communication with the REST-API.

5. DEMO OUTLINE

In our demonstration we will showcase the use of the user interface and the underlying learning extraction system by selecting a couple of documents, running the extraction, editing some entity and relation occurrences and then re-running the extraction to show the learning effects. A video outlining this procedure is available at `http://bit.ly/MdsU8F`.

6. ACKNOWLEDGMENTS

This work has been partially funded by the BMBF (German Federal Ministry of Education and Research) through the WisNetGrid project[2].

7. REFERENCES

[1] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[2] S. Elbassuoni, K. Hose, S. Metzger, and R. Schenkel. ROXXI: Reviving witness dOcuments to eXplore eXtracted Information. *Proceedings of the VLDB Endowment*, 3, 2010.

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

[4] D. Ferrucci and A. Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, Sept. 2004.

[5] P. Smrz and M. Schmidt. Information extraction in semantic wikis. In C. Lange, S. Schaffert, H. Skaf-Molli, and M. Völkel, editors, *SemWiki*, volume 464 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[6] F. M. Suchanek, M. Sozio, and G. Weikum. SOFIE: A Self-Organizing Framework for Information Extraction. In *International World Wide Web conference (WWW 2009)*, New York, NY, USA, 2009. ACM Press.

[7] T. Tudorache, N. F. Noy, S. M. Falconer, and M. A. Musen. A knowledge base driven user interface for collaborative ontology development. In P. Pu, M. J. Pazzani, E. André, and D. Riecken, editors, *IUI*, pages 411–414. ACM, 2011.

[8] D. Vrandecic and M. Krötzsch. Reusing ontological background knowledge in semantic wikis. In M. Völkel, S. Schaffert, and S. Decker, editors, *Proceedings of the First Workshop on Semantic Wikis – From Wikis to Semantics*, Budva, Montenegro, Juni 2006.

[2] `http://www.wisnetgrid.org/`