SCHNÄPPER: A Web Toolkit for Exploratory Relation Extraction

Thilo Michael and **Alan Akbik**
Technische Universität Berlin
Einsteinufer 17, 10587 Berlin
{thilo.michael,alan.akbik}@tu-berlin.de

Abstract

We present SCHNÄPPER, a Web toolkit for *Exploratory Relation Extraction*. The tool allows users to work with a very large corpus of text to identify relations of interest in an exploratory and highly interactive fashion. With this demonstration, we show how users can easily explore a corpus with pattern queries that find binary relations in text and how automatically computed suggestions can be used to guide the exploration process. We also show how extractors can be created once a relation of interest is identified. This paper describes the Web toolkit and illustrates its usage.

1 Introduction

Relation Extraction (RE) is the task of extracting instances of semantic relations between entities in unstructured data such as natural language text. Common examples are the BORNIN relationship between a person and its birthplace, or the CHILDOF relation between a parent and its child. A principal challenge in RE is how to build high quality extractors for a given set of relations at minimal effort.

One line of approaches to RE are *rule-based*, where users manually define rule-sets consisting of extraction patterns that if observed point to instances of a relation of interest. Advantages associated with rule-based RE is that there is a high level of direct control over the extraction process, where (ideally) rule-writers build interpretable and maintainable rule-sets, enabling both the extension and error analysis of rule-based extractors (Chiticariu et al., 2013). Indeed, in a number of recent works, rule-based RE approaches have been found to outperform previous machine-learning based state-of-the-art systems, for tasks such as temporal expression detection (Strötgen and Gertz, 2010) and OpenIE (Del Corro and Gemulla, 2013).

Exploratory search for relations. Recently, (Akbik et al., 2014) introduced the paradigm of *Exploratory Relation Extraction* and argued that workflows and tooling can be developed in such a way as to enable an interactive and open ended search for relations. The key ideas are that extraction patterns should be very easy to define and quick to test, much in the same way as exploratory keyword queries in a Web search engine (Marchionini, 2006). In addition, tooling can use statistics computed from the available data and previous user interactions to guide the user through suggestions for query refinements. Such data-guidance, coupled with a high degree of interactivity, would allow users to start the search for relations with a relatively vague information need and progressively refine their queries until a relation of interest is identified and an appropriate extractor is created.

Contributions. With this demo, we present SCHNÄPPER, a Web-based tool for Exploratory Relation Extraction (ERE) that demonstrates the incremental, data-guided workflow introduced in (Akbik et al., 2014). By making the demonstrator publicly available, we aim to showcase the ease-of-use and intuitive nature of the proposed approach. The demo is intended to underline a central claim of ERE, which is that non-experts can use it to easily explore a corpus for relational information and build extractors. Additionally, by using a large portion of the CLUEWEB09[1] corpus as dataset, we aim to highlight the applicability of such an approach to very large datasets.

Paper outline. We first give a quick overview over the ERE workflow in Section 2. We then present SCHNÄPPER, our Web interface (Section 3) and walk through an example workflow with the tool. We then briefly give an overview over related work

[1] http://www.lemurproject.org/clueweb09/index.php

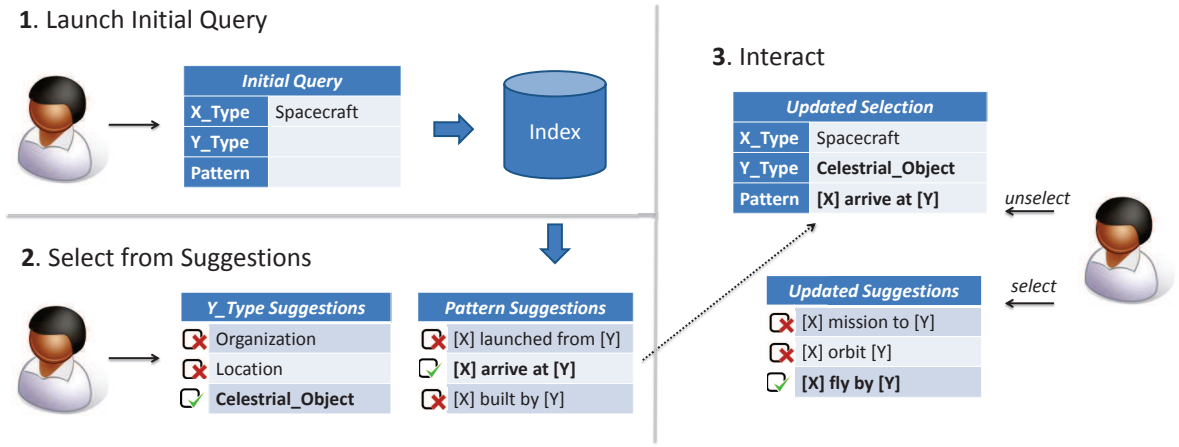

Figure 1: Illustration of the Exploratory Relation Extraction example workflow discussed in Section 2.2.

and give an outlook of possible future additions to the toolkit and the method itself.

2 Exploratory Relation Extraction

We demonstrate an approach to finding *binary relations* in text that has been proposed in (Akbik et al., 2014). Each relation holds between two entities: a *subject* and an *object* entity. Users explore a corpus for information by selecting and composing extraction patterns.

2.1 Pattern Language

Extraction patterns consist of two components:

1. Dependency subtrees. The first component is the lexico-syntactic pattern that connects two entities in a sentence. Here, we allow *arbitrary* subtrees in a sentence's dependency tree, as long as they span two entities of interest. To generalize the patterns, they are stemmed and the two entities are replaced by the placeholders `[X]` and `[Y]`. Examples of subtree patterns are `[X] and [Y] married` and `[X] 's father [Y]`[2]. However, since any subtree is a possible pattern, many subtrees with less obvious meanings are also possible; in the end, it is up to the user to make the decision which patterns are relevant and which are not.

2. Entity type restrictions Optionally, patterns may be further restricted to match only entities of certain fine-grained types, such as PERSON, LOCATION, LANGUAGE or MOVIE. The type restrictions may be set individually for each subject and object entities. Since the subject is replaced with the placeholder `[X]` in a pattern, its restriction is referred to as *X_Type*, while the object restriction is referred to as *Y_Type*.

Preemptive pattern extraction. Following the idea of preemptive Information Extraction (Shinyama and Sekine, 2006), we pre-extract and store all subtrees and entity types from a given corpus for each sentence with at least two named entities. This allows not only fast retrieval of matching entity pairs for a given set of subtrees and type restrictions, but also allows us to compute pattern correlations over the entire dataset for the presently selected setup. In the next section, we show how fast retrieval and pattern correlations are used to aid the exploration process.

2.2 Example Workflow

We illustrate the exploration process with an example workflow, the first steps of which are depicted in Figure 1. Assume that our user is interested in relations that involve "spacecraft", but is unsure of what types of relations may be found for such entities in the given corpus.

Initial query (1). The user starts by issuing an initial query that is strongly underspecified: By setting *X_Type* to SPACECRAFT and leaving the *Pattern* and *Y_Type* fields in the query unspecified, the user searches for all sentences that contain at least one entity of the desired type. At this point, there are no other restrictions to the query with regards to patterns or object entity types.

Explore by reacting to suggestions (2). After issuing the query, the system responds with both a list of sentences that match the query (not illustrated in Figure 1) and well as, more importantly,

[2]For the purpose of readability, we do not display the deep syntactic information from the subtrees. Instead, we only show the lexical portion of the patterns. Here, some verbs, such as participles and gerunds, are not stemmed for readability purposes.

suggestions for patterns and object entity type restrictions that correlate with the user query.

The user can now choose from the suggestions: For instance, by selecting the object type LOCATION and the pattern `[X] launched from [Y]`, the user may direct the exploration process towards relations that indicate locations (cities, countries, sites) from which a spacecraft was launched. Similarly, by choosing ORGANIZATION as object type and `[X] built by [Y]` as pattern, the user may select organizations (contractors, space agencies) that constructed or designed spacecraft as the focus of interest.

In the example shown in Figure 1, the user instead selects the object type CELESTIALOBJECT and the pattern `[X] arrive at [Y]`. This directs the search towards relations that indicate spacecraft missions to celestial objects.

User interactions (3). This user interaction updates both the query as well as the suggestions for patterns and restrictions. Now pattern suggestions are more specific to the previous selection; For instance, by selecting either the pattern `[X] orbit [Y]` or `[X] fly by [Y]`, the user can specify relations for spacecraft that have achieved orbit around celestial objects, or have made flybys. By following a process of querying, inspecting results, selecting and unselecting subtrees and restrictions, the user can interactively explore the given corpus for relations of interest. Once an interesting relation is identified, the user can use the same approach to build an extractor by compiling a list of relevant patterns from the suggestions. Typically, the more patterns a user selects, the higher the recall of the created extractor will be.

Store extractor. When the user has identified an interesting relation and selected a list of relevant patterns, she can export the extraction results (i.e. all relation instances found by the extractor). The user can also save the extractor and provide a descriptive name for the relation for possible later reuse.

3 Web Demonstration

We now present SCHNÄPPER[3], our Web toolkit for Exploratory Relation Extraction.

[3]The tool was named after the *Petroicidae* famliy of birds, which in German are called *Schnäpper*. This name stems from the verb *schnappen* (Schmitthenner, 1837), which translates as "*to grab*" or "*to catch*". We found this fitting since the tool is used to "grab" or "catch" information.

3.1 Web Interface

In order to make the use of SCHNÄPPER as straightforward as possible, the user interface is clearly structured into four panels that fit onto one screen. The top half of the screen consists of three panels in which the user can select patterns and entity type restrictions. The bottom half of the screen is the result panel which displays a sample of extraction results for the currently selected patterns and entity type restrictions. See Figure 2 for the screen and a breakdown of the panels, which we explain in more detail in the following:

Pattern panel (1) Of the three panels in the upper half of the screen, the pattern panel assumes the center stage. Here, the user can enter keywords in the search field to find appropriate patterns. If at least one user interaction has already been made (e.g. one pattern or type restriction selected), a list of pattern suggestions is presented in gray. Single clicking on a pattern suggestion gives a small number of example sentences and entity pairs for which this pattern holds (this is illustrated in field **(6)** in Figure 2). Double-clicking on a pattern adds it to the extractor; it is then highlighted blue and suggestions as well as the result panel are updated to reflect the selection. By double-clicking on a selected pattern, users may remove it again from the selection.

Entity type restriction panels (2) Extractors may also have entity type restrictions which restrict lexico-syntactic patterns to only apply to entities of certain types. The top right and top left panels are used to define restrictions for the subject and object of a binary relation respectively. Here, users have a choice between three different ways of selecting entity type restrictions. The first and default option is to use FREEBASE entity types (Bollacker et al., 2008). I.e. the user might select the subject of a relation to be only of the FREEBASE type SPACECRAFT, ORGANIZATION or CELESTIALOBJECT.

The user might also restrict a relation to one specific entity. For instance, by restricting the object of a BORNIN relation to be the country "Finland", the extractor will only find persons born in Finland.

Finally, the user can restrict entities to be only those found with a previously created extractor. Users can embed extractors in this way to find more complex relations. For instance, an extrac-

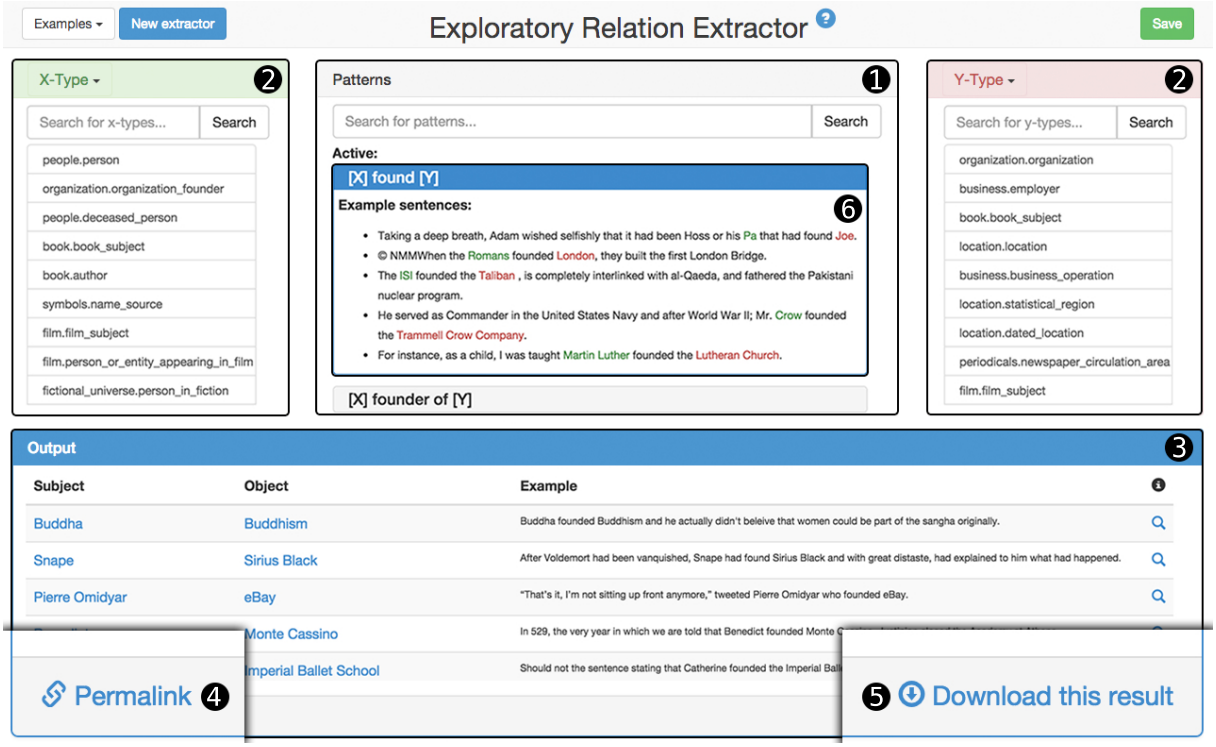

Figure 2: Screen capture of the SCHNÄPPER tool showing the *pattern panel (1)* with an activated pattern showing a list of example sentences (6), the *entity type restriction panels (2)* and the *result panel (3)*. The *permalink button (4)* and the *download button (5)* are located at the bottom.

tor that finds "Persons born in Finland" may be used to restrict the subject entity of another extractor. The other extractor could then find a relation between "Persons born in Finland" and entities of type BUILDING, for example the relation "Buildings designed by persons from Finland".

Similar to the pattern panel, double-clicking is used to select or unselect type restrictions. Upon each interaction, the suggestions as well as the result panel are updated to reflect the current selection.

Result panel (3) The lower half of the screen is the result panel which lists a set of entity pairs that are found with the presently selected patterns and restrictions. Each entity pair is displayed along with the sentence that matches the pattern. By clicking the magnifying glass symbol next to an entity pair, more details are shown, including the entity pair's FREEBASE ids and a list of sentences that match the selected patterns.

Storing and exporting extractors After finishing building an extractor, users can export the setup as a JSON by clicking the download button in the lower right corner of the screen (see field **(5)** in Figure 2). This exports the selected patterns and restrictions, together with a result list of entity pairs found with the extractor. In addition, users can generate a "permalink" by clicking the button in the lower left corner of the screen (see field **(4)** in Figure 2). This allows users to generate links to created extractors and share them electronically.

3.2 Example Usage

We now briefly give an example of using the tool. Assume a user is interested in a relation between persons and the companies that they have founded.

There are several entry points the user may choose from. For instance, the user might search for appropriate entity types in the *X_Type* and *Y_Type* panels. Another option is to start by looking for appropriate patterns. For this, the user can use the search box in the pattern panel **(1)** to search for the general term "found". This results in a list of patterns being displayed, which includes the pattern `[X] found [Y]`. By single-clicking on it, the user can see a list of sentences that include this pattern. This is illustrated in field **(6)** in Figure 2.

With a double click on the pattern the user activates it and can now see the output of the extractor in the result panel **(3)** as well as patterns

and entity types that are suggested based on the current selection. Scanning through the result panel, the user finds that while many matching sentences do indeed express the desired relation (like "*Pierre Omidyar founded eBay*"), some others do not ("*Snape found Sirius Black*").

The tool however also presents three sets of suggestions that the user can use to refine the patterns. For instance, for both *X_Type* and *Y_Type* a ranked list of suggestions highlighted gray appears **(2)**. As illustrated in Figure 2, it suggests PERSON as *X_Type* and ORGANIZATION as *Y_Type*. The user can affirm suggestions by double clicking on them. When selecting ORGANIZATION as *Y_Type*, the result panel is updated to reflect the most recent changes. Scanning through the results the user sees that the extraction quality has greatly improved as there are far fewer false positives in the list.

The user may now try to further improve the extractor by selecting more specific patterns. The tool suggests the pattern `[X] be founder of [Y]`, which more accurately describes the relation the user wants to extract. Again by single-clicking on the suggestion, the user can see example sentences that match this pattern, as well as the selected entity type restrictions. Double-clicking on the pattern adds it to the extractor, which now consists of two patterns. With multiple patterns selected, the tool is now able to suggest patterns more accurately, offering patterns such as `[Y] founded by [X]`, `[X] start [Y]` and `[X] co-found [Y]`. By selecting them and implicitly rejecting those suggestions that do not reflect the desired relation (like the correlated patterns `[X] president of [Y]` or `[X] CEO of [Y]`), the user incrementally creates an extractor.

After multiple iterations of selecting suggested patterns and entity type restrictions the user is able to download the results of the extractor by using the download button (5) at the bottom of the page.

3.3 Implementation Details

We use CLUEWEB09 as corpus and make use of FACC1 annotations (Gabrilovich et al., 2013) to determine entity mentions and their FREEBASE types. We extract all English sentences that contain at least 2 FREEBASE entities, yielding over 160 million sentences. We then parse these sentences using the CLEARNLP pipeline (Choi and McCallum, 2013) and preemptively generate all subtrees for all entity pairs in all sentences. Together with information on the entity types, we store all information in a Lucene index for fast retrieval.

The web server handles the requests from the browser and returns information in the JSON data format. Saved extractors are stored using the Java database engine H2. We only materialize the selected patterns, not the extraction results. This way the saved extractors always stay valid, even if the underlying corpus is expanded or changed.

3.4 Hands-on Demonstration

We plan a hands-on demonstration in which users work with SCHNÄPPER to explore the CLUEWEB09 corpus for relations of interest. Our purpose is twofold: One the one hand we would like to make the case for the simplicity and intuitive nature of the proposed approach. One the other hand, we would like to gather feedback from the NLP community for possible future improvements to the approach. In particular some of the more advanced features such as embedding extractors within other extractors may be interesting to discuss in a hands-on demo.

4 Previous Work

Recent work in the field of rule-based RE has investigated workflows and tooling to facilitate the creation of extractors. (Li et al., 2012) presented a wizard-like approach to guide users in the process of building extractors. (Akbik et al., 2013) presented an example-driven workflow that allows even users who are unfamiliar with NLP to write extractors using lexico-syntactic patterns over dependency trees. Similarly, (Grishman and He, 2014) create a toolkit for persons who are experts in a *domain* of interest, but not in NLP. Users create extractors for pre-defined entities and relations by seeding example instances in a semi-supervised fashion. (Gupta and Manning, 2014) use a similar bootstrapping approach and create a tool for visualizing learned patterns for diagnostic purposes. Finally, (Freedman et al., 2011) focus on reducing effort in a user-driven process by including elements from active learning and bootstrapping, but target their tool at NLP experts.

Unlike the approach presented with this demo, these approaches are mostly intended for traditional RE in which relations of interest are spec-

ified in advance. With this demo, we instead support an *exploratory* workflow in which relations of interest may be discovered through user interactions with available data at little effort.

5 Outlook

While SCHNÄPPER is currently focused on binary relations only, we are investigating the application of comparable workflows at the entity level. Ideally, we would like to be able to create extractors that find named entities of custom types and embed them into custom relation extractors. While, as the demo shows, it is already possible to embed extractors into other extractors, more research is required fully develop the process of creating entity extractors, which possibly includes developing a different pattern language for the entity level. With more extensive capabilities of creating custom entity extractors, such tooling could conceivably be used to use the approach for knowledge base population tasks (Surdeanu and Ji, 2014). The approach could be also used to quickly create custom knowledge bases for specialized topics such as the biomedical domain (Hunter and Cohen, 2006). Another point of interest is that, since the tooling is Web-based, collaborative aspects of creating custom knowledge bases can be investigated in this context.

References

Alan Akbik, Oresti Konomi, and Michail Melnikov. 2013. Propminer: A workflow for interactive information extraction and exploration using dependency trees. In *ACL System Demonstrations*. Association for Computational Linguistics.

Alan Akbik, Thilo Michael, and Christoph Boden. 2014. Exploratory relation extraction in large text corpora. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 2087–2096.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Laura Chiticariu, Yunyao Li, and Frederick R Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 827–832.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366. International World Wide Web Conferences Steering Committee.

Marjorie Freedman, Lance Ramshaw, Elizabeth Boschee, Ryan Gabbard, Gary Kratkiewicz, Nicolas Ward, and Ralph Weischedel. 2011. Extreme extraction: machine reading in a week. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1437–1446. Association for Computational Linguistics.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Ralph Grishman and Yifan He. 2014. An information extraction customizer. In *Text, Speech and Dialogue*, pages 3–10. Springer.

Sonal Gupta and Christopher D Manning. 2014. Spied: Stanford pattern-based information extraction and diagnostics. *Sponsor: Idibon*, page 38.

Lawrence Hunter and K Bretonnel Cohen. 2006. Biomedical language processing: what's beyond pubmed? *Molecular cell*, 21(5):589–594.

Yunyao Li, Laura Chiticariu, Huahai Yang, Frederick R Reiss, and Arnaldo Carreno-fuentes. 2012. Wizie: a best practices guided development environment for information extraction. In *Proceedings of the ACL 2012 System Demonstrations*, pages 109–114. Association for Computational Linguistics.

Gary Marchionini. 2006. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46.

Friedrich Schmitthenner. 1837. *Kurzes deutsches Wörterbuch für Etymologie, Synonymik und Orthographie*. Jonghaus.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311. Association for Computational Linguistics.

Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324. Association for Computational Linguistics.

Mihai Surdeanu and Heng Ji. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proc. Text Analysis Conference (TAC2014)*.