Expert-Guided Entity Extraction using Expressive Rules

Mayank Kejriwal
kejriwal@isi.edu
USC Information Sciences Institute
Marina del Rey, California

Runqi Shao
runqisha@isi.edu
USC Information Sciences Institute
Marina del Rey, California

Pedro Szekely
pszekely@isi.edu
USC Information Sciences Institute
Marina del Rey, California

ABSTRACT

Knowledge Graph Construction (KGC) is an important problem that has many domain-specific applications, including semantic search and predictive analytics. As sophisticated KGC algorithms continue to be proposed, an important, neglected use case is to empower domain experts who do not have much technical background to construct high-fidelity, interpretable knowledge graphs. Such domain experts are a valuable source of input because of their (both formal and learned) knowledge of the domain. In this demonstration paper, we present a system that allows domain experts to construct knowledge graphs by writing sophisticated rule-based entity extractors with minimal training, using a GUI-based editor that offers a range of complex facilities.

ACM Reference Format:
Mayank Kejriwal, Runqi Shao, and Pedro Szekely. 2019. Expert-Guided Entity Extraction using Expressive Rules. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331392

1 INTRODUCTION

Semi-automatically building structured knowledge bases from raw Web (and also Natural Language) corpora has emerged as an important research area in multiple fields, including information retrieval, natural language processing (NLP), question answering, Web-scale fuzzy querying over open-world data and Semantic Web [16], [2], [3], [4]. In the NLP community, a knowledge *base* is generally defined as a set of triples of the form (h, r, t) where h and t are head and tail entities (resp.) and r is a relationship connecting two entities. In recent years, such bases have been increasingly visualized as *graphs*, in part due to the popularity of the Google Knowledge Graph [17], as well as advances in the Semantic Web community. In such a (multi-relational) graph-theoretic representation, head and tail entities are labeled nodes, and relationships are labeled, directed edges connecting the nodes. Based on the community,finer-grained definitions have also been proposed e.g., including provenance in reified statements.

In contrast with generic (i.e. open-world) KGs like Freebase, DBpedia or the Google Knowledge Graph, domain-specific KGs have also emerged as an important use case for domain experts who wish to gain insights from a domain-specific corpus. Unlike generic KGs, domain-specific KGs (from the perspective of either construction or downstream application) have not witnessed much research attention. Some recent Named Entity Recognition (NER) experiments that we conducted on a human-trafficking specific corpus of sentences that law enforcement was interested in analyzing showed that even standard tools do not perform well (Figure 1). Thus, domain-specific customization is necessary to extract 'common' fields like names and locations with high fidelity KG from raw sources.

Figure 1: Input sentences from the online sex advertisement domain. Named entities were incorrectly extracted by pre-trained modules from SpaCy due to the unusual nature of the domain.

One option for customizing the domain is that experts have to laboriously annotate large quantities of training data, following which a deep learning-based information extractor could be trained. Potentially, the process could be crowdsourced but, in addition to costing more, the assumption is that the data can be released publicly. This is not as common as one might suppose e.g., several domains that we have encountered had strict access and release requirements due to their sensitive nature.

A second option is for experts to look at examples from the domain, and specify sets of rules that are both interpretable and that experts can 'play' with. Since the expert usually has no programming expertise, such rules must be specifiable in an intuitive manner, and be amenable to example-based 'trial and error'. An added benefit arises when such rules can be combined with other extraction modules, such as a pre-trained machine learning-based named entity recognizer, or a Semantic Web resource like GeoNames[1].

In this paper, we describe a GUI-based rule specification system that allows users to write expressive and intuitive rule-sets, including rules not currently allowable in NLP packages like SpaCy

[1] http://www.geonames.org/

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331392

[1], without any knowledge of either programming or regular expressions. A visualization of the system dashboard is produced in Figure 2; we further describe scenarios and user experience in Section 3. To ensure fast, correct and scalable execution, the system compiles each rule in the rule-set into a set of SpaCy rules that can be executed in the backend to produce sets of extractions from raw data. Additionally, we have integrated the rule editor and compiler with the DIG architecture [11], [10], which permits (1) combining extractions from the system with extractions from more advanced modules like Conditional Random Fields, (2) recording extraction provenance, which allows users to view the impact of their specifications on overall quality. Concerning real-world usage, the system has already been used for fulfilling a variety of domain-specific KGC needs, including extracting attributes (e.g., phone numbers) in the online sex ad domain (to assist investigators track down leads), securities fraud, illegal weapons sales and 'ordinary' domains like museums.

2 RELATED WORK

With the increasing importance of structured data on the Web, Knowledge Graph Construction (KGC) has witnessed much research attention in the last decade, an industry-scale success story being the Google Knowledge Graph [17]. In the broader research literature, including natural language processing, KGC primarily includes information extraction (IE), but can also include other post-extraction steps such as knowledge graph identification and cleaning [14], [5]. For good surveys of IE, particularly Web IE, we point the reader to [16], [4], [2].

Major techniques for Web IE include both wrapper induction [12], [8], and *declarative* frameworks like regular expressions [6], [15], or natural language rules such as those that can be implemented and executed in a scalable fashion using NLP packages like SpaCy and Stanford NER [1], [7]. We also note that end-to-end KG construction and search technology that offers *front-end* interfaces are still rare [9], [13], and we are not aware of any cases where such a system has been successfully integrated into a broader search and analytics platform. The system presented herein offers a GUI both for non-programming domain experts to construct knowledge bases over large Web corpora, and also integrates with the Domain-specific Insight Graph (DIG) search engine to offer complex facilities over extractions (e.g., faceted search, dossier generation and fuzzy structured querying).

3 SYSTEM AND USER EXPERIENCE

The example in Figure 1, introduced earlier, illustrates one possible user experience whereby the user wants to extract 'common' named entities like names or locations, but is unable to rely on standard packages like SpaCy or Stanford NER due to the unusual nature of the domain (in this case, human trafficking). Instead, the user starts by looking at some example sentences, and forming rule templates using the 'template atoms' (also called token specifications) shown in Figure 2. For example, the topmost rule template, labeled self-explanatorily as 'my name is <proper-noun>' is composed of four atoms. Each atom is highly configurable, and can simply be constants (such as 'name'), or NLP artefacts like particles or adverbs, or regular expression artefacts like alphanumeric sequences (overlaid portion of Figure 2).

A second scenario arises when the attribute (and potentially, the domain) is uncommon in terms of publicly available tools being able to extract them from raw data (Figure 3). Because of the regular syntactic nature of attributes like telephone numbers or ages, as well as contextual regularities in the surrounding text, users are often able to get good coverage by specifying rules.

More specifically, a rule-based extractor for an entity consists of a collection of rules. Each rule targets extraction in a specific context with high precision. Multiple rules are used to extract an entity in multiple contexts. The extractions of a collection of rules are the union of the extractions of each rule, excluding extractions whose extent is completely contained within the extent of other extractions.

The input to the rule extractor is a sequence of tokens. Each rule consists of a sequence of token specifications. Each token specification defines a Boolean function $f(token)$. A rule is said to match a sequence of tokens if each token specification evaluates to true for each token in the sequence. The extraction of a rule on a matched sequence of tokens is the set of tokens matched by token specifications that have been marked as output tokens (orange border in Figure 2). By default the output is formatted as a string by concatenating the output tokens separated by space. Users can customize the output using a simple template language.

Token specifications can be marked as optional. When token specifications are optional, the rule editor behaves as if two rules had been defined, one with the optional token and one without the optional token. When a rule contains multiple optional tokens, the cross-product of all possible rules is considered. The extractions of rules with optional consist of the union of all maximal length rules in the cross-product.

The DIG rule editor is designed to extract entities defined using letters, numbers and symbols. Examples include entities such as person, organization and locations, extracted in traditional NER tools, as well as entities with specific syntax such as phone numbers and email addresses. The default DIG tokenizer defines any non-alphanumeric character as a separate token. Users can also customize the tokenizer. The five different token specifications currently supported by the system are enumerated in Table 1.

3.1 Implementation and Applications

The rule extractor is implemented as an extension to the SpaCy rule extractor, which offers similar, but simpler token specifications. Each DIG token specification is mapped to a set of SpaCy token specifications that collectively match a superset of the tokens that should be matched by the DIG token specification. Each DIG rule is implemented by the collection of SpaCy rules resulting from the creation of a cross-product of the SpaCy token specifications in the each collection. A post-processing step removes redundant matches for those cases when the collection of SpaCy token specifications matches a superset of the tokens that should be matched by the DIG token specification. For example, SpaCy does not support minimum and maximum limits for number tokens. In this case, the SpaCy rule matches arbitrary numbers, and the post-processing step removes

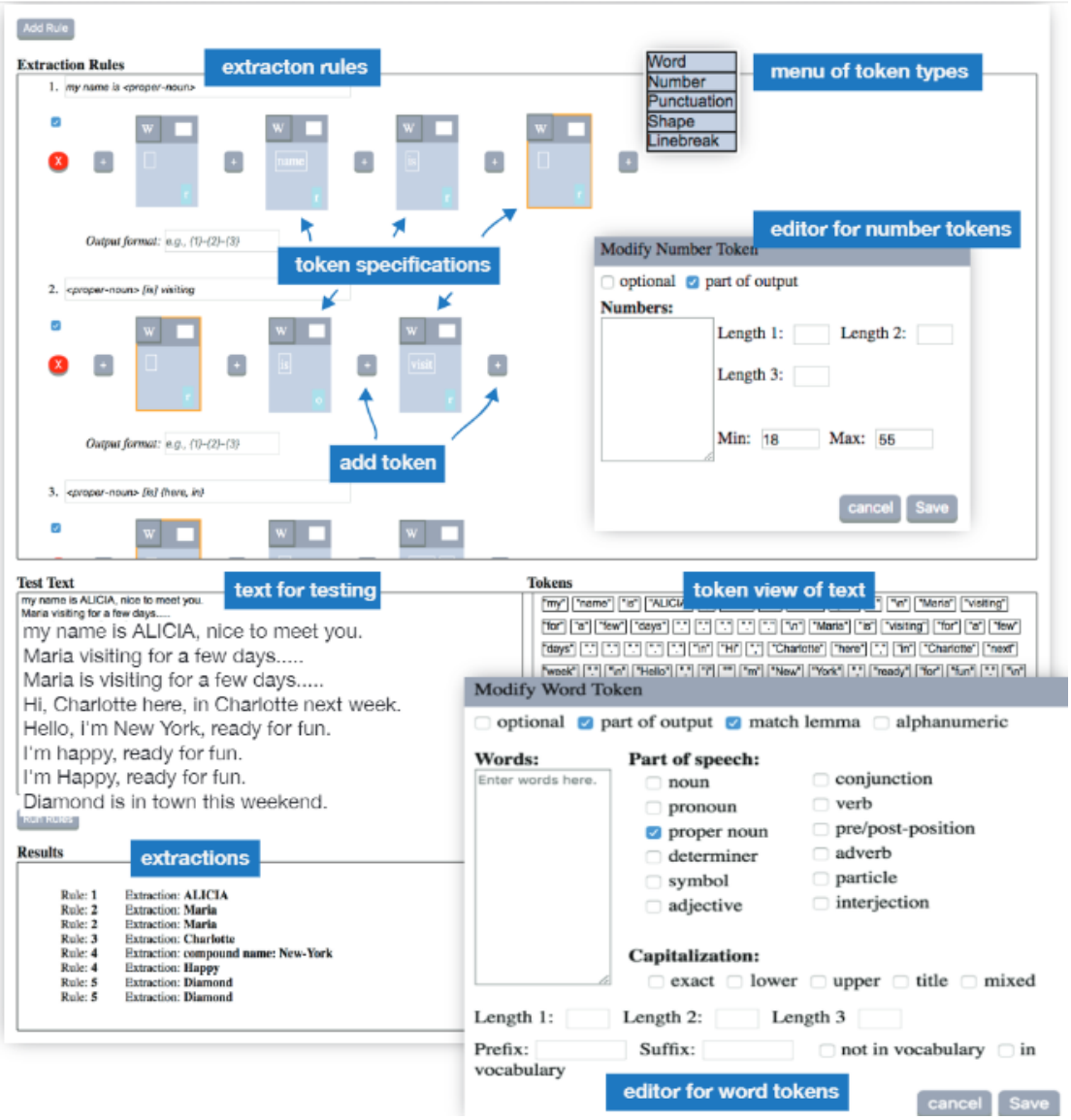

Figure 2: (Underlaid) The rule specification dashboard, with the components described in Section 3. (Overlaid) Configuration options for an atom that can be included in a larger rule template.

```
Hey im 21 f want
I am 18 years old or over.
22 years .............!!
23 yrs old and beautiful 😍
🌼🌼 Age 24 : 929-204-6145 - Massachusetts
Masseuse's Age: 33
```

Figure 3: Sentences containing uncommon (but useful to experts) attributes such as phone number and age that need to be extracted.

matches for numbers outside the desired range. A further post-processing step removes that are strict subsets of matches produced by other SpaCy rules in the collection of all SpaCy rules generated from a single DIG rule.

The software implementing the DIG rule editor is available on GitHub. The processor[2] and user interface[3] are both publicly available. The full system featured in this demonstration can also be downloaded[4]. All software carries an MIT license.

[2]https://github.com/usc-isi-i2/etk/blob/master/etk/spacy_extractors/customized_extractor.py
[3]https://github.com/usc-isi-i2/spacy-ui
[4]https://github.com/usc-isi-i2/dig-etl-engine

Table 1: Token specifications supported by the rule editor.

Token type	Description
Word tokens	match tokens consisting of a sequence of alphanumeric characters (see Figure 2). The word-token specification can be defined explicitly (by listing specific tokens) and implicitly (based on token characteristics) such as NLP features like POS tags and lemmatization, membership in a vocabulary e.g., English word, and syntactic features such as capitalization, length, prefix and suffix).
Number tokens	match tokens consisting of a sequence of digits (see Figure 2), and can also be defined explicitly by listing specific numbers to match, or implicitly based on range and number of digits.
Punctuation tokens	select a subset of punctuation symbols.
Shape tokens	match tokens consisting of a sequence of alphanumeric characters. Shapes can be defined explicitly using sequences of characters "X", "x" and "d", where "X" matches an upper-case letter, "x" matches a lower-case letter and "d" matches a digit. Shape tokens can also be defined implicitly using part of speech tags, a prefix and a suffix.
Line-break tokens	match tokens that break text into multiple lines and are defined by a minimum and maximum number of line-break characters.

Concerning real-world applications, the DIG rule editor has been used to define extractors for a variety of entities, including phone numbers for most countries in Europe and North and South America, email addresses, ages, person names, dates, massage parlor names, stock tickers, addresses, social media handles, artist names, as well as a variety of specific extractors for specific firearm classified ads and penny stock promotion web sites.

3.2 Demonstration Plan

In an anticipated live demonstration, we will showcase user experiences involving both unusual domains and attributes, using real-world data that was collected as part of the DARPA Memex program, which funded research into areas pertinent to domain-specific search, including crawling, domain specification, Web-scale domain discovery, human-centered search, information extraction, entity resolution and structured querying.

Additionally, the system described in this paper has already been integrated into the Domain-specific Insight Graphs (DIG) ecosystem, which allows domain experts and users to construct and search KGs from scratch, without any programming effort. We will also demonstrate the integration of the rule editor with the DIG system, and allow the user to search extractions from real-world data. The users will also get to explore the downstream effects (including provenance) of their rule specification efforts, including in advanced analytics offered by DIG such as dossier generation and facets.

4 CONCLUSION

Knowledge Graph Construction (KGC) from raw data is an important step in building end-to-end domain-specific search engines. Entity extraction is a key component of KGC and has received attention in the NLP community as Named Entity Recognition (NER). For unusual domains, or uncommon attributes, standard off-the-shelf NER systems do not perform well or require large quantities of training data to customize. In this paper, we presented a rule specification system that allows domain experts to specify rules-sets without any programming, and that can be visualized and explored in the DIG domain-specific search engine.

REFERENCES

[1] 2017. spaCy Natural Language Package. https://spacy.io/. Accessed: 2017-09-19.
[2] Charu C Aggarwal and ChengXiang Zhai. 2012. *Mining text data*. Springer Science & Business Media.
[3] Tim Berners-Lee, James Hendler, Ora Lassila, et al. 2001. The semantic web. *Scientific american* 284, 5 (2001), 28–37.
[4] Chia-Hui Chang, Mohammed Kayed, Moheb R Girgis, and Khaled F Shaalan. 2006. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering* 18, 10 (2006), 1411–1428.
[5] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial intelligence* 118, 1-2 (2000), 69–113.
[6] Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. 2016. A Relational Framework for Information Extraction. *ACM SIGMOD Record* 44, 4 (2016), 5–16.
[7] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 363–370.
[8] Sergio Flesca, Giuseppe Manco, Elio Masciari, Eugenio Rende, and Andrea Tagarelli. 2004. Web wrapper induction: a brief survey. *AI communications* 17, 2 (2004), 57–61.
[9] Maeda F. Hanafi, Azza Abouzied, Laura Chiticariu, and Yunyao Li. 2017. SEER: Auto-Generating Information Extraction Rules from User-Specified Examples. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6672–6682. https://doi.org/10.1145/3025453.3025540
[10] Rahul Kapoor, Mayank Kejriwal, and Pedro Szekely. 2017. Using contexts and constraints for improved geotagging of human trafficking webpages. In *Proceedings of the Fourth International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*. ACM, 3.
[11] Mayank Kejriwal and Pedro Szekely. 2017. Information Extraction in Illicit Web Domains. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 997–1006.
[12] Nicholas Kushmerick, Daniel S Weld, and Robert Doorenbos. 1997. Wrapper induction for information extraction. (1997).
[13] Yunyao Li, Elmer Kim, Marc A. Touchette, Ramiya Venkatachalam, and Hao Wang. 2015. VINERy: A Visual IDE for Information Extraction. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1948–1951. https://doi.org/10.14778/2824032.2824108
[14] Feng Niu, Ce Zhang, Christopher Ré, and Jude Shavlik. 2012. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *International Journal on Semantic Web and Information Systems (IJSWIS)* 8, 3 (2012), 42–73.
[15] Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. 2012. DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference. *VLDS* 12 (2012), 25–28.
[16] Sunita Sarawagi et al. 2008. Information extraction. *Foundations and Trends® in Databases* 1, 3 (2008), 261–377.
[17] Amit Singhal. 2012. Introducing the knowledge graph: things, not strings. *Official google blog* (2012).