Interactive Relation Extraction in Main Memory Database Systems

Rudolf Schneider **Cordula Guder** **Torsten Kilias** **Alexander Löser**
Jens Graupmann **Oleksandr Kozachuk**
Beuth University of Applied Sciences, Luxemburger Straße 10, 13353 Berlin, Germany
Exasol AG, Neumeyerstraße 22, 90411 Nürnberg, Germany
{ruschneider,s57515,tkilias,aloeser}@beuth-hochschule.de
{jens.graupmann, oleksandr.kozachuk}@exasol.com

Abstract

We present INDREX-MM, a main memory database system for interactively executing two interwoven tasks, declarative relation extraction from text and their exploitation with SQL. INDREX-MM simplifies these tasks for the user with powerful SQL extensions for gathering statistical semantics, for executing open information extraction and for integrating relation candidates with domain specific data. We demonstrate these functions on 800k documents from Reuters RCV1 with more than a billion linguistic annotations and report execution times in the order of seconds.

1 Introduction

Relation Extraction (RE) is the task of extracting semantic relations between two or more entities from text. Often these relations are loaded into a relational database system for further exploitation. One line of approaches to RE is rule-based, where users manually define rule-sets consisting of extraction patterns that if observed point to instances of a relation. These approaches are easy to debug, permit the user a high level of direct control over the extraction process and can outperform machine-learning based state-of-the-art models (Chiticariu et al., 2013). However, writing rules is a time consuming and iterative process, in particular for extracting uncommon relationship types with high recall and precision.

Our task: Complement existing in-house relational data with insights from text. While browsing news, a supply chain analyst performs research on suppliers of a car rental company, product recalls. She desires to complement an existing table *productrecall(supplier, product)*, with relations extracted from news text. Currently, the user performs these task with two separate systems, a system for extracting a relation *productrecall(supplier, product)*, such as (Krishnamurthy et al., 2008), and a relational database management system (RDBMS) for joining, grouping, aggregating and ordering. In a typical work flow, the user ships existing tables from the RDBMS to bootstrap text and ships back extracted relations to the RDBMS for analytical queries. This costly work flow is iterated until an analytical query shows desired results. Moreover, the user must learn to manage both systems.

Contribution. Ideally, users could execute both, analytical and relation extraction tasks, in a single database system and could leverage built-in query optimizations. Another crucial requirement is interactive query execution, in particular for extracting rare relation types with high recall and precision. We demonstrate INDREX-MM[1], a Main-Memory Relational Database System (MM-RDBMS) that permits this functionality, either as fast back-end for interactive relation extraction applications, such as (Michael and Akbik, 2015), or on command line. INDREX-MM provides a broad and powerful set of SQL-based query operators for declarative relation extraction. These include query predicates for detecting span proximity, predicates for testing overlapping spans or span containment, scalar functions for returning the context of a span, or user defined table generating functions for consolidating spans. Further, the system supports executing regular expressions and built-in operators from the RDBMS, such as joins,

[1] see our online demonstration at http://dbl43.beuth-hochschule.de/html/indrex-mm/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 103–106, Osaka, Japan, December 11-17 2016.

unions or aggregation functions. These additional operators permit the user basic operations for *looking up* words in sentences describing entities or other potential relation arguments. The system also supports the user *learning* about potential open relation candidates where these words appear in, or about distributions of potentials synonymous relation names. Finally, we support the user in *investigating* new relations. Our work in (Kilias et al., 2015) shows details and extensive performance evaluations. INDREX-MM bases on EXASOL, a parallel main-memory and column-oriented database. It permits integration via standard interfaces, such as JDBC, or business intelligence tools, like *Tableau*.

2 Demonstration Outline

We demonstrate how INDREX-MM supports the user in three elementary steps during the declarative relation extraction process, for which figure 1 gives a high-level overview. Each of these steps 'filters out' irrelevant sentences and only keeps sentences containing relations of the type *productrecall(supplier, product)*.

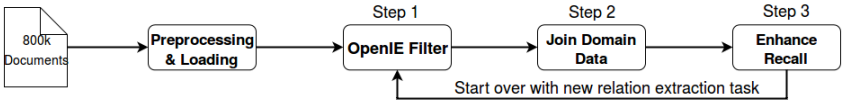


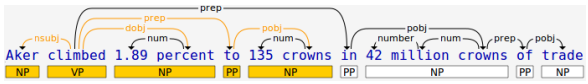
Figure 1: Relation Extraction process using Open Information Extraction in INDREX-MM.

Batch loading base annotations in a flat, sparse and cache affine data structure. Text mining workloads rarely require full scans of all table data, but do often require full scans of a small subset of the columns. Our base table layout from (Kilias et al., 2015) supports such work flows. This schema partitions data per *(document, span)*; we denote a span with its beginning and ending character. Many operations on text are 'local' on a single document. Hence, our partition scheme permits a MM-RDBMS to ship data for a single document 'close' to the CPU and in orders of magnitudes faster cache structures. For each span we provide additional attributes denoting annotation types, such as tokenization, sentence recognition, part-of-speech tagging, named entity recognition, user-defined types, dependency tagging, or noun- and verb-phrase chunking.[2] We add attributes for referencing spans to containment relations in the same document. For example, a span for a sentence may contain additional spans denoting organisations. Such a flat and sparse table layout pre-joins data already at data loading time and avoids most joins at query execution time. Because of the columnar layout in a MM-RDBMS, *NULL values* in attributes do not harm query execution time.

```
1 select NSUBJ_VALUE, VERB_VALUE, DOBJ_VALUE, PREP_TOKEN ...
2 FROM chunks_with_id c_nsubj, verb_chunks_with_id c_verb, ...
3 WHERE
4     <char_contains("c_nsubj.span1", "dep.d_nsubj")>
5     AND <char_contains("c_dobj.span1", "dep.d_dobj")>
6     AND <char_contains("c_pobj.span1", "dep.d_pobj")>
7     AND <char_same_region("t_prep.span1", "dep.d_prep")>
8     AND <char_same_region("t_verb.span1", "dep.d_verb")>
9     AND <char_contains("c_verb.span1", "t_verb.span1")>
10    AND <char_contains("s.span1", "t_verb.span1")>;
```

NSUBJ_VALUE	VERB_V...	DOBJ_VALUE	PREP_...	POBJ_VALUE
Aker	climbed	1.89 percent	to	135 crowns
Draper	beat	's Karol Kucera	in	straight sets
The Port Authority	approved	a series	including	a runway extension
Foreigners	bought	more U.S. securities	in	Q3

(a) Extractor query and result example.



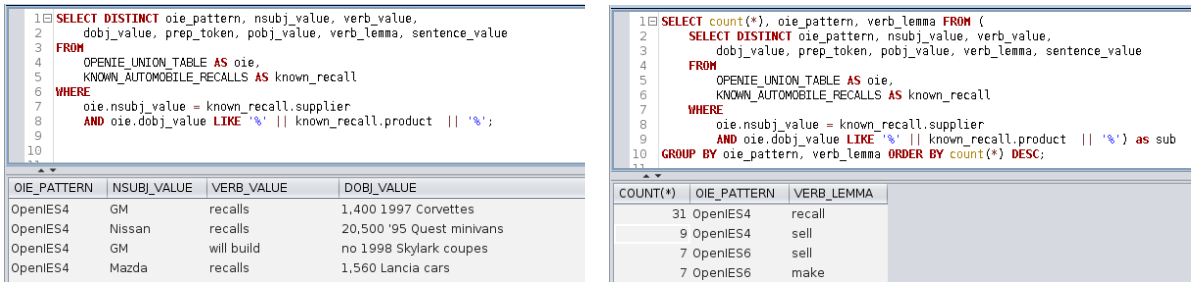
(b) Dependency parse and phrase chunks used in extractor query.

Figure 2: Query example of an Open Information Extraction pattern.

Step 1: Filtering relation candidates with Open Information Extraction. Open Information Extraction (OIE) is the task of extracting relations from large corpora and without requiring a pre-specified vocabulary. Relations are n-ary and arguments do not follow a pre-defined type set. From the perspective of a database, we understand OIE as selective filters connecting arguments in sentences. Recent work in clause-based OIE (Del Corro and Gemulla, 2013) shows effective filters for n-ary relations. INDREX-MM supports OIE as black box or as customizable and debuggable database views: One approach is

[2]We use Stanford CoreNLP 3.6 for this task.

executing OIE outside a MM-RDBMS as a black box, load results into an OIE table and reference spans to the annotation table. We noticed that such black boxes are difficult to debug, break with the programming paradigm of the database, and if the code does not match the corpus requirements of the user, she must wait for an update of the OIE system. Contrary, we provide the user in INDREX-MM a set of 'ready-to-use' OIE filters in SQL as views as shown in figure 2. The user can add SQL-predicates from additional OIE approaches, such as (Angeli et al., 2015), can debug directly on her corpus, while the MM-RDBMS takes over on optimizing the execution.



```
SELECT DISTINCT oie_pattern, nsubj_value, verb_value,
    dobj_value, prep_token, pobj_value, verb_lemma, sentence_value
FROM
    OPENIE_UNION_TABLE AS oie,
    KNOWN_AUTOMOBILE_RECALLS AS known_recall
WHERE
    oie.nsubj_value = known_recall.supplier
    AND oie.dobj_value LIKE '%' || known_recall.product  || '%';
```

OIE_PATTERN	NSUBJ_VALUE	VERB_VALUE	DOBJ_VALUE
OpenIES4	GM	recalls	1,400 1997 Corvettes
OpenIES4	Nissan	recalls	20,500 '95 Quest minivans
OpenIES4	GM	will build	no 1998 Skylark coupes
OpenIES4	Mazda	recalls	1,560 Lancia cars

(a) Join of the Union OIE table with in house data regarding known product recalls of the company's suppliers.

```
SELECT count(*), oie_pattern, verb_lemma FROM (
    SELECT DISTINCT oie_pattern, nsubj_value, verb_value,
        dobj_value, prep_token, pobj_value, verb_lemma, sentence_value
    FROM
        OPENIE_UNION_TABLE AS oie,
        KNOWN_AUTOMOBILE_RECALLS AS known_recall
    WHERE
        oie.nsubj_value = known_recall.supplier
        AND oie.dobj_value LIKE '%' || known_recall.product  || '%') as sub
GROUP BY oie_pattern, verb_lemma ORDER BY count(*) DESC;
```

COUNT(*)	OIE_PATTERN	VERB_LEMMA
31	OpenIES4	recall
9	OpenIES4	sell
7	OpenIES6	sell
7	OpenIES6	make

(b) Relation candidates grouped, counted and ordered by pattern and verb. The most frequent combination is OIE-Pattern 4 and the verb "recall".

Figure 3: Use of in-house data to spot patterns of product recall mentions in the OIE schema.

Step 2: Joining OIE relations with domain data into a universal schema and spotting patterns. After step 1 relations connect two or more relation arguments. However, we need to filter out irrelevant relations and only keep relations that belong to our desired relation type *productrecall*. For example, we keep relations connecting a company with predicates, such as 'recalls', 'withdraws' and discard relations with 'sold' or 'has refused'. For executing this task and analogue to universal schemas (Riedel et al., 2013), we join arguments of OIE-relations with in-house domain specific relations representing the same semantic type, such as a table describing product recalls of the suppliers of a company. As a result, our universal schema represents relations, mainly candidate patterns of our desired relation type, and few patterns for other semantic types (see figure 3a). The fast execution performance of INDREX-MM permits the user to filter out these irrelevant patterns manually. For example, she aggregates, groups and counts patterns with standard SQL, orders patterns by frequency and marks unsuitable patterns (see figure 3b). For spotting additional semantic patterns, we provide synonyms from Wordnet. INDREX-MM also supports loading existing lexical patterns from the literature in a table, such as Hearst patterns (Hearst, 1992) or patterns from ConceptNet 5[3]. The user can execute a join and utilize these patterns as additional filters for OIE candidates. Focus of our current research is applying in-database-analytics for pattern generation, such as clustering techniques (see our work in (Akbik et al., 2012)).

Step 3: Applying selectional restriction and enhancing recall. For further enhancing recall, the user keeps lexical patterns for predicates from the last step but applies various selectional restrictions to arguments. INDREX-MM supports selectional restrictions to one or many argument types. For example, the user may keep the company name of relations from step 2, but relaxes the second argument. As a result, she may spot new relations of *productrecall(supplier, product)*, in particular relations between previously known companies and previously unknown products.

3 Discussion

Execution on one billion annotations in seconds. We measure the relation extraction process from above in INDREX-MM on Reuters RCV1 with 800k documents and 1.2 billion annotations. For each of the four steps mentioned above we measure the execution time and how selective each filtering step prunes sentences. For evaluating precision, we asked two independent students to draw a sample of 100 sentences randomly after each step and to count the number of correct relations for our desired type.

[3] $http://conceptnet5.media.mit.edu$

Step	Time	Relations	RL100	Examples
BL	180 min	15.785.155	0	-
1 OIE	9,9s	13.695.006	10	All OIE pattern (Mitsubishi, raised its production plan, October)
2 PR	49ms	134	31	**Product recall**(GM, recalls, 1,400 1997 Corvettes)
3 PR	619ms	921	61	**Product recall**(Tensor, recalls, halogen bulbs)
2 AL	16.64s	662	35	**Alliance**(LUKoil, signed, a $2-billion deal, with SOCAR)
3 AL	2.505s	3.265	91	**Alliance**(Xillix, signed, an agreement, with Olympus)
2 AC	5.643s	112	41	**Acquisition**(Quaker, reviews, Snapple)
3 AC	7.031s	1654	73	**Acquisition**(Quaker, acquired, Snapple, for, $1.8 billion)

Table 1: Performance for each step. After phase BL, we loaded 15.7 Mio sentences and estimate one relation per sentence. In step 1, we extract OIE relations from sentences using the 7 basic patterns from ClausIE resulting in slightly fewer OIE relations than sentences. For phase 2 and 3 we show results for *productrecall(supplier, product)*, *alliance(company, company)* and *acquisition(company, company)*. We count correct relations on a randomly taken sample of 100 sentences (RL100).

Table 1 shows our measurements and example sentences. One-time batch loading (denoted with *BL* in Table 1) takes roughly 180 minutes, because the MM-RDBMS executes compressions and builds index structures before we can run queries. In a streaming scenario the MM-RDBMS uses delta indexing techniques and permits hitting queries while new data is inserted.

INDREX-MM exploits data locality and leverages multi-core shared memory architectures. Declarative relation extraction systems, such as SystemT (Krishnamurthy et al., 2008) or GATE[4], need to conduct expensive data shipping between different NLP components and databases. Such data shipping is a major performance bottleneck. Contrary, INDREX-MM avoids data shipping, rather ships functionality to data, and even leverages multiple built-in optimizations of main memory RDBMSs, such as massive parallel execution with multi-cores, compression techniques and columnar based table layouts, cache affine data structures, single instruction multiple data (SIMD) or result materializations.

Acknowledgements Our work is funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) under grant agreement 01MD16011E (Project: Medical Allround-Care Service Solutions).

References

Alan Akbik, Larysa Visengeriyeva, Priska Herger, Holmer Hemsen, and Alexander Löser. 2012. Unsupervised discovery of relations and discriminative extraction patterns. In *COLING*, pages 17–32.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*, pages 344–354.

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *EMNLP 2013*, pages 827–832.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *World Wide Web*, pages 355–366.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545.

Torsten Kilias, Alexander Löser, and Periklis Andritsos. 2015. INDREX: In-Database Relation Extraction. *Information Systems*, 53:124–144.

Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2008. Systemt: a system for declarative information extraction. *SIGMOD Record*, 37(4):7–13.

Thilo Michael and Alan Akbik. 2015. SCHNAPPER: A web toolkit for exploratory relation extraction. In *ACL, System Demonstrations*, pages 67–72.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*, pages 74–84.

[4] $https://gate.ac.uk/ie/$