Exploratory Relation Extraction in Large Text Corpora

Alan Akbik **Thilo Michael** **Christoph Boden**
Database Systems and Information Management Group
Einsteinufer 17, 10587 Berlin, Germany
{firstname.lastname}@tu-berlin.de

Abstract

In this paper, we propose and demonstrate *Exploratory Relation Extraction* (ERE), a novel approach to identifying and extracting relations from large text corpora based on user-driven and data-guided incremental exploration. We draw upon ideas from the information seeking paradigm of Exploratory Search (ES) to enable an exploration process in which users begin with a vaguely defined information need and progressively sharpen their definition of extraction tasks as they identify relations of interest in the underlying data. This process extends the application of Relation Extraction to use cases characterized by imprecise information needs and uncertainty regarding the information content of available data.

We present an interactive workflow that allows users to build extractors based on entity types and human-readable extraction patterns derived from subtrees in dependency trees. In order to evaluate the viability of our approach on large text corpora, we conduct experiments on a dataset of over 160 million sentences with mentions of over 6 million FREEBASE entities extracted from the CLUEWEB09 corpus. Our experiments indicate that even non-expert users can intuitively use our approach to identify relations and create high precision extractors with minimal effort.

1 Introduction

1.1 Motivation and Problem Statement

Relation Extraction (RE) is the task of creating extractors that automatically find instances of semantic relations in unstructured data such as natural language text (Riloff, 1996). An example extraction task might be to find instances of the EDUCATEDAT relation, which relates persons to their educational institution and may include the entity pair <*Sigmund Freud, University of Vienna*> as relation instance. Motivated by an explosion of readily available sources of text data such as the Web, RE offers intriguing possibilities for querying and analyzing data as well as extracting and organizing the contained information (Sarawagi, 2008). As scalable computing architectures capable of processing ever larger amounts of data are being developed (Dean and Ghemawat, 2004) and dependency parsers are becoming more accurate and more robust (Petrov and McDonald, 2012), so rises the potential of developing means to directly access the structured information contained in natural language text.

In spite of such positive trends however, currently established methods of creating relation extractors suffer from a number of limitations. The first is one of *cost*; the process of creating extractors requires either labeled data to be produced at sufficient quality and quantity in order to train a supervised machine learning algorithm (Culotta and Sorensen, 2004; Mintz et al., 2009), or the manual creation of a complex set of extraction rules (Strötgen and Gertz, 2010; Reiss et al., 2008). In either case, the process is tedious and time-consuming and requires trained specialists with an extensive background in NLP, rule-writing or machine learning (Chiticariu et al., 2013). Worse, this process needs to be repeated for every relation and domain of interest. Due to this cost, great care must be taken when deciding which relation types to look for in a given text corpus.

This leads to the second limitation, namely the necessary *a priori specification* of relations. Current methods generally require a careful upfront definition of the RE tasks in order to start producing labeled training data or extraction rule-sets. Practical scenarios, however, are often characterized by imprecise and rapidly changing information needs and uncertainty regarding the type of information contained in large, given text corpora (Chiticariu et al., 2013). This severely limits the practicability of currently established RE methods.

1.2 Exploratory Search for Relations

To address these limitations, we propose a process of *exploration* for relations of interest in available data. We propose to substantially reduce entry barriers into RE so that extraction tasks no longer need to be exactly pre-specified and expensively prepared by generating labeled training data in advance. Instead, we propose a manual, rule-based approach in which extraction rules are kept very simple so that users can formulate natural language-like patterns as exploratory queries for relations against a text corpus.

We draw inspiration from the information seeking paradigm of *Exploratory Search (ES)* (Marchionini, 2006; White and Roth, 2009), where users start with a vaguely defined information need and - with a mix of look-up, browsing, analysis and exploration - progressively discover information available to address it and simultaneously concretize their information need. One of the challenges associated with the often desired capability of ES is the design of interactive interfaces to support users as they navigate through complex environments. Similarly, our challenge is to create an intuitive workflow that allows non-experts in NLP to engage in relation exploration.

We propose to simplify the search for information by using natural language-like queries that match subtrees in large corpora of dependency parsed data while hiding the complexity from the users. Explorative queries return matching relation instances and source sentences, as well as suggestions for further queries computed from the available data. By following a process of experimental querying and accepting or rejecting pattern suggestions, users identify relations of interest and group patterns into extractors. Our goal is to make use of such data-guidance to facilitate exploration while giving as much explicit control to a user as possible.

1.3 Contributions

In this paper, we propose and demonstrate *Exploratory Relation Extraction (ERE)*, a user-driven and data-guided incremental exploration approach to Relation Extraction. We give details on our relation extraction pattern language and introduce a guided, interactive workflow aimed at allowing users to explore parsed text corpora for relations at minimal effort. We conduct two experiments on a large corpus of over 160 million sentences from the CLUEWEB09 to determine in how far non-experts can use ERE to discover and extract relations. We discuss the results of the user study, as well as strengths and weaknesses of our proposed approach.

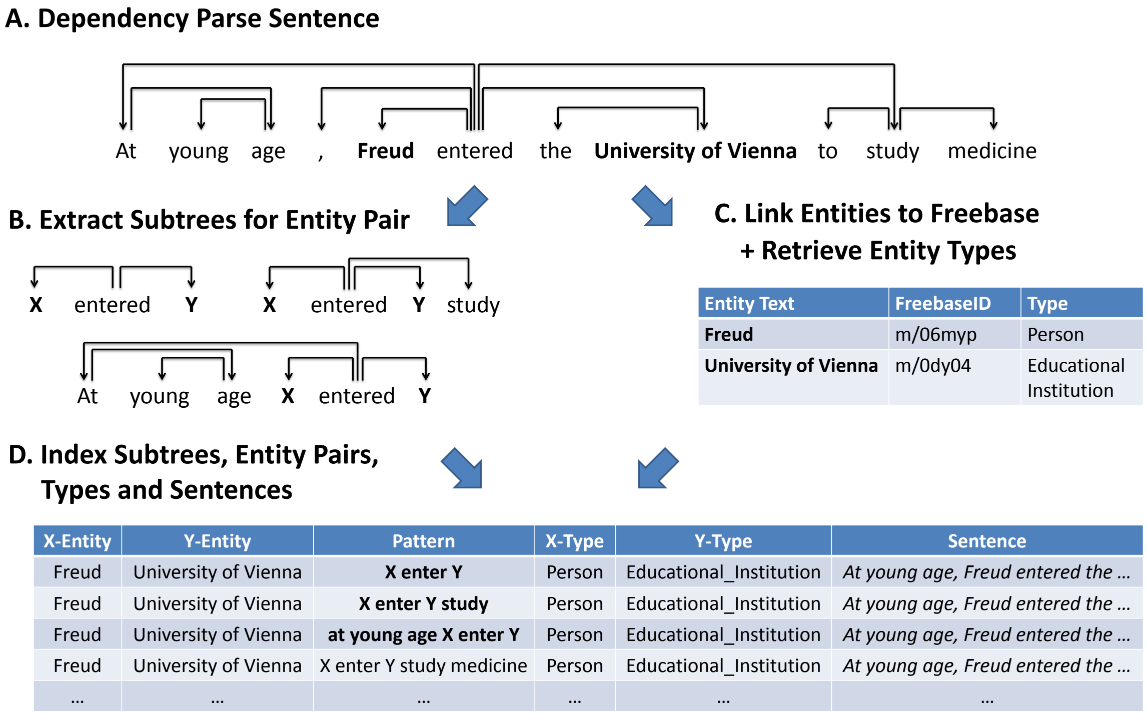
2 Exploratory Relation Extraction

In this section, we present our approach for Exploratory Relation Extraction. We provide details on how we define extraction patterns and how we preemptively extract all subtrees in dependency trees from a given text corpus (Section 2.1). We then outline a data-guided incremental workflow to explore the indexed data for relations (Section 2.2) and illustrate this with an exemplary execution (Section 2.3).

2.1 Human-Readable Relation Extraction Patterns

Like much previous work in RE (Culotta and Sorensen, 2004; Schutz and Buitelaar, 2005; Uszkoreit, 2011), we define extraction patterns using features from dependency-parsed sentences. As recent work has shown (Del Corro and Gemulla, 2013; Akbik et al., 2013b), patterns in dependency trees are well-suited to manual rule based RE, as they enable more succinct and thus more human-readable rule sets. Following this work, we define RE patterns as subtrees in dependency trees.

In our work, we follow the idea of Preemptive Information Extraction (Shinyama and Sekine, 2006) in which all possible relations for a given text corpus are preemptively generated in advance. Applied


A. Dependency Parse Sentence
At young age , Freud entered the University of Vienna to study medicine
B. Extract Subtrees for Entity Pair
X entered Y
X entered Y study
At young age X entered Y
C. Link Entities to Freebase + Retrieve Entity Types
D. Index Subtrees, Entity Pairs, Types and Sentences

Entity Text	FreebaseID	Type
Freud	m/06myp	Person
University of Vienna	m/0dy04	Educational Institution

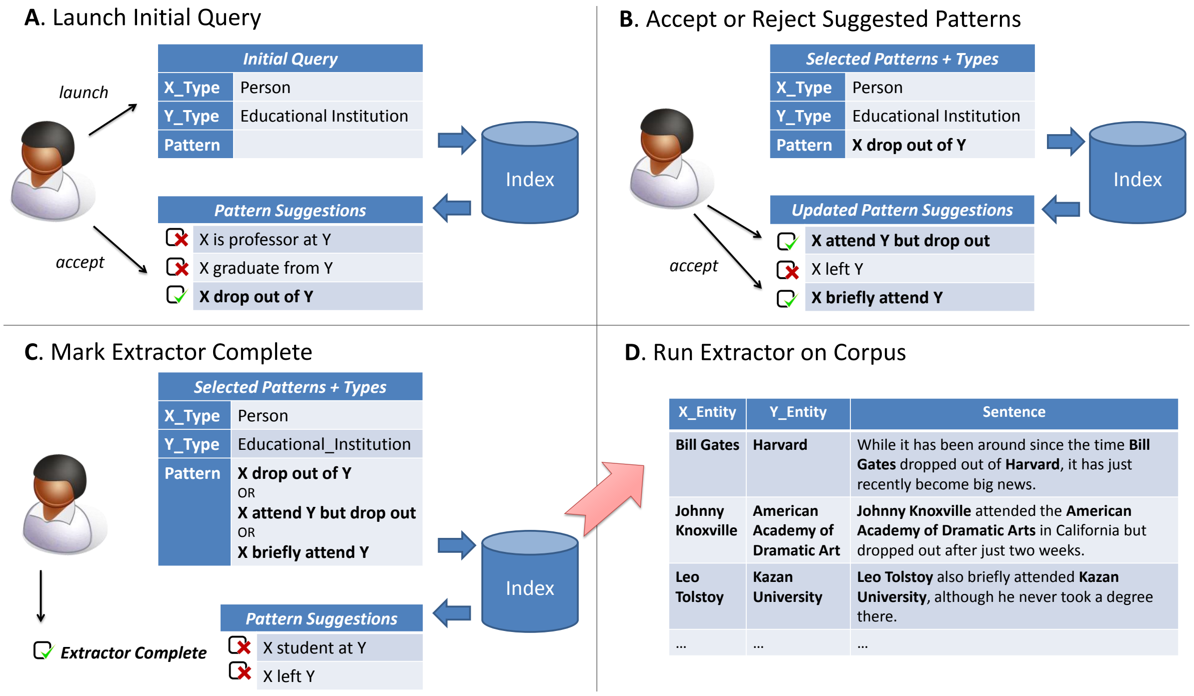
X-Entity	Y-Entity	Pattern	X-Type	Y-Type	Sentence
Freud	University of Vienna	**X enter Y**	Person	Educational_Institution	*At young age, Freud entered the ...*
Freud	University of Vienna	**X enter Y study**	Person	Educational_Institution	*At young age, Freud entered the ...*
Freud	University of Vienna	**at young age X enter Y**	Person	Educational_Institution	*At young age, Freud entered the ...*
Freud	University of Vienna	X enter Y study medicine	Person	Educational_Institution	*At young age, Freud entered the ...*
...	...	...	...	...	...

Figure 1: Illustration of the subtree generation process. We parse each sentence in a given document collection using a dependency parser and annotate all entities (**A**). Then, we generate all possible subtrees in the dependency tree that span pairs of annotated entities, three of which are illustrated in **(B)**, and link entities to their FREEBASE IDs to determine their entity types (**C**). We then generate a lexical, lemmatized representation of these subtrees which we store along with the entity pair, their entity types and sentence they are observed with **(D)**.

to our problem this means that we generate all possible dependency subtrees, arguing that depending on the user's information need, any such pattern may be valuable. Since we are interested in binary relations only, we generate only those subtrees that span two named entities in a sentence. In addition, we also determine the fine-grained entity types for named entities in order to allow users to optionally restrict patterns to match only entities of certain types. Previous work has shown the benefit of including fine-grained type restrictions into patterns (Akbik et al., 2013a).

We illustrate this process with an example sentence in Figure 1, for which we determine all subtrees that span the indicated entity pair. In the subtrees, we replace the entity tokens with the placeholders "*X*" and "*Y*", where the former is the placeholder for the X-entity and the latter the placeholder for the Y-entity. For better human-readability, we lexicalize the patterns by lemmatizing the words and discarding information on typed dependencies. We also link the entities in the sentence to entries in the FREEBASE knowledge base (Bollacker et al., 2008), allowing us to retrieve their fine grained entity types.

We then index the information on lexicalized patterns, the entities they span and their types, as well as the sentences in which the patterns were found (Figure 1D). This allows users to query for any combinations of patterns and entity type restrictions and retrieve matching entity pairs and sentences from the index. For instance, a user may query for all entity pairs that match the "*at young age X enter Y*" pattern, and optionally restrict the Y-entity to be only of type ORGANIZATION, or more specific types such as CHURCH or UNIVERSITY. We argue that because patterns are lexicalized variants of dependency subtrees and entity type restrictions can have human readable names, such queries are intuitive to users even without an NLP background. The use and preemptive indexing of human-readable patterns decreases the entry barriers into the ERE process, as this enables users to exploratively query parsed text corpora.


A. Launch Initial Query
launch
accept
Index
B. Accept or Reject Suggested Patterns
accept
Index
C. Mark Extractor Complete
Index
Extractor Complete
D. Run Extractor on Corpus

Initial Query	
X_Type	Person
Y_Type	Educational Institution
Pattern	

Pattern Suggestions
☒ X is professor at Y
☒ X graduate from Y
☑ **X drop out of Y**

Selected Patterns + Types	
X_Type	Person
Y_Type	Educational Institution
Pattern	**X drop out of Y**

Updated Pattern Suggestions
☑ **X attend Y but drop out**
☒ X left Y
☑ **X briefly attend Y**

Selected Patterns + Types	
X_Type	Person
Y_Type	Educational_Institution
Pattern	**X drop out of Y** OR **X attend Y but drop out** OR **X briefly attend Y**

Pattern Suggestions
☒ X student at Y
☒ X left Y

X_Entity	Y_Entity	Sentence
Bill Gates	Harvard	While it has been around since the time **Bill Gates** dropped out of **Harvard**, it has just recently become big news.
Johnny Knoxville	American Academy of Dramatic Art	**Johnny Knoxville** attended the **American Academy of Dramatic Arts** in California but dropped out after just two weeks.
Leo Tolstoy	Kazan University	**Leo Tolstoy** also briefly attended **Kazan University**, although he never took a degree there.
...	...	...

Figure 2: Illustration of the exploratory relation extraction process. The user begins with specifying entity types of interest and receives a set of pattern suggestions (**A**). Intrigued by the pattern "X drop out from Y", the user affirms this pattern. This prompts updated pattern suggestions which the user affirms or rejects (**B**). When no more interesting patters are offered, the user marks the extractor as complete (**C**) and runs it on a corpus, retrieving relation instances and matching sentences (**D**).

2.2 Guidance From Available Data

A second key component is to provide guidance in the exploration process by computing suggestions for patterns from user input and enabling an interactive workflow that allows users to work with available data. Such guidance is needed for two reasons: First, though much effort is invested in human-readable extraction patterns, users may need support in formulating patterns and choosing entity type restrictions. This is especially the case when users are non-experts in the domain of interest and they strive to identify a range of appropriate patterns. Second, users may be uncertain of the information content of a given text corpus. By providing guidance through automatic pattern suggestions that reflect available information, we help users find patterns for their information need.

Users formulate an *entry point* to launch the exploration process, either by providing entity types, patterns or both. We guide the formulation of this initial query through autocomplete options. If the user enters only types for the entities, the system offers the most common patterns that are observed between entities of these types. The user can also search for patterns that contain a certain keyword.

In either case, the system suggests patterns that meet the user-defined entry point. Patterns are ordered by their absolute count in the corpus so that more common patterns are displayed at the top of the list. In addition, verb-based patterns are favored using a scoring metric that assigns extra points to patterns that include verbs. To assist a user in understanding a pattern, we optionally display example sentences and entity pairs in which it matches.

The user then starts a process of selecting (and de-selecting) entity type restrictions and pattern, thus refining the extractor while being guided by constantly updated pattern suggestions. The user continues this process until satisfied with the created extractor at which point it can be saved and the discovered relation instances downloaded. The user can now repeat the workflow to create more extractors.

2.3 Exploration Workflow Example

Suppose we have a user who is given a large text corpus and is asked to link persons to their respective educational institutions, but is unsure of what type of relevant information may be found in the corpus. Knowing only that relations should hold between entities of type PERSON and entities of type EDUCATIONAL_INSTITUTION, the user starts an exploration process by providing only these entity type restrictions. This is illustrated in Figure 2A).

A query is run against the index that identifies common patterns that hold between entities of such types, including "*X be professor at Y*", "*X study at Y*" and "*X drop out from Y*". Recall that each pattern is a human-readable version of a subtree in a dependency tree with two placeholders for entities, namely "X" and "Y". These placeholders may match named entities of any type, or can be restricted to matching only entities of certain types such as persons, organizations or locations. By clicking on a pattern, the user retrieves entity pairs and sentences in which a pattern matches; For example, the user is informed that the pattern "*X study at Y*" finds the relation instance <*Bill Gates, Harvard University*> in the sentence "***Bill Gates** briefly studied at **Harvard University***.".

Intrigued by the pattern "*X drop out from Y*", the user affirms this pattern and rejects all other suggestions. This causes a new query to be run against the parsed data, this time consisting of the entity restrictions as well as the pattern. As the query is now more concrete, the pattern suggestions are updated to reflect this new information. The user is presented with similar patterns such as "*Y dropout X*" and "*X attend Y but drop out*". This is illustrated in Figure 2B).

The user repeats this, selecting or de-selecting patterns (Figure 2B). At each interaction, suggestions are updated to reflect the current selection. When the user is satisfied with the identified relation, the selected set of patterns and restrictions is saved as an extractor (Figure 2C) and executed against the entire text corpus (Figure 2D). This returns lists of matching relation instances and sentences. The user has thus started with an imprecise information need and identified a relation of interest in a given text corpus, namely a relation for persons that attended an educational institution but did not graduate.

3 Experiments

In order to examine in how far our approach indeed contributes to overcoming the limitations of RE outlined in Section 1.1, namely the significant cost and the necessary a-priory specification of relations, we conduct a user study with 10 subjects that have little or no NLP background. We ask the users to apply the workflow for two separate tasks: An *extraction task* in which users are given four clearly defined semantic relations and an *exploration task* in which users are asked to identify relations for more vaguely defined information needs. We only provide the users with a brief introduction into the workflow. For the extraction task, we measure the time spent per extractor and estimate the quality of the created extractors in terms of precision and recall. For the exploration as well as for the extraction task we also qualitatively inspect discovered relations and evaluate user feedback.

3.1 Datasets

ClueWeb09. As source of text data, we use the English language portion of the well-known CLUEWEB09[1] reference corpus, consisting of roughly 5 billion crawled Web pages. We use boilerplating to remove HTML markup and sentence splitting to determine English language sentences.

FACC1. We use the recently released FACC1 (Gabrilovich et al., 2013) resource, a high quality named entity linking effort that was executed on the CLUEWEB09 corpus, linking over 6 billion entity mentions to their corresponding FREEBASE entries. Using this data, we identify over 160 million sentences in CLUEWEB09 that contain at least two entities we can link to FREEBASE. We parse all such sentences using the ClearNLP toolkit (Choi and McCallum, 2013).

Gold Standard Relation Annotations. As gold standard, we use the FREEBASE relation annotations as well as annotations from the "*Relation Extraction Corpus*"[2] a large, human-judged dataset of five relations about public figures on Wikipedia that was released by Google. Four of these relations involve

[1] http://lemurproject.org/clueweb09/

[2] http://code.google.com/p/relation-extraction-corpus/

	EducatedAt					**GraduatedWithDegree**				
	#Inst	P	R	#Pat	Time	#Inst	P	R	#Pat	Time
User 1	**58,611**	0.99	0.2	**51**	12 min	17,698	1.0	0.27	34	17 min
User 2	48,782	0.99	**0.31**	34	15 min	12,180	1.0	0.27	27	14 min
User 3	25,435	0.88	0.12	12	8 min	**54,371**	0.93	0.53	24	8 min
User 4	33,095	0.99	0.23	25	12 min	7,196	1.0	0.22	9	10 min
User 5	47,668	0.76	0.16	29	13 min	34,942	1.0	0.48	3	5 min
User 6	20,356	0.99	0.15	18	14 min	10,290	1.0	0.25	12	14 min
User 7	22,889	0.62	0.01	8	4 min	37,119	0.71	0.6	19	4 min
User 8	31,412	0.98	0.19	13	15 min	1,251	0.46	0.04	10	14 min
User 9	14,169	0.99	0.1	6	8 min	13,104	0.6	0.17	13	12 min
User 10	29,289	0.99	0.19	17	15 min	35	1.0	0.02	4	20 min
Average	33,171	0.92	0.17	21	11.6 min	18,819	0.87	0.29	16	11.8 min

	BornIn					**DiedIn**				
	#Inst	P	R	#Pat	Time	#Inst	P	R	#Pat	Time
User 1	**158,222**	0.7	0.26	18	9 min	**25,779**	0.7	0.14	32	9 min
User 2	72,888	0.79	0.21	23	17 min	13,582	0.86	0.13	12	12 min
User 3	89,825	0.84	0.22	21	7 min	15,849	0.86	0.13	12	7 min
User 4	66,899	0.81	0.21	19	14 min	13,542	0.86	0.13	11	8 min
User 5	65,213	0.82	0.19	19	15 min	21,105	0.85	0.13	10	9 min
User 6	131,275	0.83	0.25	16	13 min	14,423	0.85	0.13	8	9 min
User 7	7,851	0.85	0.03	5	4 min	15,980	0.85	0.14	17	4 min
User 8	52,927	0.82	0.17	10	15 min	25,090	0.74	0.14	8	14 min
User 9	56,724	0.84	0.18	10	12 min	15,728	0.85	0.14	8	9 min
User 10	58,347	0.94	0.22	10	15 min	14,112	0.86	0.13	8	10 min
Average	76,017	0.82	0.19	15	12.1 min	33,171	0.82	0.13	13	9.1 min

Table 1: Evaluation results for the 4 well-defined relations in the extraction task. We note differences from user to user, especially with regards to the number of found instances (**#Inst**), the number of selected patterns (**#Pat**) and the time spent per relation. Extractors generally find large amounts of relation instances at high precision (**P**), while recall values (**R**) are lower. Users are ordered by the total number of patterns they selected. User 1 selected the most patterns overall and found the most instances for the BornIn, DiedIn and EducatedAt relations (highlighted bold). User 10 both spent the most time overall while selecting the fewest patterns. User 7 spent the least amount of time overall.

Freebase entities, namely BornIn, DiedIn, EducatedAt and GraduatedWithDegree. We use these relations in the extraction task.

3.2 Extraction Task

We evaluated the user-created extractors against the gold standard annotations. However, even with relatively large sources of annotations, only roughly 5% of entity pairs in our 160 million sentences have a known Freebase relation. We therefore compute precision and recall only for labeled entity pairs, and separately list the absolute number of extracted relation instances.

Large amounts of relation instances at high precision. As Table 1 indicates, many users were able to create extractors that find very large amounts of instances (over 100.000 instances in some cases) at high precision in an average time of 9 to 12 minutes, while recall values tend to be lower. This tendency to favor precision at the cost of recall has been observed in previous works on rule-based RE (Wang et al., 2012). Nevertheless, we analyzed precision and recall in greater detail by manually evaluating a sample of 200 false positives and 200 false negatives by hand to discover the reasons for precision and recall loss.

Mismatch between gold standard and results. As Table 2 shows, false positives are most commonly due to inconsistencies between extraction results and the gold standard annotations concerning the level of granularity of a relation instance. For example, we found BORNIN and DIEDIN relation instances that indicated a person's place of birth or death at lower or higher granularity than FREEBASE records. An example of this is given in Table 2 for Abraham Lincoln's place of death; we find the more granular <Lincoln, Hildene>, while the gold standard expects <Lincoln, Vermont>. While different from the gold standard, such instances are not false, which suggests that actual precision may be higher than the measured values indicate.

Missed patterns and entity types. The most common causes of recall loss are patterns that users failed to select. In Table 2, we distinguish between "common" patterns that were found by at least one user and "long tail" patterns that were found by none. While we did not expect a user-driven approach to identify long tail patterns, we were surprised that some users failed to find more common patterns. Similarly, the second most common cause of precision loss are entity type restrictions that users failed to correctly select, again to our surprise. We proceeded to interview the users to determine reasons for this.

3.3 Exploration Task

We also asked users to explore the corpus for a vaguely defined information need, namely for relations that pertain to "celebrities", as well as one arbitrary relation. Users spent widely varying amounts of time (between 5 and 50 minutes) on this task due to differences in motivation, as some users had interpreted the search for "interesting" relations as a challenge. For each relation, users provided a short description.

Some relations not in Freebase. While the most common types of relations found for entities of type CELEBRITY regarded different types of romantic involvements with other celebrities such as marriages and divorces, some relations were identified that are not found in FREEBASE. This included a relation that connects a celebrity to the sports team they support or the car they drive (see Table 3). This indicates a potential for using ERE to identify new relations for addition to existing knowledge bases.

Closed-class words can be relevant. Interestingly, one user also worked with patterns that involved closed-class word classes, such as "if" and "whether". Table 3 shown an example of a relation that indicates speculative birthplaces using such words.

3.4 User Feedback and Discussion

Approach more suited to exploration than extraction. When interviewing the users, we found that they generally favored the exploration over the extraction tasks as here the search could be directed to more fine-granular and specialized relations. One of the main problems encountered was the "halting problem", i.e. the question of when to stop adding patterns to an extractor. For some relations, such as BORNIN, users already found thousands of relation instances after selecting the first pattern, which caused two problems; First, they were unsure of the quality of the selected pattern(s), as they were unable to manually check thousands of relation instances for their validity. Second, they were unsure if more patterns were even needed if the first few already found such amounts of relation instances. These problems were not encountered in the exploration tasks, as here users could decide the information need for themselves and select patterns accordingly.

Difficulties concerning entity types. Another main difficulty related to the precise meaning of FREEBASE entity types; For instance, there are several location types, such as LOCATION.LOCATION, LOCATION.DATED_LOCATION and LOCATION.STATISTICAL_REGION, which users found to be confusing, a problem that was compounded by occasional entity linking errors. Many users expressed the desire to specify custom entity types as restrictions in order to have a similar level of control here as over the choice of patterns.

Low entry barriers but allow additional complexity. Overall, we found that users were generally able to start exploring the corpus using our workflow immediately after the brief introduction. Users stated the natural language-like representation of patterns to be intuitively readable, although for some it required a trial and error process to understand how patterns matched entities in sentences. Similarly, some users wished to understand in greater detail how entity types are determined and whether this could

FALSE POSITIVES		
CLASS	COUNT	EXAMPLE SENTENCE
FB Mismatch	95	**Lincoln** died at **Hildene** , his Vermont home, on July 26, 1926.
Type Error	82	[..] the scene where **Boromir** is killed in The **Fellowship of the Ring**.
FB Incomplete	14	Later that year, on December 27, **Dorr** died in **Providence**, in his native Rhode Island.
Other	9	Brieven van liederen **Rascal Flatts** die in het schijfcd album omvatten **Feels Like Today**.

FALSE NEGATIVES		
CLASS	COUNT	EXAMPLE SENTENCE
Common	87	**Klein** holds a **Bachelor of Arts**.
Long Tail	79	**Roger Blandford** is a native of England and took his **BA**, MA and [..].
Other	34	[..], 1974; **MS**, 1976; PhD, University of Pierre and **Marie Curie**, 1982.

Table 2: Analysis of 200 false positives and 200 false negatives to determine error classes for precision and recall loss. Each error class is listed with an example sentence. Main reasons for false positives included a mismatch in granularity between extraction results and annotations, wrongly specified types by the users or cases in which instances were found that were not in FREEBASE. Main reasons for false negatives were mostly patterns that users failed so select, either common patterns, or more rare patterns from the long tail.

NAME	DESCRIPTION	EXAMPLE PATTERNS	EXAMPLE INSTANCES
CELEBRITYDIVORCE	Divorce between two celebrities	"X and Y divorce", "X divorce Y",	<Nicole Kidman, Tom Cruise> <Federline, Spears>
CELEBRITYDRIVESCAR	Finds the cars that celebrities drive	"X drives Y", "X 's car Y",	<Arnold Schwarzenegger, H1> <Leonardo DiCaprio, Toyota Prius>
CONTESTEDBITHPLACE	Relates persons to their speculative birthplace	"if X born in Y", "whether X born in Y",	<Barack Obama, Kenya> <Barack Obama, Nigeria>

Table 3: Examples for relations discovered in the exploration task. CELEBRITYDIVORCE represents a commonly discovered relation, while CELEBRITYDRIVESCAR represents a relation that is presently not part of Freebase. CONTESTEDBITHPLACE is an example of a relation that utilizes closed-world words in patterns.

be influenced. This indicates the need for adding options in future work that give more experienced users more technical information (and control) on dependency trees and FREEBASE types.

4 Previous Work

While no directly comparable approach to Exploratory Relation Extraction is known to us, we take inspiration from a number of previous works.

Exploratory Search (Marchionini, 2006; White and Roth, 2009) is an information seeking paradigm in the field of Information Retrieval, where - like in our proposed approach - users begin an exploration process with an imprecise information need and progressively discover available information to address and sharpen it. Unlike our approach, users search for documents and must consume the unstructured information themselves. We instead apply this paradigm to RE and strive to find structured, relational information in text corpora of unknown content as well as generate Realtion Extractors in the process.

Preemptive Information Extraction (Shinyama and Sekine, 2006), as well as much work in Open Information Extraction (Yates et al., 2007) that builds on this idea, is the preemptive (or open) extraction of all possible relations in a text corpus. We draw inspiration from this idea in our preemptive subtree generation approach; however, while we extract all possible subtrees for each relation regardless of whether they point to a relation or not, Preemptive and OpenIE approaches aim to produce facts and therefore much more narrowly extract predicates using rule-sets (Del Corro and Gemulla, 2013), classifiers (Schmitz et al., 2012) or both (Etzioni et al., 2011).

Manual Rule-Based RE. We also build our work on the field of manual, rule-based RE, which has been observed to be predominantly preferred industry solution due to interpretability of extraction rules and easy adaption to changing domains (Chiticariu et al., 2013; Chiticariu et al., 2010). The lack of tools to assist rule developers in exploring and choosing between different automatically generated rules has been stated to be one of the major challenges associated with rule-based RE systems. Recent research has moved towards more guided (Li et al., 2012) and more interactive (Akbik et al., 2013b) workflows for the creation of rule-based extractors. Our proposed approach follows this direction, but is the first approach to combine both with automatic suggestions and enable exploratory search for relations.

Precomputing Resources of Relational Patterns. Our work also bears some resemblance to previous work that have grouped similar extraction patterns into clusters (Li et al., 2011) or arranged them in a taxonomy (Nakashole et al., 2012), with the goal of facilitating relation extraction efforts. Contrary to these works, we do not precompute a static resource but rather continuously re-compute pattern suggestions on the basis of user interactions and the text corpus that the user is working with. In addition, our suggestions are based on both user-selected patterns as well as entity type restrictions.

5 Conclusion and Future Work

In this paper, we proposed Exploratory Relation Extraction as a method of exploring text corpora of uncertain content for relations of interest given an imprecise information need. We have presented and evaluated a user-driven and data-guided incremental exploration workflow that enables non-expert users to identify relations and create high precision extractors with minimal effort. Our results indicate that applying ideas from Exploratory Search to RE is beneficial and can extend the application of RE to use cases characterized by more imprecise information needs and uncertainty regarding the information content of available data. In order to facilitate the discussion of our approach with the research community, we release our work publicly through a Web demonstrator[3].

Future work will investigate extending the approach to relations that hold between an arbitrary number of entities as well as the detection of custom entity types. We aim to allow users to store and combine extractors - for example relation extractors that use custom entity type detectors - to address more complex information needs and distribute the exploration and extraction processes along larger groups of users. This way we seek to enable collaborative RE approaches for creating large knowledge bases from text.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This research is funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no ICT-2009-4-1 270137 'Scalable Preservation Environments' (SCAPE) and the German Federal Ministry of Education and Research (BMBF) under grant no. 01ISI2033 RADAR'.

References

A. Akbik, L. Visengeriyeva, J. Kirschnick, and A. Löser. 2013a. Effective selectional restrictions for unsupervised relation extraction. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*.

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

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, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick R Reiss, and Shivakumar Vaithyanathan. 2010. Systemt: an algebraic approach to declarative information extraction. In *ACL*, pages 128–137. Association for Computational Linguistics.

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. Transitionbased dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria*.

[3] The demonstrator is available at http://lucene.textmining.tu-berlin.de/

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 137–150, Berkeley, CA, USA. USENIX Association.

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.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 3–10. AAAI Press.

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).

Yunyao Li, Vivian Chu, Sebastian Blohm, Huaiyu Zhu, and Howard Ho. 2011. Facilitating pattern discovery for relation extraction with semantic-signature-based clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1415–1424. ACM.

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.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/AFNLP*, pages 1003–1011.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59.

Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. 2008. An algebraic approach to rule-based information extraction. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 933–942. IEEE.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

Sunita Sarawagi. 2008. Information extraction. *Foundations and trends in databases*, 1(3):261–377.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Alexander Schutz and Paul Buitelaar. 2005. Relext: A tool for relation extraction from text in ontology extension. In *The Semantic Web–ISWC 2005*, pages 593–606. Springer.

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.

Hans Uszkoreit. 2011. Learning relation extraction grammars with minimal human intervention: strategy, results, insights and plans. In *Computational Linguistics and Intelligent Text Processing*, pages 106–126. Springer.

Chang Wang, Aditya Kalyanpur, James Fan, Branimir K Boguraev, and DC Gondek. 2012. Relation extraction and scoring in deepqa. *IBM Journal of Research and Development*, 56(3.4):9–1.

Ryen W White and Resa A Roth. 2009. Exploratory search: Beyond the query-response paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–98.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics.