# WizIE: A Best Practices Guided Development Environment for Information Extraction

**Yunyao Li Laura Chiticariu Huahai Yang Frederick R. Reiss Arnaldo Carreno-fuentes**
IBM Research - Almaden
650 Harry Road
San Jose, CA 95120
{yunyaoli,chiti,hyang,frreiss,acarren}@us.ibm.com

## Abstract

Information extraction (IE) is becoming a critical building block in many enterprise applications. In order to satisfy the increasing text analytics demands of enterprise applications, it is crucial to enable developers with general computer science background to develop high quality IE extractors. In this demonstration, we present WizIE, an IE development environment intended to reduce the development life cycle and enable developers with little or no linguistic background to write high quality IE rules. WizIE provides an integrated wizard-like environment that guides IE developers step-by-step throughout the entire development process, based on best practices synthesized from the experience of expert developers. In addition, WizIE reduces the manual effort involved in performing key IE development tasks by offering automatic result explanation and rule discovery functionality. Preliminary results indicate that WizIE is a step forward towards enabling extractor development for novice IE developers.

## 1 Introduction

Information Extraction (IE) refers to the problem of extracting structured information from unstructured or semi-structured text. It has been well-studied by the Natural Language Processing research community for a long time. In recent years, IE has emerged as a critical building block in a wide range of enterprise applications, including financial risk analysis, social media analytics and regulatory compliance, among many others. An important practical challenge driven by the use of IE in these applications is usability (Chiticariu et al., 2010c): specifically, how to enable the ease of development and maintenance of high-quality information extraction rules, also known as *annotators*, or *extractors*.

Developing extractors is a notoriously labor-intensive and time-consuming process. In order to ensure highly accurate and reliable results, this task is traditionally performed by trained linguists with domain expertise. As a result, extractor development is regarded as a major bottleneck in satisfying the increasing text analytics demands of enterprise applications. Hence, reducing the extractor development life cycle is a critical requirement. Towards this goal, we have built WizIE, an IE development environment designed primarily to (1) enable developers with little or no linguistic background to write high quality extractors, and (2) reduce the overall manual effort involved in extractor development.

Previous work on improving the usability of IE systems has mainly focused on reducing the manual effort involved in extractor development (Brauer et al., 2011; Li et al., 2008; Li et al., 2011a; Soderland, 1999; Liu et al., 2010). In contrast, the focus of WizIE is on lowering the extractor development entry barrier by means of a wizard-like environment that guides extractor development based on best practices drawn from the experience of trained linguists and expert developers. In doing so, WizIE also provides natural entry points for different tools focused on reducing the effort required for performing common tasks during IE development.

Underlying our WizIE are a state-of-the-art IE rule language and corresponding runtime engine (Chiticariu et al., 2010a; Li et al., 2011b). The runtime engine and WizIE are commercially avail-

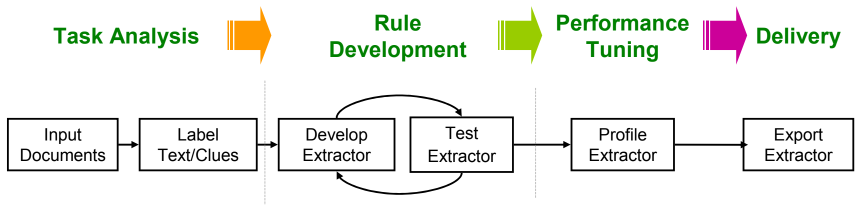


Figure 1: Best Practices for Extractor Development

able as part of IBM InfoSphere BigInsights (IBM, 2012).

## 2 System Overview

The development process for high-quality, high-performance extractors consists of four phases, as illustrated in Fig. 1. First, in the *Task Analysis* phase, concrete extraction tasks are defined based on high-level business requirements. For each extraction task, IE rules are developed during the *Rule Development* phase. The rules are profiled and further fine-tuned in the *Performance Tuning* phase, to ensure high runtime performance. Finally, in the *Delivery* phase, the rules are packaged so that they can be easily embedded in various applications.

WizIE is designed to assist and enable both novice and experienced developers by providing an intuitive wizard-like interface that is informed by the best practices in extractor development throughout each of these phases. By doing so, WizIE seeks to provide the key missing pieces in a conventional IE development environment (Cunningham et al., 2002; Li et al., 2011b; Soundrarajan et al., 2011), based on our experience as expert IE developers, as well as our interactions with novice developers with general computer science background, but little text analytics experience, during the development of several enterprise applications.

## 3 The Development Environment

In this section, we present the general functionality of WizIE in the context of extraction tasks driven by real business use cases from the media and entertainment domain. We describe WizIE in details and show how it guides and assists IE developers in a step-by-step fashion, based on best practices.

### 3.1 Task Analysis

The high-level business requirement of our running example is to identify *intention to purchase* for movies from online forums. Such information is of great interest to marketers as it helps predict future purchases (Howard and Sheth, 1969). During the first phrase of IE development (Fig. 2), WizIE guides the rule developer in turning such a high-level business requirement into concrete extraction tasks by explicitly asking her to select and manually examine a small number [1] of sample documents, identify and label snippets of interest in the sample documents, and capture clues that help to identify such snippets.

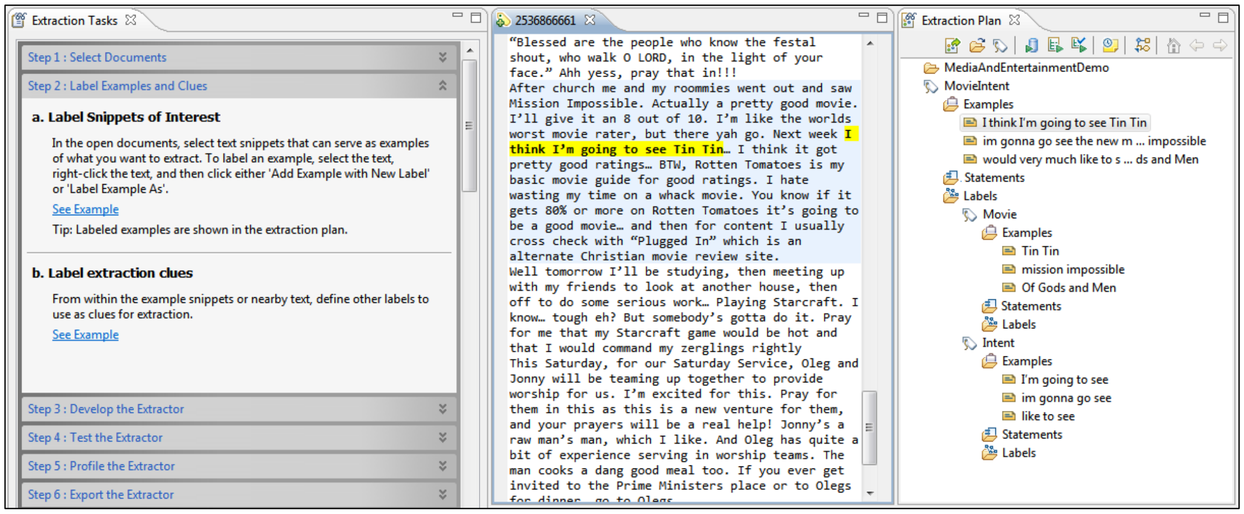
The definition and context of the concrete extraction tasks are captured by a tree structure called the *extraction plan* (e.g. right panel in Fig. 2). Each leaf node in an extraction plan corresponds to an atomic extraction task, while the non-leaf nodes denote higher-level tasks based on one or more atomic extraction tasks. For instance, in our running example, the business question of identifying intention of purchase for movies has been converted into the extraction task of identifying *MovieIntent* mentions, which involves two atomic extraction tasks: identifying *Movie* mentions and *Intent* mentions.

The extraction plan created, as we will describe later, plays a key role in the IE development process in WizIE. Such tight coupling of task analysis with actual extractor development is a key departure from conventional IE development environments.

### 3.2 Rule Development

Once concrete extraction tasks are defined, WizIE guides the IE developer to write actual rules based on best practices. Fig. 3(a) shows a screenshot of the second phase of building an extractor, the *Rule Development* phase. The *Extraction Task* panel on the left provides information and tips for rule development, whereas the *Extraction Plan* panel on the right guides the actual rule development for each extraction task. As shown in the figure, the types of rules associated with each label node fall into three categories: *Basic Features*, *Can-*

[1]The exact sample size varies by task type.


Extraction Tasks

Step 1 : Select Documents

Step 2 : Label Examples and Clues

**a. Label Snippets of Interest**

In the open documents, select text snippets that can serve as examples of what you want to extract. To label an example, select the text, right-click the text, and then click either 'Add Example with New Label' or 'Label Example As'.

See Example

Tip: Labeled examples are shown in the extraction plan.

**b. Label extraction clues**

From within the example snippets or nearby text, define other labels to use as clues for extraction.

See Example

Step 3 : Develop the Extractor

Step 4 : Test the Extractor

Step 5 : Profile the Extractor

Step 6 : Export the Extractor

2536866661

```
“Blessed are the people who know the festal
shout, who walk O LORD, in the light of your
face.” Ahh yess, pray that in!!!
After church me and my roommies went out and saw
Mission Impossible. Actually a pretty good movie.
I’ll give it an 8 out of 10. I’m like the worlds
worst movie rater, but there yah go. Next week I
think I’m going to see Tin Tin… I think it got
pretty good ratings… BTW, Rotten Tomatoes is my
basic movie guide for good ratings. I hate
wasting my time on a whack movie. You know if it
gets 80% or more on Rotten Tomatoes it’s going to
be a good movie… and then for content I usually
cross check with “Plugged In” which is an
alternate Christian movie review site.
Well tomorrow I’ll be studying, then meeting up
with my friends to look at another house, then
off to do some serious work… Playing Starcraft. I
know… tough eh? But somebody’s gotta do it. Pray
for me that my Starcraft game would be hot and
that I would command my zerglings rightly
This Saturday, for our Saturday Service, Oleg and
Jonny will be teaming up together to provide
worship for us. I’m excited for this. Pray for
them in this as this is a new venture for them,
and your prayers will be a real help! Jonny’s a
raw man’s man, which I like. And Oleg has quite a
bit of experience serving in worship teams. The
man cooks a dang good meal too. If you ever get
invited to the Prime Ministers place or to Olegs
for dinner, go to Olegs
```

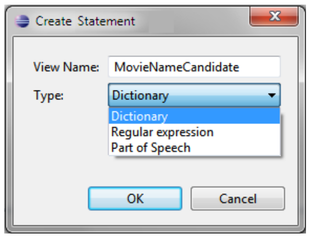
Extraction Plan

- MediaAndEntertainmentDemo
- MovieIntent
  - Examples
    - I think I'm going to see Tin Tin
    - im gonna go see the new m … impossible
    - would very much like to s … ds and Men
  - Statements
  - Labels
    - Movie
      - Examples
        - Tin Tin
        - mission impossible
        - Of Gods and Men
      - Statements
      - Labels
    - Intent
      - Examples
        - I'm going to see
        - im gonna go see
        - like to see
      - Statements
      - Labels

Figure 2: Labeling Snippets and Clues of Interest

*didate Generation* and *Filter&Consolidate*. This categorization is based on best practices for rule development (Chiticariu et al., 2010b). As such, the extraction plan groups together the high-level specification of extraction tasks via examples, and the actual implementation of those tasks via rules.

The developer creates rules directly in the Rule Editor, or via the *Create Statement* wizard, accessible from the *Statements* node of each label in the Extraction Plan panel:


Create Statement

View Name: MovieNameCandidate

Type: Dictionary

- Dictionary
- Regular expression
- Part of Speech

The wizard allows the user to select a type for the new rule, from predefined sets for each of the three categories. The types of rules exposed in each category are informed by best practices. For example, the Basic Features category includes rules for defining basic features using regular expressions, dictionaries or part of speech information, whereas the Candidate Generation category includes rules for combining basic features into candidate mentions by means of operations such as sequence or alternation. Once the developer provides a name for the new rule (*view*) and selects its type, the appropriate rule template (such as the one illustrated below) is automatically generated in an appropriate file on disk and displayed in the editor, for further editing [2].

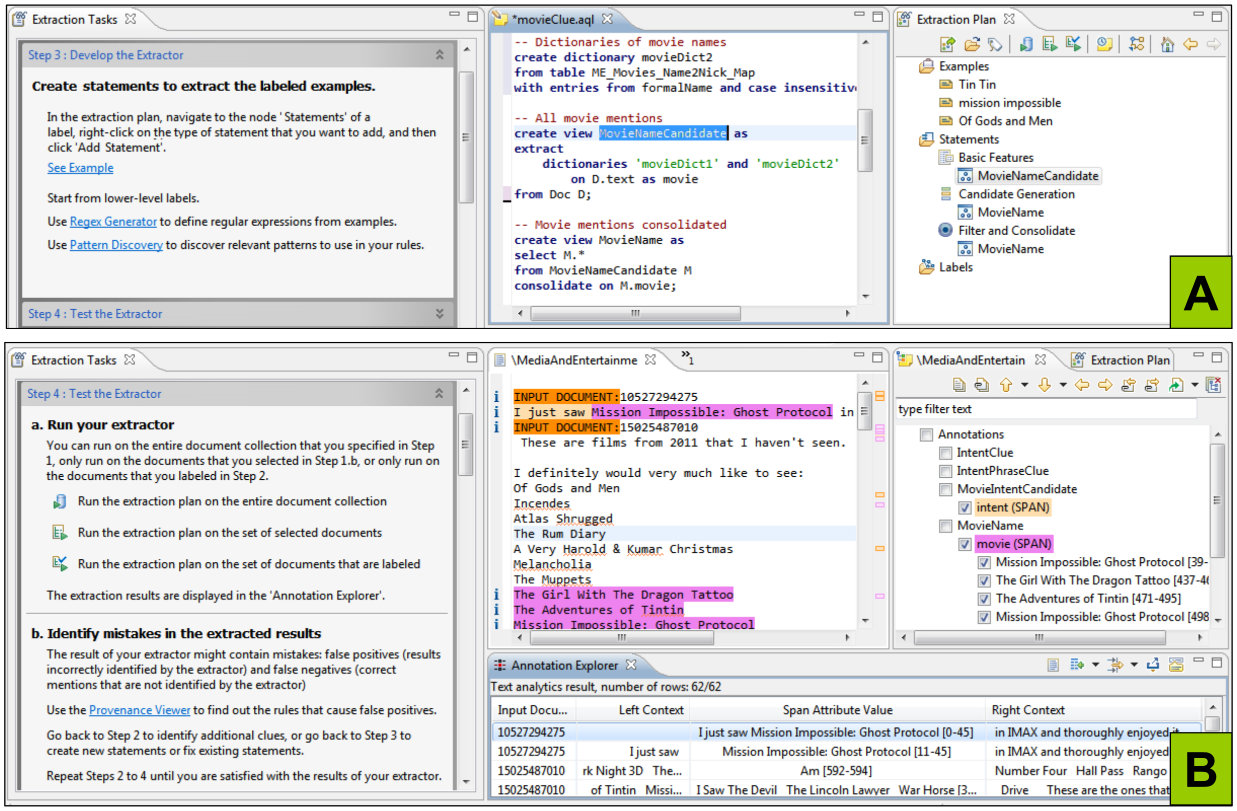
```
create dictionary MovieNameCandidateDict
from file '<path to your dictionary here>'
with language as 'en';

create view MovieNameCandidate as
extract dictionary 'MovieNameCandidateDict'
    on R.<input column> as match
from <input view> R;
```

Once the developer completes an iteration of rule development, WizIE guides her in testing and refining the extractor, as shown in Fig. 3(b). The *Annotation Explorer* at the bottom of the screen gives a global view of the extraction results, while other panels highlight individual results in the context of the original input documents. The Annotation Explorer enables filtering and searching results, and comparing results with those from a previous iteration. WizIE also provides a facility for manually labeling a document collection with “ground truth” annotations, then comparing the extraction results with the ground truth in order to formally evaluate the quality of the extractor and avoid regressions during the development process.

An important differentiator of WizIE compared with conventional IE development environments is a suite of sophisticated tools for automatic *result explanation* and *rule discovery*. We briefly describe them next.

**Provenance Viewer.** When the user clicks on an extracted result, the Provenance Viewer shows a complete explanation of how that result has been pro-

[2] Details on the rule syntax can be found in (IBM, )



Step 3 : Develop the Extractor

**Create statements to extract the labeled examples.**

In the extraction plan, navigate to the node 'Statements' of a label, right-click on the type of statement that you want to add, and then click 'Add Statement'.

See Example

Start from lower-level labels.

Use Regex Generator to define regular expressions from examples.

Use Pattern Discovery to discover relevant patterns to use in your rules.

Step 4 : Test the Extractor

```
-- Dictionaries of movie names
create dictionary movieDict2
from table ME_Movies_Name2Nick_Map
with entries from formalName and case insensitiv

-- All movie mentions
create view MovieNameCandidate as
extract
    dictionaries 'movieDict1' and 'movieDict2'
        on D.text as movie
from Doc D;

-- Movie mentions consolidated
create view MovieName as
select M.*
from MovieNameCandidate M
consolidate on M.movie;
```

Examples
Tin Tin
mission impossible
Of Gods and Men
Statements
Basic Features
MovieNameCandidate
Candidate Generation
MovieName
Filter and Consolidate
MovieName
Labels

Step 4 : Test the Extractor

**a. Run your extractor**

You can run on the entire document collection that you specified in Step 1, only run on the documents that you selected in Step 1.b, or only run on the documents that you labeled in Step 2.

Run the extraction plan on the entire document collection

Run the extraction plan on the set of selected documents

Run the extraction plan on the set of documents that are labeled

The extraction results are displayed in the 'Annotation Explorer'.

**b. Identify mistakes in the extracted results**

The result of your extractor might contain mistakes: false positives (results incorrectly identified by the extractor) and false negatives (correct mentions that are not identified by the extractor)

Use the Provenance Viewer to find out the rules that cause false positives.

Go back to Step 2 to identify additional clues, or go back to Step 3 to create new statements or fix existing statements.

Repeat Steps 2 to 4 until you are satisfied with the results of your extractor.

\MediaAndEntertainme

```
INPUT DOCUMENT:10527294275
I just saw Mission Impossible: Ghost Protocol in
INPUT DOCUMENT:15025487010
 These are films from 2011 that I haven't seen.

I definitely would very much like to see:
Of Gods and Men
Incendes
Atlas Shrugged
The Rum Diary
A Very Harold & Kumar Christmas
Melancholia
The Muppets
The Girl With The Dragon Tattoo
The Adventures of Tintin
Mission Impossible: Ghost Protocol
```

\MediaAndEntertain | Extraction Plan

type filter text

Annotations
IntentClue
IntentPhraseClue
MovieIntentCandidate
✓ intent (SPAN)
MovieName
✓ movie (SPAN)
✓ Mission Impossible: Ghost Protocol [39-
✓ The Girl With The Dragon Tattoo [437-4(
✓ The Adventures of Tintin [471-495]
✓ Mission Impossible: Ghost Protocol [498

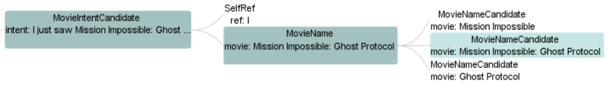
Annotation Explorer

Text analytics result, number of rows: 62/62

| Input Docu... | Left Context | Span Attribute Value | Right Context |
|---|---|---|---|
| 10527294275 | | I just saw Mission Impossible: Ghost Protocol [0-45] | in IMAX and thoroughly enjoyed it |
| 10527294275 | I just saw | Mission Impossible: Ghost Protocol [11-45] | in IMAX and thoroughly enjoyed |
| 15025487010 | rk Night 3D The... | Am [592-594] | Number Four Hall Pass Rango |
| 15025487010 | of Tintin Missi... | I Saw The Devil The Lincoln Lawyer War Horse [3... | Drive These are the ones that |

Figure 3: Extractor Development: (a) Developing, and (b) Testing.

duced by the extractor, in the form of a graph that demonstrates the sequence of rules and individual pieces of text responsible for that result. Such explanations are critical to enable the developer to understand why a false positive is generated by the system, and identify problematic rule(s) that could be refined in order to correct the mistake. An example explanation for an incorrect *MovieIntent* mention *“I just saw Mission Impossible”* is shown below.

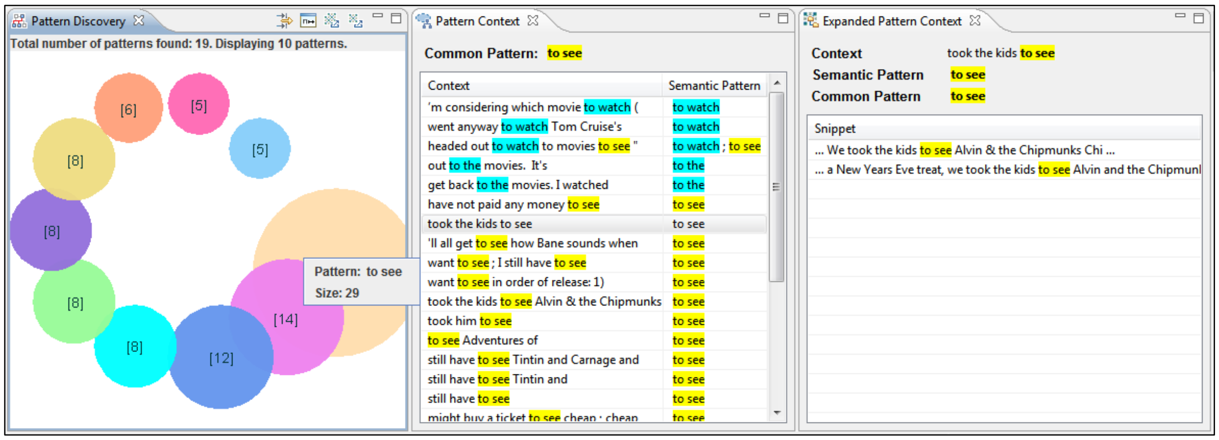


MovieIntentCandidate
intent: I just saw Mission Impossible: Ghost ...
SelfRef
ref: I
MovieName
movie: Mission Impossible: Ghost Protocol
MovieNameCandidate
movie: Mission Impossible
MovieNameCandidate
movie: Mission Impossible: Ghost Protocol
MovieNameCandidate
movie: Ghost Protocol

As can be seen, the *MovieIntent* mention is generated by combining a *SelfRef* (matching first person pronouns) with a *MovieName* mention, and in turn, the latter is obtained by combining several *MovieNameCandidate* mentions. With this information, the developer can quickly determine that the *SelfRef* and *MovieName* mentions are correct, but their combination in *MovieIntentCandidate* is problematic. She can then proceed to refine the *MovieIntentCandidate* rule, for example, by avoiding any *MovieIntentCandidate* mentions containing a past tense verb form such as *saw*, since past tense in not usually indicative of intent (Liu et al., 2010).

**Pattern Discovery.** Negative contextual clues such as the verb *“saw”* above are useful for creating rules that filter out false positives. Conversely, positive clues such as the phrase *“will see”* are useful for creating rules that separate ambiguous matches from high-precision matches. WizIE's *Pattern Discovery* component facilitates automatic discovery of such clues by mining available sample data for common patterns in specific contexts (Li et al., 2011a). For example, when instructed to analyze the context between *SelfRef* and *MovieName* mentions, Pattern Discovery finds a suite of common patterns as shown in Fig. 4. The developer can analyze these patterns and choose those suitable for refining the rules. For example, patterns such as *“have to see”* can be seen as positive clues for intent, whereas phrases such as *“took ... to see”* or *“went to see”* are negative clues, and can be used for filtering false positives.

**Regular Expression Generator.** WizIE also enables the discovery of regular expression patterns. The Regular Expression Generator takes as input a


Pattern Discovery

Total number of patterns found: 19. Displaying 10 patterns.

Pattern Context

Common Pattern: to see

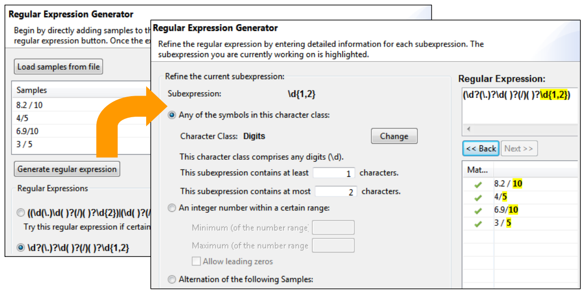
| Context | Semantic Pattern |
|---|---|
| 'm considering which movie to watch ( | to watch |
| went anyway to watch Tom Cruise's | to watch |
| headed out to watch to movies to see " | to watch ; to see |
| out to the movies. It's | to the |
| get back to the movies. I watched | to the |
| have not paid any money to see | to see |
| took the kids to see | to see |
| 'll all get to see how Bane sounds when | to see |
| want to see ; I still have to see | to see |
| want to see in order of release: 1) | to see |
| took the kids to see Alvin & the Chipmunks | to see |
| took him to see | to see |
| to see Adventures of | to see |
| still have to see Tintin and Carnage and | to see |
| still have to see Tintin and | to see |
| still have to see | to see |
| might buy a ticket to see cheap : cheap | to see |

Expanded Pattern Context

| | |
|---|---|
| Context | took the kids to see |
| Semantic Pattern | to see |
| Common Pattern | to see |

| Snippet |
|---|
| ... We took the kids to see Alvin & the Chipmunks Chi ... |
| ... a New Years Eve treat, we took the kids to see Alvin and the Chipmunl |

Figure 4: Pattern Discovery


Regular Expression Generator

Begin by directly adding samples to th
regular expression button. Once the e

Load samples from file

| Samples |
|---|
| 8.2 / 10 |
| 4/5 |
| 6.9/10 |
| 3 / 5 |

Generate regular expression

Regular Expressions

( ) ((\d(\.)\d( )?(/)( )?\d{2})|(\d( )?(/
Try this regular expression if certain

(●) \d?(\.)?\d( )?(/)( )?\d{1,2}

Regular Expression Generator

Refine the regular expression by entering detailed information for each subexpression. The subexpression you are currently working on is highlighted.

Refine the current subexpression:

Subexpression: \d{1,2}

(●) Any of the symbols in this character class:

Character Class: Digits

Change

This character class comprises any digits (\d).

This subexpression contains at least 1 characters.

This subexpression contains at most 2 characters.

( ) An integer number within a certain range:

Minimum (of the number range

Maximum (of the number range

☐ Allow leading zeros

( ) Alternation of the following Samples:

Regular Expression:

(\d?(\.)?\d( )?(/)( )?\d{1,2})

<< Back   Next >>

| Mat... | |
|---|---|
| ✓ | 8.2 / 10 |
| ✓ | 4/5 |
| ✓ | 6.9/10 |
| ✓ | 3 / 5 |

Figure 5: Regular Expression Generator

set of sample mentions and suggests regular expressions that capture the samples, ranging from more specific (higher accuracy) to more general expressions (higher coverage). Figure 5 shows two regular expressions automatically generated based on mentions of movie ratings, and how the developer is subsequently assisted in understanding and refining the generated expression. In our experience, regular expressions are complex concepts that are difficult to develop for both expert and novice developers. Therefore, such a facility to generate expressions based on examples is extremely useful.

### 3.3 Performance Tuning

Once the developer is satisfied with the quality of the extractor, WizIE guides her in measuring and tuning its runtime performance, in preparation for deploying the extractor in a production environment. The Profiler observes the execution of the extractor on a sample input collection over a period of time and records the percentage of time spent executing each rule, or performing certain runtime operations. After the profiling run completes, WizIE displays the top 25 most expensive rules and runtime operations, and the overall throughput (amount of input data processed per unit of time). Based on this information, the developer can hand-tune the critical parts of the extractor, rerun the Profiler, and validate an increase in throughput. She would repeat this process until satisfied with the extractor's runtime performance.

### 3.4 Delivery and Deployment

Once satisfied with both the result quality and runtime performance, the developer is guided by WizIE's Export wizard through the process of exporting the extractor in a compiled executable form. The generated executable can be embedded in an application using a Java API interface. WizIE can also wrap the executable plan in a pre-packaged application that can be run in a map-reduce environment, then deploy this application on a Hadoop cluster.

## 4 Evaluation

A preliminary user study was conducted to evaluate the effectiveness of WizIE in enabling novice IE developers. The study included 14 participants, all employed at a major technology company. In the pre-study survey, 10 of the participants reported no prior experience with IE tasks, two of them have seen demonstrations of IE systems, and two had brief involvement in IE development, but no experience with WizIE. For the question *"According to your understanding, how easy is it to build IE applications in general ?"*, the median rating was 5, on a scale of 1 (very easy) to 7 (very difficult).

The study was conducted during a 2-day training session. In Day 1, participants were given a thorough introduction to IE, shown example extractors, and instructed to develop extractors without WizIE. Towards the end of Day 1, participants were asked to solve an IE exercise: develop an extractor for the high-level requirement of identifying mentions of company revenue by division from the company's official press releases. WizIE was introduced to the participants in Day 2 of the training, and its features were demonstrated and explained with examples. Participants were then asked to complete the same exercise as in Day 1. Authors of this demonstration were present to help participants during the exercises in both days. At the end of each day, participants filled out a survey about their experience.

In Day 1, none of the participants were able to complete the exercise after 90 minutes. In the survey, one participant wrote *"I am in sales so it is all difficult"*; another participant indicated that *"I don't think I would be able to recreate the example on my own from scratch"*. In Day 2, most participants were able to complete the exercise in 90 minutes or less using WizIE. In fact, two participants created extractors with accuracy and coverage of over 90%, when measured against the ground truth. Overall, the participants were much more confident about creating extractors. One participant wrote *"My first impression is very good"*. On the other hand, another participant asserted that *"The nature of the task is still difficult"*. They also found that WizIE is useful and easy to use, and it is easier to build extractors with the help of WizIE.

In summary, our preliminary results indicate that WizIE is a step forward towards enabling extractor development for novice IE developers. In order to formally evaluate WizIE, we are currently conducting a formal study of using WizIE to create extractors for several real business applications.

## 5 Demonstration

In this demonstration we showcase WizIE's step-by-step approach to guide the developer in the iterative process of IE rule development, from task analysis to developing, tuning and deploying the extractor in a production environment. Our demonstration is centered around the high-level business requirement of identifying intent to purchase movies from blogs and forum posts as described in Section 3. We start by demonstrating the process of developing two relatively simple extractors for identifying *MovieIntent* and *MovieRating* mentions. We then showcase complex state-of-the-art extractors for identifying *buzz* and *sentiment* for the media and entertainment domain, to illustrate the quality and runtime performance of extractors built with WizIE.

## References

F. Brauer, R. Rieger, A. Mocan, and W. M. Barczynski. 2011. Enabling information extraction by inference of regular expressions from sample entities. In *CIKM*.

L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. 2010a. SystemT: an algebraic approach to declarative information extraction. ACL.

L. Chiticariu, R. Krishnamurthy, Y. Li, F. Reiss, and S. Vaithyanathan. 2010b. Domain adaptation of rule-based annotators for named-entity recognition tasks. EMNLP.

L. Chiticariu, Y. Li, S. Raghavan, and F. Reiss. 2010c. Enterprise Information Extraction: Recent Developments and Open Challenges. In *SIGMOD (Tutorials)*.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. Gate: an architecture for development of robust hlt applications. In *ACL*.

J.A. Howard and J.N. Sheth. 1969. *The Theory of Buyer Behavior*. Wiley.

IBM. InfoSphere BigInsights - Annotation Query Language (AQL) reference. http://ibm.co/kkzj1i.

IBM. 2012. InfoSphere BigInsights. http://ibm.co/jjbjfa.

Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. V. Jagadish. 2008. Regular expression learning for information extraction. In *EMNLP*.

Y. Li, V. Chu, S. Blohm, H. Zhu, and H. Ho. 2011a. Facilitating pattern discovery for relation extraction with semantic-signature-based clustering. In *CIKM*.

Y. Li, F. Reiss, and L. Chiticariu. 2011b. SystemT: A Declarative Information Extraction System. In *ACL (Demonstration)*.

B. Liu, L. Chiticariu, V. Chu, H. V. Jagadish, and F. Reiss. 2010. Automatic Rule Refinement for Information Extraction. *PVLDB*, 3(1):588–597.

S. Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, February.

B. R. Soundrarajan, T. Ginter, and S. L. DuVall. 2011. An interface for rapid natural language processing development in UIMA. In *ACL (Demonstrations)*.