IKE - An Interactive Tool for Knowledge Extraction

Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, Dirk Groeneveld
Allen Institute for Artificial Intelligence
{bhavanad,sumithrab,chrisc,peterc,orene,dirkg}@allenai.org

Abstract

Recent work on information extraction has suggested that fast, interactive tools can be highly effective; however, creating a usable system is challenging, and few publicly available tools exist. In this paper we present IKE, a new extraction tool that performs fast, interactive bootstrapping to develop high-quality extraction patterns for targeted relations. Central to IKE is the notion that an extraction pattern can be treated as a search query over a corpus. To operationalize this, IKE uses a novel query language that is expressive, easy to understand, and fast to execute - essential requirements for a practical system. It is also the first interactive extraction tool to seamlessly integrate symbolic (boolean) and distributional (similarity-based) methods for search. An initial evaluation suggests that relation tables can be populated substantially faster than by manual pattern authoring while retaining accuracy, and more reliably than fully automated tools, an important step towards practical KB construction. We are making IKE publically available (http://allenai.org/software/interactive-knowledge-extraction).

1 Introduction

Knowledge extraction from text remains a fundamental challenge for any system that works with structured data. Automatic extraction algorithms, e.g., (Angeli et al., 2015; Carlson et al., 2009; Nakashole et al., 2011), have proved efficient and scalable, especially when leveraging existing search engine technologies, e.g., (Etzioni et al., 2004), but typically produce noisy results, e.g., the best F1 score for the KBP slot filling task was 0.28, as reported in (Angeli et al., 2015). Weakly supervised automatic bootstrapping methods (Carlson et al., 2010; Gupta and Manning, 2014) are more precise in the initial bootstrapping iterations, but digress in later iterations, a problem generally referred to as semantic drift.

More recently there has been work on more interactive methods, which can be seen as a “machine teaching” approach to KB construction (Amershi et al., 2014; Amershi et al., 2015; Li et al., 2012). For example, (Soderland et al., 2013) showed that users can be surprisingly effective at authoring and refining extraction rules for a slot filling task, and (Freedman et al., 2011) demonstrated that a combination of machine learning and user authoring produced high quality results. However, none of these approaches have evolved into publically available tools.

In this paper we present IKE, a usable, general-purpose tool for interactive extraction. Central to IKE is the notion that an extraction pattern can be treated as a search query over a corpus, building on earlier work by (Cafarella et al., 2005). It addresses the resulting requirements of expressiveness, comprehensibility, and speed with a novel query language based on chunking rather than parsing, and is the first tool to seamlessly integrate symbolic (boolean) and distributional (similarity-based) methods for search. It also includes a machine learning component for suggesting new queries to the user. A preliminary evaluation suggests that relation tables can be populated substantially faster with IKE than by manual pattern authoring (and more reliably than fully automated tools), while retaining accu-
racy, suggesting IKE has utility for KB construction.

<table>
<thead>
<tr>
<th>Query</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>the dog</td>
<td>matches “the” followed by “dog”</td>
</tr>
<tr>
<td>NP grows</td>
<td>an NP followed by “grows”</td>
</tr>
<tr>
<td>(NP) grows</td>
<td>Capture the NP and place in column 1 (1 column table)</td>
</tr>
<tr>
<td>(NP) conducts (NP)</td>
<td>Capture the two NPs into columns 1 and 2 (2 column table)</td>
</tr>
<tr>
<td>(?&lt;Energy&gt; NP) is conducted by (?&lt;Material&gt; NP)</td>
<td>Capture the two NPs and place in columns named Energy and Material</td>
</tr>
<tr>
<td>the {cat,dog}</td>
<td>“the” followed by “cat” or “dog”</td>
</tr>
<tr>
<td>cats and {NN,NNS}</td>
<td>“cats and” followed by NN or NNS</td>
</tr>
<tr>
<td>JJ* dog</td>
<td>Zero or more JJ then “dog”</td>
</tr>
<tr>
<td>JJ+ dog</td>
<td>One or more JJ then “dog”</td>
</tr>
<tr>
<td>JJ[2-4] dog</td>
<td>2 to 4 JJ then “dog”</td>
</tr>
<tr>
<td>dog .[0-4] tail</td>
<td>“dog” followed by any 0 to 4 words followed by “tail”</td>
</tr>
<tr>
<td>dog~50</td>
<td>Matches “dog” and the 50 words most distributionally similar to “dog”</td>
</tr>
<tr>
<td>. dog</td>
<td>Any word then “dog”</td>
</tr>
<tr>
<td>$colors</td>
<td>Any entry in the single-column “colors” table</td>
</tr>
<tr>
<td>$colors ~100</td>
<td>same plus 100 most similar words</td>
</tr>
<tr>
<td>$flower.color</td>
<td>Any in the “color” column of “flower” table</td>
</tr>
</tbody>
</table>

Table 1: IKE’s Query Language, described by example.

2 Interactive Knowledge Extraction (IKE)

We first overview IKE and a sample workflow using it. IKE allows the user to create relation tables, and populate them by issuing pattern-based queries over a corpus. It also has a machine learning component that suggests high-quality broadenings or narrowings of the user’s queries. Together, these allow the user to perform fast, interactive bootstrapping.

2.1 IKE’s Query Language

A key part of IKE is treating an extraction pattern as a search query. To do this, the query language must be both comprehensible and fast to execute. To meet these requirements, IKE indexes and searches the corpus using a chunk-based rather than dependency-based representation of the text corpus. IKE’s query language is presented by example in Table 1. The query language supports wildcards, window sizes, POS tags, chunk tags, and general regular expression queries similar to TokenRegex (Chang and Manning, 2014) and Lucene, ElasticSearch’s (Gormley and Tong, 2015) query language. Additionally, IKE supports distributional similarity based search (e.g. dog~50 would find 50 words similar to “dog”). “Capture groups”, indicated by parentheses, instruct IKE to capture the matching element(s) as candidate entries in the table being populated. The user can also reference data in other already-constructed tables using the $ prefix.

The use of a chunk-based representation has several advantages over a dependency-based one (e.g., Freedman et al., 2011; Gamallo et al., 2012; Hoffmann et al., 2015; Akbik et al., 2013). First, both indexing and search are very fast (e.g., <1 sec to execute a query over 1.5M sentences), essential for an interactive system. Second, authoring queries does not require detailed knowledge of dependency structure, making the language more accessible. Finally, the system avoids parse errors, a considerable challenge for dependency-based systems. Corresponding challenges with chunk-based representations, e.g., defining constituent boundaries in terms of POS chunks, are partially alleviated by providing predefined, higher-level POS-based patterns, e.g., for verb phrases.

2.2 Machine Learning

IKE also has a ML-based Query Suggestor to propose improved queries to the user. This module performs a depth-limited beam search to explore the space of query variants, evaluated on the user-annotated examples collected so far. Variants are generated by broadening/narrowing a query.

Narrowing a query involves searching the space of restrictions on the current query, e.g., replacing a POS tag with a specific word, adding prefixes or suffixes to the query, adjusting distributional similarity based queries etc. Similarly, the broaden feature generalizes the given user query e.g. replacing a word by its POS tag. In both cases the candidate queries are ranked by the weighted sum of the number of positive \( n_p \), negative \( n_n \), and unlabeled \( n_u \) instances it matches, the weights being user-configurable (default 2, -1, -0.05 respectively). For example, for the query

\[(\$conducts.Material) VBZ (\$conducts.Energy)\]
the top three suggested narrowings are:

\[(\text{conduits}.\text{Material}) \text{ conduits} (\text{conduits}.\text{Energy})\]

\[(\text{conduits}.\text{Material}) \text{ absorbs} (\text{conduits}.\text{Energy})\]

\[(\text{conduits}.\text{Material}) \text{ produces} (\text{conduits}.\text{Energy})\]

all patterns that distinguish positive examples from negatives well.

2.3 Example Workflow

We now describe these features in more detail by way of an example. Consider the task of acquiring instances of the binary predicate \text{conduits}(*\text{material},\text{energy}), e.g., \text{conduits}(\text{“steel”}, \text{“electricity”}). In IKE, relations are visualized as tables, so we treat this task as one of table population. A typical workflow is illustrated in Figure 1, which we now describe.

2.3.1 Define the types \text{material} and \text{energy}

First, the user defines the argument types \text{material} and \text{energy}. To define a type, IKE lets a user build a single-column table, e.g., for type \text{material}, the user:

1. Creates a single column table called Material.
2. Manually adds several representative examples in the table, e.g., “iron”, “wood”, “steel”.
3. Expands this set by searching for cosine-vector-similar phrases in the corpus, and marking valid and invalid members, e.g., the query $\text{Material} \sim 20$ searches for the 20 phrases most similar to any existing member in the Material table, where similar is defined as the cosine between the phrase embeddings. Here we use 300 dimensional word2vec embeddings learned using (Mikolov et al., 2013)’s implementation of word2vec on a science document corpus.

4. Repeat step 3 until the table adequately characterizes the intended notion of \text{material}

The process is repeated for the type \text{energy}. Note that here we are using only embedding-based features to populate a type. We could also use Hearst-style patterns (Hearst, 1992; Snow et al., 2004; Turney, 2006) to populate the type, e.g.:

\[
\begin{align*}
\text{materials such as (NP)} \\
\text{materials such as $\text{Material}$ and (NP)}
\end{align*}
\]

or a combination of patterns and word2vec expressions.

2.3.2 Create and Seed the \text{conduits} Table

The user next creates a two-column table \text{conduits}, and then uses a seed pattern to find initial pairs to populate it, e.g., the pattern

\[(\text{Material}) \text{ conduits} (\text{Energy})\]

extracts pairs of materials and energies they conduct. The user selects valid pairs to initially populate the table. Invalid pairs (negative examples) are also recorded by IKE.

2.3.3 Bootstrapping to Expand the Table

The user can now bootstrap by invoking the ML-based Query Suggestor to find additional patterns (queries) expressing the target relation (Section 2.2). It does so by searching for narrowings or broadenings of the current query that cover a large number of positive pairs and few of the negative pairs in the table so far. The user then clicks on one of these patterns to select and execute it (with edits if desired) to find more instances of the relation, marks good/bad pairs, and expands the table (Figure 2). By repeating this process, the user rapidly populates the table.

2.4 Execution Speed

IKE uses BlackLab (Institute Dutch Lexicology, 2016) for indexing the corpus. This, combined with the chunk-based representation, results in fast query execution times (e.g., <1 second for a query over 1.5M sentences), an essential requirement for an interactive system (Table 2).
**Figure 2:** Search for examples of “X absorbs Y”, where X is distributionally similar (~500) to existing entries in the Material column of the conducts table. The user then annotates examples for inclusion in the table.

<table>
<thead>
<tr>
<th>Corpus</th>
<th># sentences</th>
<th>Avg. query-time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Science textbook</td>
<td>1.2K</td>
<td>0.253</td>
</tr>
<tr>
<td>ck2.org texts</td>
<td>17K</td>
<td>0.286</td>
</tr>
<tr>
<td>SimpleWikipedia</td>
<td>1M</td>
<td>0.530</td>
</tr>
<tr>
<td>Web Subset (small)</td>
<td>1.5M</td>
<td>0.595</td>
</tr>
<tr>
<td>Web Subset (large)</td>
<td>20M</td>
<td>2.809</td>
</tr>
</tbody>
</table>

Table 2: Avg. query-times with different sized corpora.

## 3 Preliminary Evaluation

### 3.1 Experiments

Although IKE is still under development, we have conducted a preliminary evaluation, comparing it with two other methods for populating relation tables. Our interest is in how these different methods compare in terms of precision, yield and time:

- **Manual**: The user manually authors and refines patterns (without any automatic assistance) to populate a table.
- **Automatic**: The user provides an initial table with a few entries, and then lets the system bootstrap on its own, without any further interaction.
- **Interactive (IKE)**: Interactive bootstrapping, as described earlier.

The manual system was implemented in IKE by disabling the embedding-based set expansion and ML-based query suggestion features. The automatic approach was simulated in IKE by removing both user annotation steps in Figure 1 and instead adding all machine-learned patterns suggested by the Query Suggestor (Section 2.2) and instances that occur at least $k$ times in the corpus (using $k = 2$). This is a simple baseline method of bootstrapping, compared with more sophisticated methods such as co-training (Collins and Singer, 1999; Neelakantan and Collins, 2014). For a fair comparison, we compared results after 3 bootstrapping iterations (for Automatic, IKE) and a similar amount of user time (~30 mins, for Manual and IKE). The number of iterations were limited to 3 to keep the annotation time within reasonable limits.

### 3.2 Tasks and Datasets

We compared these methods to define and populate two target relations: **conducts**(material, energy), and **has-part**(animal, bodypart). All methods extract knowledge from the same corpora of science text, consisting of ~1.5M sentences (largely) about elementary science drawn from science textbooks,
Simple Wikipedia, and the Web. For each relation, two (different) users familiar with IKE were asked to construct these tables. The numbers presented in Table 3 and Table 4 are averaged over these two users. Although this study is small, it provides helpful indicators about IKE’s utility.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acquired Patterns</th>
<th>Extractions</th>
<th>Time in min.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of patterns</td>
<td>Average Precision</td>
<td>Number (total)</td>
</tr>
<tr>
<td>Manual</td>
<td>7</td>
<td>25.5</td>
<td>106</td>
</tr>
<tr>
<td>Automatic</td>
<td>108</td>
<td>34.4</td>
<td>183</td>
</tr>
<tr>
<td>IKE</td>
<td>31</td>
<td>52.7</td>
<td>112</td>
</tr>
</tbody>
</table>

Table 3: conducts(material,energy) table after 3 iterations or ~30 minutes user time. IKE helps the user discover substantially more patterns than the manual method (31 vs. 7), with better precision and in less time, resulting in the overall yield of 59 relation instances. Fully automatic bootstrapping produced a large number of lower precision (34.4%) patterns compared to IKE (52.7%) patterns.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acquired Patterns</th>
<th>Extractions</th>
<th>Time in min.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of patterns</td>
<td>Average Precision</td>
<td>Number (total)</td>
</tr>
<tr>
<td>Manual</td>
<td>16</td>
<td>25.2</td>
<td>290</td>
</tr>
<tr>
<td>Automatic</td>
<td>228</td>
<td>3.5</td>
<td>1386</td>
</tr>
<tr>
<td>IKE</td>
<td>21</td>
<td>22.5</td>
<td>449</td>
</tr>
</tbody>
</table>

Table 4: has-part(organism, bodypart) table after 3 iterations or ~30 minutes user time. Again, IKE produces the highest yield by helping the user discover 21 patterns with precision (22.5%) comparable to manual patterns (25.2%). Note that further use of IKE continues to expand the yield (e.g., after 3 more iterations of IKE the yield rises to 262 while maintaining average precision).

3.3 Results

Table 3 shows the results for building the conducts(material,energy) table. Most importantly, with IKE the user was able to discover substantially more patterns (31 vs. 7) with higher accuracy (52.7% vs. 25.5%) than the manual approach, resulting in a larger table (59 vs. 27 rows) in less time (20 vs. 30 mins). It also shows that fully automatic bootstrapping produced a large number of low quality (34.4% precision) rules, with an overall lower yield (63 rows).

Note that for both Manual and IKE, users have to decide how to spend their time budget, in particular between work on creating high-quality patterns vs. work on annotating examples found by those patterns. Thus the precision scores in the Tables reflect how the users chose to make this tradeoff, while the yield reflects their success at the overall goal, namely building a good table.

Table 4 shows similar results for constructing the has-part(organism, bodypart) table, IKE having the highest overall yield. Although this is a small case study, it suggests that IKE has value for rapid knowledge base construction.

4 Conclusion

We have presented IKE, a usable, general-purpose tool for interactive extraction. It has an expressive, easily comprehensible query language that integrates symbolic (boolean) and distributional (similarity-based) methods for search, and has a fast execution time. A preliminary evaluation suggests that IKE is effective for the task of knowledge-base construction compared to manual pattern authoring or using fully automated extraction tools. We are currently using this tool to expand the KB used by the Aristo system (Clark et al., 2016), and are making IKE publically available on our Web site at http://allenai.org/software/interactive-knowledge-extraction.

Acknowledgments

We are grateful to Paul Allen whose long-term vision continues to inspire our scientific endeavors. We would also like to thank Carissa Schoenick and Satwant Rana for their critical contributions to IKE.

References


[Angeli et al.2015] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In ACL.


[Turney2006] Peter D. Turney. 2006. Similarity of semantic relations. COLING.