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 publically 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; Hoffmann 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-

Proceedings of AKBC 2016, pages 12–17,
San Diego, California, June 12-17, 2016.

racy, suggesting IKE has utility for KB construction.

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

($conducts.Material) conducts ($conducts.Energy)
($conducts.Material) absorbs ($conducts.Energy)
($conducts.Material) produces ($conducts.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 **conducts**(*material*,*energy*), e.g., conducts("steel","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 *material* and *energy*

First, the user defines the argument types *material* and *energy*. To define a type, IKE lets a user build a single column table, e.g., for type *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

   $Material ∼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 *material*

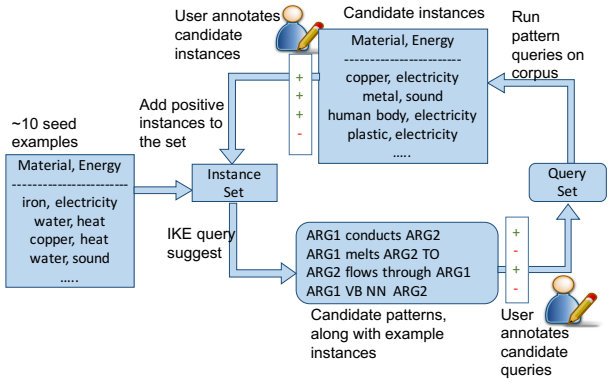

Figure 1: IKE interactive bootstrapping workflow

The process is repeated for the type 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.,:

materials such as (NP)
materials such as $Material and (NP)

or a combination of patterns and word2vec expressions.

2.3.2 Create and Seed the conducts Table

The user next creates a two-column table **conducts**, and then uses a seed pattern to find initial pairs to populate it, e.g., the pattern

($Material) conducts ($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).

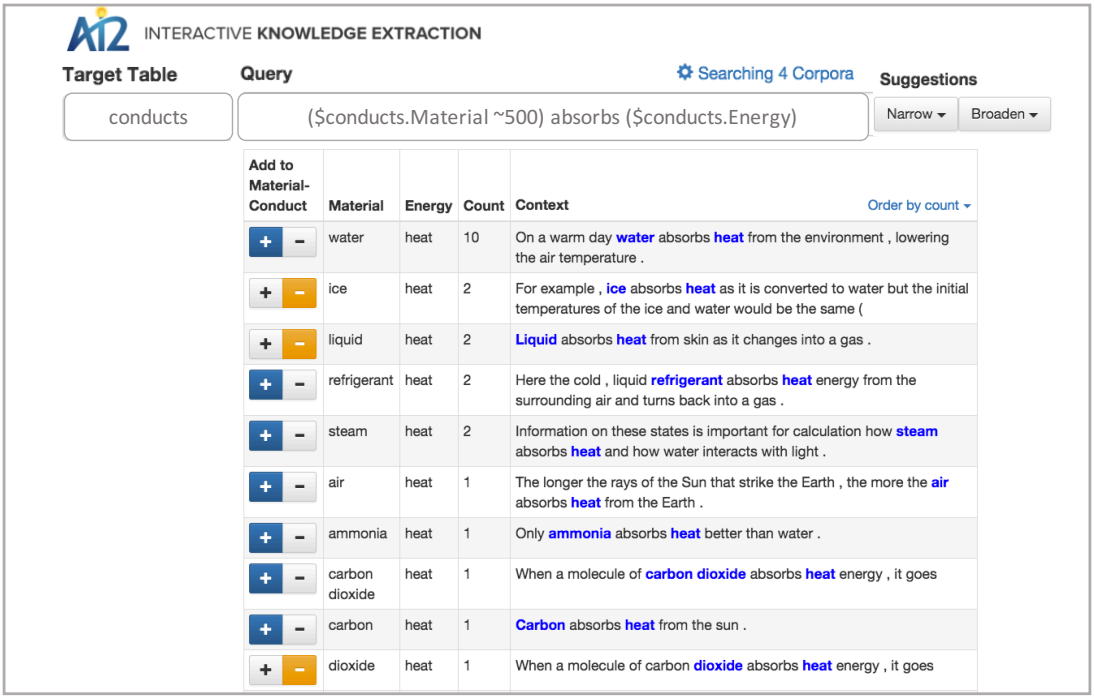

Figure 2: Search for examples of "X absorbs Y", where X is distributionally similar (∼500) to existing entries in the Material column of the **conducts** table. The user then annotates examples for inclusion in the table.

Corpus	# sentences	Avg. query-time (sec.)
Science textbook	1.2K	0.253
ck2.org texts	17K	0.286
SimpleWikipedia	1M	0.530
Web Subset (small)	1.5M	0.595
Web Subset (large)	20M	2.809

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.

Method	Acquired Patterns		Extractions		Time
	No. of patterns	Average Precision	Number (total)	Yield (+ves)	in min.
Manual	7	25.5	106	27	30
Automatic	108	34.4	183	63	-
IKE	31	**52.7**	112	**59**	20

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.

Method	Acquired Patterns		Extractions		Time
	No. of patterns	Average Precision	Number (total)	Yield (+ves)	in min.
Manual	16	25.2	290	73	35
Automatic	228	3.5	1386	48	-
IKE	21	**22.5**	449	**101**	30

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

[Akbik et al.2013] Alan Akbik, Oresti Konomi, and Michail Melnikov. 2013. Propminer: A workflow for interactive information extraction and exploration using dependency trees. In *ACL*.

[Amershi et al.2014] Saleema Amershi, Maya Cakmak, W. Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. In *AI Magazine*.

[Amershi et al.2015] Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Y. Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *CHI*.

[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*.

[Cafarella et al.2005] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. 2005. Knowitnow: Fast, scalable information extraction from the web. In *NAACL*.

[Carlson et al.2009] Andrew Carlson, Justin Betteridge, Estevam R Hruschka Jr, and Tom M Mitchell. 2009. Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, pages 1–9. Association for Computational Linguistics.

[Carlson et al.2010] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

[Chang and Manning2014] Angel X Chang and Christopher D Manning. 2014. Tokensregex: Defining cascaded regular expressions over tokens. Technical Report CSTR 2014-02, Stanford University.

[Clark et al.2016] Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proc. IJCAI'16*.

[Collins and Singer1999] Michael John Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *EMNLP&VLC'99*.

[Etzioni et al.2004] Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *WWW*.

[Freedman et al.2011] Marjorie Freedman, Lance A. Ramshaw, Elizabeth Boschee, Ryan Gabbard, Gary Kratkiewicz, Nicolas Ward, and Ralph M. Weischedel. 2011. Extreme extraction - machine reading in a week. In *EMNLP*.

[Gamallo et al.2012] Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18. Association for Computational Linguistics.

[Gormley and Tong2015] Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The Definitive Guide*. " O'Reilly Media, Inc.".

[Gupta and Manning2014] Sonal Gupta and Christopher D. Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CONLL*.

[Hearst1992] Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.

[Hoffmann et al.2011] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

[Hoffmann et al.2015] Raphael Hoffmann, Luke S. Zettlemoyer, and Daniel S. Weld. 2015. Extreme extraction: Only one hour per relation. *CoRR*, abs/1506.06418.

[Institute Dutch Lexicology2016] INL Institute Dutch Lexicology. 2016. Blacklab: A corpus search engine. https://github.com/INL/BlackLab.

[Li et al.2012] Yunyao Li, Laura Chiticariu, Huahai Yang, Frederick Reiss, and Arnaldo Carreno-Fuentes. 2012. Wizie: A best practices guided development environment for information extraction. In *ACL*.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

[Nakashole et al.2011] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. 2011. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 227–236. ACM.

[Neelakantan and Collins2014] Arvind Neelakantan and Michael Collins. 2014. Learning dictionaries for named entity recognition using minimal supervision. In *EACL*.

[Snow et al.2004] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

[Soderland et al.2013] Stephen Soderland, John Gilmer, Robert Bart, Oren Etzioni, and Daniel S. Weld. 2013. Open information extraction to kbp relations in 3 hours. In *TAC*.

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