IKE - An Interactive Tool for Knowledge Extraction

Bhavana Dalvi
Allen Institute for
Artificial Intelligence
bhavanad@allenai.org

Sumithra Bhakthavatsalam
Allen Institute for
Artificial Intelligence
sumithrab@allenai.org

Peter Clark
Allen Institute for
Artificial Intelligence
peterc@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, and provides novel solutions to these usability concerns. In particular, it uses a novel query language that is expressive, easy to understand, and fast to execute - essential requirements for a practical system - and is the first interactive extraction tool to seamlessly integrate symbolic and distributional methods for search. An initial evaluation suggests that relation tables can be populated substantially faster than by manual pattern authoring or using fully automated tools, while retaining accuracy, an important step towards practical knowledge-base construction.

1 Introduction

Knowledge extraction from text remains a fundamental challenge for any system that works with structured data. Automatic extraction algorithms (e.g., [3]) have proved efficient, but typically produce noisy results (e.g., the best F1 score for the KBP slot filling task was 0.28, as reported in [3]). Weakly supervised automatic bootstrapping methods [4, 7] 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 [1, 2]. For example, Soderland et al. [9] showed that users can be surprisingly effective at authoring and refining extraction rules for a slot filling task, and Freedman et al. [6] demonstrated that a combination of machine learning and user authoring produced high quality results. However, none of these approaches have evolved into usable tools.

In this paper we present IKE, a usable, general-purpose tool for interactive extraction. It addresses the competing 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 and distributional methods for search. A preliminary evaluation suggests that relation tables can be populated substantially faster with IKE than by manual pattern authoring or using fully automated extraction tools, while retaining accuracy, suggesting IKE has utility for the task of knowledge-base construction.

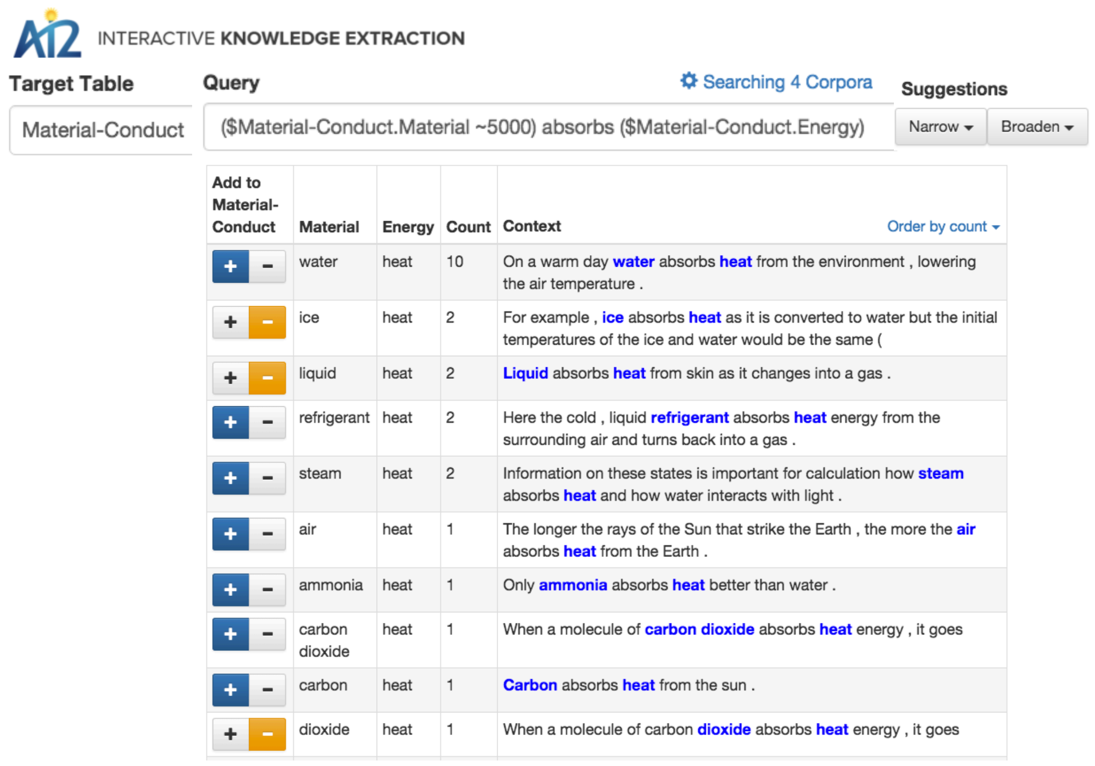
2 Interactive Knowledge Extraction (IKE)

We first describe IKE's architecture, and then 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 Query Language

Central to IKE is the notion that an *extraction pattern* can be treated as a *search query* over a corpus.



INTERACTIVE **KNOWLEDGE EXTRACTION**

Target Table: Material-Conduct

Query: ($Material-Conduct.Material ~5000) absorbs ($Material-Conduct.Energy)

Searching 4 Corpora

Suggestions: Narrow ▾ Broaden ▾

Add to Material-Conduct	Material	Energy	Count	Context (Order by count ▾)
+ −	water	heat	10	On a warm day **water** absorbs **heat** from the environment , lowering the air temperature .
+ −	ice	heat	2	For example , **ice** absorbs **heat** as it is converted to water but the initial temperatures of the ice and water would be the same (
+ −	liquid	heat	2	**Liquid** absorbs **heat** from skin as it changes into a gas .
+ −	refrigerant	heat	2	Here the cold , liquid **refrigerant** absorbs **heat** energy from the surrounding air and turns back into a gas .
+ −	steam	heat	2	Information on these states is important for calculation how **steam** absorbs **heat** and how water interacts with light .
+ −	air	heat	1	The longer the rays of the Sun that strike the Earth , the more the **air** absorbs **heat** from the Earth .
+ −	ammonia	heat	1	Only **ammonia** absorbs **heat** better than water .
+ −	carbon dioxide	heat	1	When a molecule of **carbon dioxide** absorbs **heat** energy , it goes
+ −	carbon	heat	1	**Carbon** absorbs **heat** from the sun .
+ −	dioxide	heat	1	When a molecule of carbon **dioxide** absorbs **heat** energy , it goes

Figure 1: Use of set expansion to find and select vector-similar phrases in the corpus.

To make this operational, 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 parse-based representation of the text corpus. The query language supports wildcards, window sizes, POS tags, chunk tags, as well as reference to other predicate tables already constructed. In particular, "capture groups", indicated by parentheses, instruct IKE to catch the matching element(s) as candidate entries in the table being populated. The query language is presented by example in Table1.

2.2 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. The user's workflow can be summarized as follows:

1. Define the types *material* and *energy*
2. Create a two-column table for **conducts**

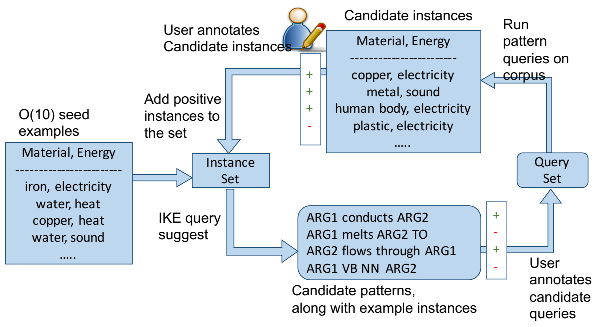


Figure 2: IKE interactive bootstrapping work-flow to create Material-Conductivity table

3. Use a seed pattern, e.g., **X:***material* **conducts Y:***energy*, to populate the **conducts** table with pairs (X,Y)
4. Repeat:
   (a) Use the ML-based query suggestor to find additional patterns relating the Xs and Ys, i.e., also expressing **conducts**.
   (b) Use those additional patterns to further populate **conducts**

This is the typical bootstrapping algorithm, here

Query	Interpretation
the dog	matches "the" followed by "dog"
"brown dog"	matches the phrase "brown dog"
NP grows	an NP followed by "grows"
(NP) grows	Capture NP as an entry in the table's column 1
(NP) conducts (NP)	Capture both NPs as arg1, arg2
(?<myGroup> NP) grows	Name the capture group "myGroup"
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 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

with a user in the loop. We now describe each of these steps.

2.2.1 Define the types *material* and *energy*

To define a type, IKE lets a user build a single column table. For example, 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 (Figure 1). The syntax **$material** $\sim$**20** denotes the 20 phrases most similar to any existing member in the *material* table, where similar is defined as the cosine vector between the phrase embeddings. The parentheses "()" tells IKE that each result is a candidate new member for this table.
4. Repeat step 3 until the table adequately characterizes the intended notion of *material*

The process is repeated for the type Energy.

2.2.2 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.

2.2.3 Bootstrapping to Expand the Table

The user can now bootstrap by invoking the Query Suggestor to find additional patterns (queries) that express the target relation. It does so by searching for narrowings or broadenings of the current query that cover a large number of positive pairs and a few negative pairs in the table so far. The user then clicks on one of these patterns to select and execute it (possibly with edits if desired) to find more instances of the relation, marks good/bad pairs, and expands the table. This process can then be repeated.

Narrowing involves searching the space of restrictions on the current query, e.g., replacing a POS tag with a specific word, whereas 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 based on the number of positive, negative, and unlabeled instances it produces. Although the suggested queries can sometimes look complicated with POS and chunk tags within them, the user only has to understand them, not author them from scratch. Hence the required skill set and data analysis workload on the user is significantly reduced. The system still gives control to the user by letting him/her filter/edit the queries, and annotate the extractions in each iteration.

By repeating this process, the user rapidly populates the table with positive examples. Negative examples are also recorded by IKE for use by the Query Suggestor.

2.3 Execution Speed

IKE uses BlackLab [8] for indexing the corpus. This, combined with the chunk-based representation, results in very fast query execution times (e.g., $<$1 second for a query over 10M sentences), an essential requirement for an interactive system (Table 2).

Corpus	# sentences	Avg. query-time (sec.)
Barrons	1.2K	0.2533
CK12	17K	0.2862
SimpleWikipedia	1M	0.5296
Web Subset (small)	1.5M	0.5953
Web Subset (large)	20M	2.809

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

3 Experimental Evaluation

3.1 Experiments

Although IKE is still under development, we have conducted a small, 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. The methods compared were:

- *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 query suggestion features. The automatic approach was simulated in IKE by removing both user annotation steps in Figure 2.2, and instead adding all patterns and instances that occur at least k times in the corpus (using $k = 2$).

3.2 Tasks and Datasets

We compared these methods to define and populate two target relations: **has-part**(*animal*,*bodypart*), and **conducts**(*material*,*energy*). All methods extract knowledge from the same corpora of science text, consisting of $\sim$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. Although this study is small and preliminary, it provides some helpful indicators about the IKE's usefulness.

3.3 Results

Method	Acquired Patterns		Extractions		Time in min.
	No. of patterns	Average Precision	Number (total)	Yield (+ves)	
Manual	5	66.7	24	16	30
Automatic	108	34.4	183	63	-
IKE	41	59.2	142	**84**	20

Table 3: conducts(*material*,*energy*) table: IKE helps the user discover substantially more patterns than the manual method (41 vs. 5), with similar precision and in less time, resulting in the overall highest yield. Fully automatic bootstrapping produced a large number of low precision patterns and overall lower yield than IKE.

Method	Acquired Patterns		Extractions		Time in min.
	No. of patterns	Average Precision	Number (total)	Yield (+ves)	
Manual	11	21.0	291	61	40
Automatic	228	3.4	1386	48	-
IKE	21	57.4	249	**143**	30

Table 4: has-part(*organism*,*bodypart*) table: Again, IKE produces the highest yield by helping the user discover substantially more (21) high-precision (57.4%) patterns.

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 (41 vs. 5) with similar accuracy (59.2% vs. 66.7%) than the manual approach, resulting in a larger table (84 vs. 16 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). Table 4 shows similar results for constructing the **has-part**(*organism*,*bodypart*) table, again IKE having the highest overall yield. Although this is a small study, it suggests that IKE has value for rapid knowledge base construction.

3.4 Future Development

IKE is still under development, with several areas needing further investigation. ... **elaborate**

4 Conclusion

We have presented IKE, a tool that performs fast, interactive bootstrapping to develop high quality extraction patterns for targeted relations. It is an example of a Machine Teaching approach to knowledge base construction, where the focus is on leveraging the user to rapidly guide the system to good results. The preliminary evaluation comparing with manual and automated approaches, although small, is encouraging as it suggests that IKE makes the best of worlds by improving recall compared with a manual process while retaining precision within acceptable bounds. We are currently using IKE to expand the collection of tables used by the Aristo system [5].

Acknowledgments:

We thank Tony Fader, Dirk Groeneveld, Oren Etzioni, and Chris Clark for their critical contributions to IKE.

References

[1] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35:105–120, 2014.

[2] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Y. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *CHI*, 2015.

[3] G. Angeli, M. J. J. Premkumar, and C. D. Manning. Leveraging linguistic structure for open domain information extraction. In *ACL*, 2015.

[4] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell. 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, 2010.

[5] P. Clark, O. Etzioni, T. Khot, A. Sabharwal, O. Tafjord, P. Turney, and D. Khashabi. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proc. IJCAI'16*, 2016.

[6] M. Freedman, L. A. Ramshaw, E. Boschee, R. Gabbard, G. Kratkiewicz, N. Ward, and R. M. Weischedel. Extreme extraction - machine reading in a week. In *EMNLP*, 2011.

[7] S. Gupta and C. D. Manning. Improved pattern learning for bootstrapped entity extraction. In *CONLL*, 2014.

[8] I. of Dutch Lexicology (INL). Blacklab; a corpus search engine. `https://github.com/INL/BlackLab`.

[9] S. Soderland, J. Gilmer, R. Bart, O. Etzioni, and D. S. Weld. Open information extraction to kbp relations in 3 hours. In *TAC*, 2013.