

Entity Extraction Enables “Discovery”

By Steven Cohen, Vice President, Products | Basis Technology

A discovery search is one in which you don’t know, or can’t know, all relevant search terms. Automated entity extraction lets you discover what you don’t know.

Consider three scenarios: 1.) A real estate agent wants to search Craig’s List for the prices of all two-bedroom condominiums in Boston over 1200 square feet with two bathrooms and indoor parking. 2.) An intelligence analyst wants to search the world’s newspapers for the names, locations, and dates associated with a certain diplomat’s visit to the U.S. 3.) An industrial designer for a leading automaker wants to search the thousands of customer emails received last year for all references to the color of the company’s hot-selling new model.

The usefulness of these searches is clear. Yet, each is impossible or impractical with today’s web and enterprise search engines. That’s because conventional search technology does not enable “discovery search.” What is discovery search? A discovery search is a search in which you are looking for something you cannot *fully specify* until you find it. You don’t know all the search terms that will return all instances of the desired information. And even if you did, you probably would not have time to type them all into a search box.

Take the real estate scenario. There are countless ways sellers describe the attributes of properties. Yet a person reading these listings usually has no problem parsing the sentences and extracting the key “entities” of interest - price, square footage, the presence of certain amenities, and so on. Whether bathroom is spelled out, abbreviated, shortened to “bath,” or preceded by a number or a word - to the buyer the meaning is obvious. It’s impossible for anyone to know, never mind manually type, all the possible combinations search terms that would find every qualified apartment regardless of how its classified ad were written.

The same applies to the other two scenarios. How would you specify a search for all the places a diplomat *might* have visited? Or how could you search for every possible occurrence of colors - especially since new words for colors are invented all the time?

You do it by using a technology called **entity extraction**. Entity extraction automatically locates important search terms for you based on the same contextual cues people would use. The ability to locate these entities - units of meaning - in any text is what enables discovery search to find all the information that fits a definition even when you don’t know all the search terms the definition fits. This is something the current generation of search technologies does not do.

Two-Dimensional Search

First generation search tools look for information in basically two ways: by word matching and by link analysis:

Mid 1990s / Alta Vista / Keyword Lookup

The very early search tools, of which Alta Vista is the model example, looked for words (keywords) in documents that match the words (search terms) the user types into the search box. Relevance is based on the document's internal content attributes - for example, the document title, the number of times the sought-after string appears in the document, how close to the beginning of the document the terms appear, and the proximity of words to each other in a multi-word search term.

Late 1990s / Google / Social Context

Rather than just look at a document in isolation to determine search relevance, Google and most other web search engines today also look at how many other documents link to it. If more people link to a document, the document must be more important, and hence more relevant to more searches.

Late 1990s— Early 2000s / Ask Jeeves / Topic Communities

Ask.com (formerly Ask Jeeves) was first to apply machine language processing to semantic content - specifically that a user is asking a question. Where other search engines ignored words like "Why," or "What" because they appear too often everywhere to have relevance anywhere, Ask regards such words as question markers. Another technique Ask uses is "dynamic topic clustering." This weighs both the raw number of inbound links, like Google, and the number of links from sites with a high degree of relevant subject content *that also link to each other*. If a site is highly referenced within a "topic community," it is probably better suited to answer a question on that topic.

Discover: the Next Generation in Search

None of these methods, however, let you search on meaning rather than on words. If you were looking for information on red cars, for example, even the technology that powers Ask.com would only return answers from sources explicitly about the word *red*. It would not return answers about *burgundy* or *maroon* unless those sources included the word *red* as well. That might pose a problem for, say, a car designer looking for the most popular color combinations mentioned in a year's worth of customer emails.

Adding the word *color* to the search string as a workaround might improve results, but again, not all red-related content would include the word *color* either. Another issue is the fact that the word *red* has meanings outside the context of design, especially as a metaphor for danger, as in *red zone*. A designer might want to exclude those.

Bush is an even richer example of a word whose meaning is different depending on context. A search on that word might return articles about President Bush as well as articles about landscaping. A journalist looking for documents about President Bush would want to specify the *person* attribute in the search.

What is missing in these scenarios is the ability to search on a term's semantic attributes. The computer cannot automatically discover all the "entities" - i.e., concepts expressed as words or phrases - that have those attributes. With first generation search, you must already know all the right keywords to type in the search box. You must also have taken into account all the semantic misinterpretations (as in other kinds of bushes). And you would also need the skill to express terms so as to exclude those misinterpretations.

This ability to automatically discover semantic matches and exclude semantic non-matches is what entity extraction is all about.

How Entity Extraction Works

Words derive much of their meanings from the visual context in which they appear on a page. The same word will often have very different meanings depending on various visual cues. Similarly, the same meaning will often be expressed by very different words, also depending on visual cues. Many cues also work both visually and verbally. These cues, or contextual features, include:

- Proximity to other words
- Written forms of the word (e.g., abbreviations, capitalization)
- Parts of speech (e.g., is the word used as a subject, predicate, object, etc.)
- Punctuation
- And many more ...

To form a concept from a word, a person selects one of several possible definitions for the word, depending on the word's context. Entities are the *representations* that express on the page the concept in people's heads.

Suppose, for example, that you wanted to find all the names of people in a document - *all* names, even those of people you might not have previously considered. Suppose also that the document contains the quote: "Mr. John Xyzzy spoke..." Even though you may never have seen the name Xyzzy before, you would infer that Xyzzy is a person's name. You would base that inference on that fact that the word is capitalized and otherwise matches a pattern: "Mr. [capitalized noun] [capitalized noun] [verb]..."

Entity extraction recreates in a computer the process of applying and recognizing context. Take the word *date*. When that word appears in a block of text, is it an entity with one set of contextual features - as in a *point in time* - or it is a completely different set of contextual features - which add up to *food*? There are literally hundreds of different features that indicate when the word *date* is more about time or about food.

Linguists and computational linguists have used context for decades to create natural language processing software. The barrier to automating entity extraction has been how to efficiently teach computers to recognize entities correctly. Manually programming the rules to do this is impractical - there are just too many subtle variations in how people attribute meaning to a word in all possible contexts. Some features are obvious but many are not. It would be virtually impossible to specify all of them explicitly as rules for a computer to follow in every text that a computer might encounter in real-world applications.

Rather than program computers to match contexts, an alternative approach has been to essentially teach computers the language itself. In other words, to teach computers the grammatical and morphological rules of English, French, Chinese, or whatever natural language is to be parsed. Once the computer “understood” a language, it could presumably recognize within sampled text the instances of entities a user might specify. A natural language parser, however, has two main drawbacks:

- Natural languages are enormously difficult to express as explicit rule sets - so a parser would take years to construct
- Every language has a different set of rules - so the parser built for one language would not work for another

A third approach - different from either programming computers to match contexts explicitly or constructing natural language parsers - is statistical machine learning.

A statistical approach uses a three-step process that can be completed in a few weeks versus the years the other approaches might take just to extract even one entity (*color*, for example). Those three steps are:

1. A computational linguist specifies an entity’s contextual features
2. Natural language speakers tag a statistically sufficient number of examples of the entity in sampled text (typically thousands)
3. The features and tagged examples are fed into a computer, which generates a model that will recognize instances of the entity when presented with text not previously sampled

Statistical machine learning makes entity extraction practical and offers several key benefits:

Language independence The software algorithms are the same regardless of language or script, the only difference being the entity-specific context features and tagged samples supplied.

Entity extensibility Adding new features and examples to an existing model is straightforward - so adding new entities is relatively easy and so too is accommodating changes to the language - for example, as new jargon appears.

Automated Discovery

The model will return new instances of an entity that fit the context, even if they didn't exist when the entity was first modeled. Searchers don't have to know what they don't know (or don't know yet) to define an inclusive search.

The Impact

Statistically powered entity extraction distinguishes the newest generation of search from earlier generations. People will no longer have to rely on their personal knowledge (or simple brute force) to find all the instances of the information they wish to find in unstructured content. But people are not the only candidates for using this technology. Adding entity extraction to XML tagging tools, for example, provides an efficient way to bring the mountains of unstructured data - both enterprise and web - into a database where it can be manipulated and processed just like structured data can.

In fact, in the very near future the very idea of "unstructured data" as a separate category may itself seem quaint. We are looking at a new kind of convergence, one just as important as the convergence between data and communications. When that happens, it will be because computers can identify what words mean, not just what they look like.



By Steven Cohen
Vice President, Products

Steve co-founded Basis Technology with Carl Hoffman in 1995. He is responsible for the planning and operations of Basis Technology's linguistic product research and development. His previous roles at Basis Technology included VP of Technical Services and VP of Operations. Prior to starting Basis Technology, Steve was the engineering manager for Cognex Corporation's Tokyo office and development manager for SMT device Inspection. He has consulted on software internationalization for Foxboro Corporation and started his career as an embedded systems engineer at Teradyne. Steve earned a bachelor's degree in Electrical Engineering from MIT and studied at Waseda University in Tokyo.

Steve can be reached at
stevec@basistech.com

Basis Technology Corporation

T 1.617.386.2000
1.800.697.2062 (toll-free)
F 617.386.2020
E info@basistech.com