

The Linguist’s Search Engine: An Overview

Philip Resnik	Aaron Elkins
Department of Linguistics and UMIACS	UMIACS
University of Maryland	University of Maryland
College Park, MD 20742	College Park, MD 20742
resnik@umd.edu	aelkiss@umiacs.umd.edu

Abstract

The Linguist’s Search Engine (LSE) was designed to provide an intuitive, easy-to-use interface that enables language researchers to seek linguistically interesting examples on the Web, based on syntactic and lexical criteria. We briefly describe its user interface and architecture, as well as recent developments that include LSE search capabilities for Chinese.

1 Introduction

The idea for the Linguist’s Search Engine originated in a simple frustration shared by many people who study language: the fact that so much of the argumentation in linguistic theory is based on subjective judgments. Who among us has not, in some talk or class, heard an argument based on a “starred” (deemed-ungrammatical) example, and whispered to someone nearby, *Did that sound ok to you?* because we thought it sounded fine? As Bard et al. (1996) put it, each linguistic judgment is a “small and imperfect experiment”. Schütze (1996) and Cowart (1997) provide detailed discussion of instability and unreliability in such informal methods, which can lead to biased or even misleading results.

Recent work on linguistics methodology draws on the perception literature in psychology to provide principled methods for eliciting gradient, rather than discrete, linguistic judgments (Sorace and Keller, 2005). In addition, at least as far back

as Rich Pito’s 1992 *tgrep*, distributed with the Penn Treebank, computationally sophisticated linguists have had the option of looking at naturally occurring data rather than relying on constructed sentences and introspective judgments (e.g., Christ, 1994; Corley et al., 2001; Blaheta, 2002; Kehoe and Renouf 2002; König and Lezius, 2002; Fletcher 2002; Kilgarriff 2003). Unfortunately, many linguists are unwilling to invest in psycholinguistic methods, or in the computational skills necessary for working with corpus search tools. A variety of people interested in language have moved in the direction of using Web search engines such as Google as a source of naturally occurring data, but conventional search engines do not provide the mechanisms needed to perform many of the simplest linguistically informed searches – e.g., seeking instances of a particular verb used only intransitively.

The Linguist’s Search Engine (LSE) was designed to provide the broadest possible range of users with an intuitive, linguistically sophisticated but user-friendly way to search the Web for naturally occurring data. Section 2 lays out the LSE’s basic interface concepts via several illustrative examples. Section 3 discusses its architecture and implementation. Section 4 discusses the current status of the LSE and recent developments.

2 LSE Interface Concepts

The design of the LSE was guided by a simple basic premise: a tool can’t be a success unless people use it. This led to the following principles in its design:

- Minimize learning/ramp-up time.
- Have a linguist-friendly look and feel.
- Permit rapid interaction.
- Permit large-scale searches.
- Allow searches using linguistic criteria.

Some of these principles conflict with each other. For example, sophisticated searches are difficult to specify in a linguist-friendly way and without requiring some learning by the user, and rapid interaction is difficult to accomplish for Web-sized searches.

2.1 Query By Example

The LSE adopts a strategy one can call “query by example,” in order to provide sophisticated search functionality without requiring the user to learn a complex query language. For example, consider the so-called “comparative correlative” construction (Culicover and Jackendoff, 1999). Typing *the bigger the house the richer the buyer* automatically produces the analysis in Figure 1, which can be edited with a few mouse clicks to get the generalized structure in Figure 2, converted with one button push into the LSE’s query language, and then submitted in order to find other examples of this construction, such as *The higher the rating, the lower the interest rate that must be paid to investors; The more you bingo, the more chances you have in the drawing; The more we plan and prepare, the easier the transition.*

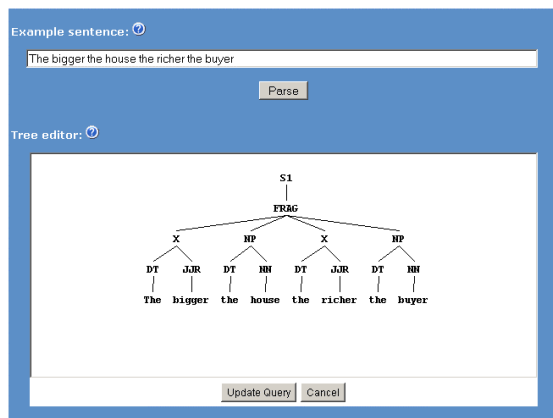


Figure 1. Querying by example

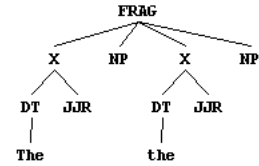


Figure 2. Generalized query

Crucially, users need not learn a query language, although advanced users can edit or create queries directly if so desired. Nor do users need to agree with (or even understand) the LSE’s automatic parse, in order to find sentences with parses similar to the exemplar. Indeed, as is the case in Figure 1, the parse need not even be entirely reasonable; what is important is that the structure produced when analyzing the query will be the *same* structure produced via analysis of the corresponding sentences in the corpus.

Other search features include the ability to specify immediate versus non-immediate dominance; the ability to negate relationships (e.g. a VP that does *not* immediately dominate an NP); the ability to specify that words should match on all morphological forms; the ability to match nodes based on WordNet relationships (e.g. all descendants of a particular word sense); the ability to save and reload queries; the ability to download results in keyword-in-context (KWIC) format; and the ability to apply a simple keyword-based filter to avoid offensive results during live demonstrations.

Results are typically returned by the LSE within a few seconds, in a simple search-engine style format. In addition, however, the user has rapid access to the immediate preceding and following contexts of returned sentences, their annotations, and the Web page where the example occurred.

2.2 Built-In and Custom Collections

Linguistically annotating and indexing the entire Web is beyond impractical, and therefore there is a clear tradeoff between rapid response time and the ability to search the Web as a whole. In order to manage this tradeoff, the LSE provides, by default, a built-in collection of English sentences taken randomly from a Web-scale crawl at the Internet

Archive.¹ This static collection is often useful by itself.

In order to truly search the entire Web, the LSE permits users to define their own custom collections, piggybacking on commercial Web search engines. Consider, as an example, a search involving the verb *titrate*, which is rare enough that it occurs only twice in a collection of millions of sentences. Using the LSE's "Build Custom Collection" functionality, the user can specify that the LSE should:

- Query Altavista to find pages containing any morphological form of *titrate*
- Extract only sentences containing that verb
- Annotate and index those sentences
- Augment the collection by iterating this process with different specifications

Doing the Altavista query and extracting, parsing, and indexing the sentences can take some time, but the LSE permits the user to begin searching his or her custom collection as soon as *any* sentences have been added into it. Typically dozens to hundreds of sentences are available within a few minutes, and a typical custom collection, containing thousands or tens of thousands of sentences, is completed within a few hours. Collections can be named, saved, augmented, and deleted.

Currently the LSE supports custom collections built using searches on Altavista and Microsoft's MSN Search. It is interesting to note that the search engines' capabilities can be used to create custom collections based on extralinguistic criteria; for example, specifying pages originating only in the *.uk* domain in order to increase the likelihood of finding British usages, or specifying additional query terms in order to bias the collection toward particular topics or domains.

3 Architecture and Implementation

The LSE's design can be broken into the following high level components:

- User interface
- Search engine interface
- NLP annotation
- Indexing
- Search

The design is centered on a relational database that maintains information about users, collections, documents, and sentences, and the implementation combines custom-written code with significant use of off-the-shelf packages. The interface with commercial search engines is accomplished straightforwardly by use of the WWW::Search perl module (currently using a custom-written variant for MSN Search).

Natural language annotation is accomplished via a parallel, database-centric annotation architecture (Elkiss, 2003). A configuration specification identifies dependencies between annotation tasks (e.g. tokenization as a prerequisite to part-of-speech tagging). After documents are processed to handle markup and identify sentence boundaries, individual sentences are loaded into a central database that holds annotations, as well as information about which sentences remain to be annotated. Crucially, sentences can be annotated in parallel by task processes residing on distributed nodes.

Indexing and search of annotations is informed by the recent literature on semistructured data. However, linguistic databases are unlike most typical semistructured data sets (e.g., sets of XML documents) in a number of respects – these include the fact that the dataset has a very large schema (tens of millions of distinct paths from root node to terminal symbols), long path lengths, a need for efficient handling of queries containing wildcards, and a requirement that all valid results be retrieved. On the other hand, in this application incremental updating is not a requirement, and neither is 100% precision: results can be overgenerated and then filtered using a less efficient comparison tools such as *tgrep2*. Currently the indexing scheme follows ViST (Wang et al., 2003), an approach based on suffix trees that indexes structure and content together. The variant implemented in the LSE ignores insufficiently selective query branches, and achieves more efficient search by modifying the ordering within the structural index, creating an in-memory tree for the query, ordering processing of

¹ The built-in LSE Web collection contains 3 million sentences at the time of this writing. We estimate that it can be increased by an order of magnitude without seriously degrading response time, and we expect to do so by the time of the demonstration.

query branches from most to least selective, and memoizing query subtree matches.

4 Status and Recent Developments

The LSE “went live” on January 20, 2004 and approximately 1000 people have registered and tried at least one query. In response to a recent survey, several dozen LSE users reported having tried it more than casually, and there are a dozen or so reports of the LSE having proven useful in real work, either for research or as a tool that was useful in teaching. Resnik et al. (2005) describe two pieces of mainstream linguistics research – one in psycholinguistics and one in theoretical syntax – in which the LSE played a pivotal role.

The LSE software is currently being documented and packaged up, for an intended open-source release.² In addition to continuing linguistic research with the LSE, we are also experimenting with alternative indexing/search schemes. Finally, we are engaged in a project adapting the LSE for use in language pedagogy – specifically, as a tool assisting language teaching specialists in creating training and testing materials for learners of Chinese. For that purpose, we are experimenting with a built-in collection of Chinese Web documents that includes links to their English translations (Resnik and Smith, 2003).

Acknowledgments

This work has received support from the National Science Foundation under ITR grant IIS01130641, and from the Center for the Advanced Study of Language under TTO32. The authors are grateful to Christiane Fellbaum and Mari Broman Olsen for collaboration and discussions; to Rafi Khan, Saurabh Khandelwal, Jesse Metcalf-Burton, G. Craig Murray, Usama Soltan, and James Wren for their contributions to LSE development; and to Doug Rohde, Eugene Charniak, Adwait Ratnaparkhi, Dekang Lin, UPenn’s XTAG group, Princeton’s WordNet project, and untold others for software components used in this work.

References

Bard, E.G., Robertson, D. and A. Sorace. Magnitude estimation of linguistic acceptability. *Language* 72.1: 32-68, 1996.

Christ, Oli. A modular and flexible architecture for an integrated corpus query system, COMPLEX’94, Budapest, 1994.

Corley, Steffan, Martin Corley, Frank Keller, Matthew W. Crocker, and Shari Trewin. Finding Syntactic Structure in Unparsed Corpora: The Gsearch Corpus Query System, *Computers and the Humanities*, 35:2, 81-94, 2001.

Cowart, Wayne. *Experimental Syntax: Applying Objective Methods to Sentence Judgments*, Sage Publications, Thousand Oaks, CA, 1997.

Culicover, Peter and Ray Jackendoff. The view from the periphery: the English comparative correlative. *Linguistic Inquiry* 30:543-71, 1999.

Elkiss, Aaron. A Scalable Architecture for Linguistic Annotation. Computer Science Undergraduate Honors Thesis. University of Maryland. May 2003.

Fletcher, William. Making the Web More Useful as a Source for Linguistic Corpora, North American Symposium on Corpus Linguistics, 2002.

Kehoe, Andrew and Antoinette Renouf, WebCorp: Applying the Web to linguistics and linguistics to the Web, in *Proceedings of WWW2002*, Honolulu, Hawaii, 7-11 May 2002.

Adam Kilgarriff, Roger Evans, Rob Koeling, David Tugwell. WASPBENCH: a lexicographer’s workbench incorporating state-of-the-art word sense disambiguation. *Proceedings of EACL 2003*, 211-214, 2003.

Koenig, Esther and Lezius, Wolfgang, A description language for syntactically annotated corpora. In: *Proceedings of the COLING Conference*, pp. 1056-1060, Saarbruecken, Germany, 2002.

Schuetze, Carson. *The Empirical Base of Linguistics*, University of Chicago Press, 1996.

Sorace, Antonella and Frank Keller. Gradiance in Linguistic Data. To appear in *Lingua*, 2005.

Philip Resnik and Noah A. Smith, The Web as a Parallel Corpus, *Computational Linguistics* 29(3), pp. 349-380, September 2003.

Philip Resnik, Aaron Elkiss, Ellen Lau, and Heather Taylor. The Web in Theoretical Linguistics Research: Two Case Studies Using the Linguist’s Search Engine. 31st Meeting of the Berkeley Linguistics Society, February 2005.

H Wang, S Park, W Fan, and P Yu. ViST: a dynamic index method for querying XML data by tree structures. ACM SIGMOD 2003. pp. 110-121.

² Documentation maintained at <http://lse.umiacs.umd.edu/>.