Matching needs and resources: How NLP can help theoretical linguistics

Alexis Dimitriadis
Utrecht institute of Linguistics OTS
a.dimitriadis@uu.nl

Abstract

While some linguistic questions pose challenges that could be met by developing and applying NLP techniques, other problems can best be approached with a blend of old-fashioned linguistic investigation and the use of simple, well-established NLP tools. Unfortunately, this means that the NLP component is too simple to be of interest to the computationally-minded, while existing tools are often difficult for the programming novice to use. For NLP to come to the aid of research in theoretical linguistics, a continuing investment of effort is required to bridge the gap. This investment can be made from both sides.

1 Introduction

Linguistics is in its heart an empirical discipline, and the data management and data analysis techniques of computational linguistics could, in principle, be productively brought to bear on descriptive and theoretical questions. That this does not happen as much as it could is, as I understand it, the point of departure for this colloquium. Instead of focusing on exciting research questions that are crying out for fruitful collaboration between theoretical and computational linguists, I want to examine the broader range of ways that NLP know-how could be put to productive use in the domain of theoretical linguistics, and some of the ways that this could come to happen more.

In brief, I believe that the lack of interaction is not simply due to lack of interest, or lack of information, on both sides. Rather, the goals and needs of computational interests are not always served well by catering to the community of theoretical and descriptive linguists, the so-called "Ordinary Working Linguists" with a minimum of computational skills and (equally important) no direct interest in computational questions.

Such linguists could draw a lot of benefit from boring, old-hat NLP tools that computational linguists take for granted: searchable parsed corpora, tools to search large collections of text or compute lexicostatistics, online questionnaire tools for collecting and analyzing speaker judgements, etc. Computational linguists have ready access to a number of wonderful tools of this sort. In fact these are often the building blocks and resources on which new applications at the forefront of NLP are built: Who would build a text summarization system without access to a large corpus of text to practice on?

But such uses of NLP are too simple to be of interest from the computational standpoint. Searching a huge corpus for particular syntactic structures could be invaluable to a syntactician, but making this possible is not interesting to a computational linguist: it's not research anymore. This should not be taken to suggest, however, that computational linguists ought to become more "altruistic." Creating tools targeted to non-technical linguists, even successful tools, can still have drawbacks in the long run.

2 The Linguist's Search Engine

The Linguist's Search Engine (Resnik et al. 2004) is an example of an application created for the benefit of ordinary, non-technical linguists. It allowed users to search the web for a specified syntactic structure. Out of view of the user, the engine first executed an ordinary word-match web search and then parsed the hits and matched against the search structure. The user interface (a java application) allowed the query term to be graphically constructed and refined ("query by example"). The authors' goal was to create a true web application: Easy to launch from a web browser, and easy to use without lengthy user manuals or a complicated command language. While the user interface was innovative, its linguistic function was not: The ap-

Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground, ACL 2010, pages 22–24,
Uppsala, Sweden, 16 July 2010.

plication provided a web interface to a collection of tools that had been assembled to support structured searches. The application stagnated after the end of the project, and ceased working altogether as of April, 2010.

While it was operating, the LSE was used as intended: Resnik et al. report on a number of case studies of users who independently used the search engine to carry out linguistic research. Unfortunately, however, the burden of maintenance turned out to be too great for an application that is of no real continuing interest for a computational linguist.

2.1 The cost of new tools

Complex resources are difficult to create and can be difficult to use. In the world of Language Resources, large corpora are created by the millions of words in various standardized formats, often in conjunction with integrated mega-tools for accessing and managing them. But language resources are geared for institutional clients, can cost a lot of money, and are not acquired or used effectively by individuals without access to dedicated IT support.

At the frontier of NLP, on the other hand, tools don't usually come shrink-wrapped with graphical installers. They often don't come with a graphical interface at all. A new research project may involve a new workflow to be created. Needed corpora will be bought, shared or created as needed. A typical project will involve a jumble of file formats, filters, and workflows that manage text in ad hoc ways until the sought-for result is perfected.

Making such a tool available to someone outside the project, even another computational linguist, is a time-consuming enterprise. Like any complicated body of software, it needs to be documented, encapsulated, and then configured and understood by its new users. This requires a considerable time investment which an NLP lab is willing to undertake, but which is of dubious utility to a theoretical linguist— even one who has the computer skills necessary to undertake it. In brief, the expected amount of use must justify the investment in setting up and learning the system. Tagging, parsing and tree-searching programs are commonplace, but setting up a system for one's own use is a non-trivial exercise. A syntactician looking for a few examples of a rare construction may prefer trial and error on google instead of trying to get a complex system to compile. A syntactician looking for similar data from multiple languages is even less likely to take the plunge, since the benefit derived from a single language is proportionally reduced.

3 Services and interoperability

With the goal of reducing the burden of installing complex resources and getting them to talk to each other, the CLARIN program (Common Language Resources and Technology Infrastructure) is working to establish a cutting edge infrastructure of standards and protocols, which will allow language resources and applications to be utilized remotely, and workflows to be constructed interactively in (hopefully) intuitive ways. The vision is to be able to gain remote access to a language corpus, couple it to a processing application (perhaps an experimental parser using a new syntactic analysis), send the results to yet another application for analysis, etc.

It would be great to have ready access to the tools and resources envisioned for the network. But will it be a platform for development of experimental applications by tomorrow's computational linguists, or will the command line continue to compete with web services as an interface? The answer probably depends on the benefits that CLARIN (and any such framework) will offer to researcher-developers. If adopted, it offers hopes of opening up the computational linguist's toolbox to a wider range of users.

4 Helping ourselves

Wouldn't it be great to have a simple tool for executing simple web searches, converting hits into flat text and compiling the results into a simple corpus? Throw in a tagger, a parser and a search application, and we have the functionality of the Linguist's Search Engine but in several pieces. Tools for most of these tasks are already widely available, but only as part of a complex infrastructure that requires skill and non-trivial time investment to deploy. Other tasks are solved over and over on an ad hoc basis, according to the needs of each NLP project. Until the vision of CLARIN becomes reality, ordinary linguists without access to a team of developers are out of luck.

Still, we need not agree with the perspective (held by Resnik et al. 2005, inter alia) that tools for linguists should be point-and-click and really

easy for an untrained user to figure out. Setting the bar that high greatly shrinks the pool of computational linguists willing to write software for the non-technical masses. The life cycle of the Linguist's Search Engine is a case in point.

Instead, linguists should meet the new technology halfway: As Bird (2006) has argued, no integrated tools can be expected to provide the flexibility needed for the creativity of original research. The NLTK (Natural Language Toolkit) is a more flexible alternative: It is a python library providing a high-level scripting environment for interactive linguistic exploration, with a reasonably small amount of technical skill required. Crucially, the NLTK comes with a very accessible book (Bird et al. 2009) that allows an "ordinary working linguist" to learn how to use the system.

The NLTK will still be beyond the reach of linguists unable, or unwilling, to make the necessary time investment. Is this a big problem? I believe that it should be addressed by persuading linguists (especially junior and future ones) of the benefits of achieving a minimal level of computational competence. The availability of more tools that are usable and installable with a moderate investment in training, time and equipment would encourage linguists to make this kind of investment, and would in the long run decrease the support burden for those technology folks who try to make life easier for non-programming linguists. Conversely, computational linguists would hopefully be encouraged to package their programs in a reasonably accessible format if a growing number of potential users is clamoring for them– and if "packaging" need not mean a complete point-and-click interface.

On the subject of command-line tools, I believe that the obstacle is not with the command line per se (anyone can learn to open a terminal window and type a few symbols), but with the powerful and flexible workflows that the command line makes possible. This is the bread and butter of the computational linguist (and of any programmer), and its benefits could belong to descriptive and theoretical linguists as well.

Theoretical linguistics, of course, also has NLP needs that are anything but trivial. At UiL-OTS there are projects underway to model the acquisition of phonotactic constraints; to improve textual entailments (in a linguistically informative way) by taking into account the contribution of lexical meaning; and others. These and other projects can provide challenges that a computational linguist can be happy to tackle. But for theoretical linguistics to fully benefit from NLP, we theoretical linguists need to pick up more of the tools of the computational linguist.

References

Bird, Steven. 2006. "Linguistic Data Management with the Natural Language Toolkit." Plenary talk at the Annual Meeting of the DGfS, Universität Bielefeld.

Bird, Steven, Ewan Klein, and Edward Loper. 2006. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media.

CLARIN. Common Language Resources and Technology Infrastructure. *http://www.clarin.eu/.*

Resnik, Philip, Aaron Elkiss, Ellen Lau, and Heather Taylor. 2005. "The Web in Theoretical Linguistics Research: Two Case Studies Using the Linguist's Search Engine." *31st Meeting of the Berkeley Linguistics Society,* pp. 265-276.