Evidence for Using Auto-Suggest in Syntactic Search

1st Author Name
Affiliation
Address
e-mail address

2nd Author Name
Affiliation
Address
e-mail address

ABSTRACT
In the humanities and social sciences, scholars often need to characterize the usage of a word or concept. The ability to issue queries over syntactic relations between words is useful in these situations. Previous interfaces for searching over syntactic structures require programming-style queries, but most of these scholars lack programming expertise. Auto-suggest is an alternative, but there is no previous work on what auto-suggest options for syntactic relations should look like. A controlled experiment with 400 participants found that syntactic relations are recognized with higher accuracy when contextual examples are shown, than a baseline of naming the relations alone. This indicates that auto-suggest query interfaces for syntactic search should augment the options with contextual examples.

Author Keywords
Search, Syntax, Grammatical Queries, Digital Humanities, Information Extraction

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
What does X do? How is Y described? Such questions are useful in the humanities and social sciences, when scholars attempt to characterize concepts [12]. In syntactic terms, those questions can be phrased as "What are the verbs of which X is the subject?" "What are the adjectives that modify Y?". In other fields, syntactic queries can be used developing complex patterns for recognizing entities in text, such as medical terms [6, 11], and products and organizations [2].

Most existing interfaces for querying syntactically structured text require complex program-like syntax. For example, the popular Stanford Parser includes Tregex, which as the name suggests, allows for sophisticated regular expression search over syntactic tree structures, and Tsurgeon, which allows for manipulation of the trees extracted with Tregex [10]. The Finite Structure Query tool for querying syntactically annotated corpora requires its queries to be stated in first order logic [8]. In the Corpus Query Language [7], a query is a pattern of attribute-value pairs, where values can include regular expressions containing parse tree nodes and words.

However, one survey found that even though linguists wished to make very technical linguistic queries, 55% of them did not know how to program [15]. According to another survey [4], humanities scholars and social scientists are frequently skeptical of digital tools, because they are often difficult to use. This reduces the likelihood that our target users will be willing or able to use existing structured-query tools for syntactic search.

A related approach is the query-by-example work seen in the past in interfaces to database systems [1]. For instance, the Linguist's Search Engine [13] uses a query-by-example strategy in which a user types in an initial sentence in English, and the system produces a graphical view of a parse tree as output, in addition to a nested LISP expression of the same tree. The user can either click on the tree or modify the LISP expression to generalize the query. According to Shneiderman and Plaisant [14], query-by-example has largely fallen out of favor as a user interface design approach. A downside of QBE is that the user must manipulate an example to arrive at the desired generalization.

At the same time, a related technique, auto-suggest, has become a widely-used approach in search user interfaces with strong support in terms of its usability [5]. A list of selectable options is shown under the search bar, filtered to be relevant as the searcher types. Searchers can recognize and select the option that matches their information need, without having to generate the query themselves.

How should syntactic search options be presented to make them recognizable? We conducted a controlled experiment to investigate this question. Following the principle of recognition over recall, we hypothesized that showing contextualized usage examples would make the relations more recognizable. An alternative is to name the relation of interest and show blanks where the words that satisfy the relation would appear as in *X is the subject of Y* [12]; this was our baseline condition.

We gave participants a series of identification tasks. In each task, they were shown a list of sentences containing a particular syntactic relationship between highlighted words. They were asked to identify which relationship it

was from a list of four options. We presented the options in different ways, and compared the accuracy.

Our results confirm that showing examples in the form of words or phrases that match significantly improves the accuracy with which grammatical relationships are recognized over a standard baseline. Our findings also showed that different types of relations benefited differently depending on how much context was shown in the auto-suggest examples.

These findings suggest that a query interface in which a user enters a word of interest and the system shows candidate grammatical relations augmented with examples from the text will be more successful than the baseline of simply naming the relation and showing gaps where the participating words appear.

RELATED WORK

Trees are the traditional representation of syntactic parses, so trees are often the focus of query input for collections of syntactically parsed data.

Several approaches have adopted XML representations and the associated query language families of XPATH and SPARQL. For example, LPath augments XPath with additional tree operators to give it further expressiveness [9].

EXPERIMENT

Our goal was to find out whether showing examples improves the recognizability of grammatical relations. We tested two types of examples: a list of matching words and a list of matching phrases. Words are explicitly visible in the text, but phrases provides contextual information that helps determine the relationship.

Our hypothesis was:

> H1. Grammatical relations are identified more accurately when shown with examples of contextualizing words or phrases than without.

To test H1, participants were given a series of identification tasks. In each task, they were shown a list of 8 sentences, each containing a particular relationship between highlighted words. They were asked to identify the relationship from a list of 4 choices. Additionally, one word was chosen as a *focus word* that was present in all the sentences, to make the relationship more recognizable ("life" in Figure 1).

The choices were displayed in 3 different ways (Figure 1). The **baseline** presentation (Figure 1b) named the linguistic relation and showed a blank space with a peach background for the varying word in the relationship, the focus word highlighted in yellow and underlined, and any necessary additional words necessary to convey the relationship (such as "of" for the prepositional relationship containing "of", the third option).

The **words** presentation showed the baseline design, and in addition beneath it showed the word "Examples:" followed by a list of 4 example words that could fill in the peach-colored blank slot (Figure 1c). The **phrases** presentation again showed the baseline design, beneath which was shown the phrase "Patterns like:" and a list of 4 example phrases in which fragments of text including both the peach and the yellow highlighted portions of the relationship appeared (Figure 1a).

We used a between-subjects design. The task order and the choice order were not varied: the only variation between participants was the presentation of the choices. To avoid the possibility of participants guessing the right answer by pattern-matching, we ensured that there was no overlap between the list of sentences shown, and the examples shown in the choices as words or phrases. Not every relationship shown in the distractors was tested, and distractors were chosen to be ones intuitively most likely to be mistaken for the relationship shown.

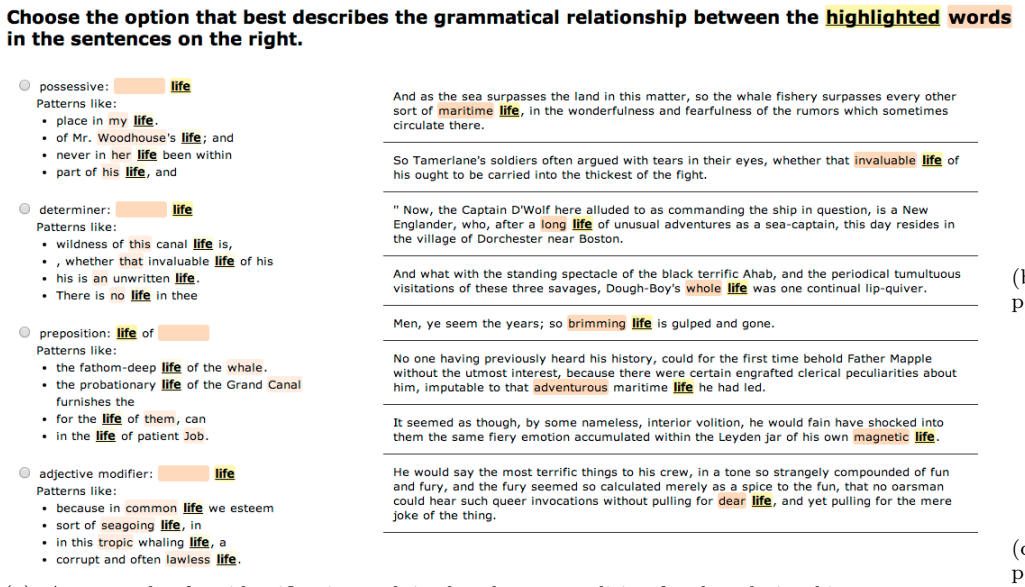
The tasks were generated using the Stanford Dependency Parser [3] on the text of *Moby Dick* by Herman Melville. We tested the 12 most common grammatical relationships in the novel, which fell into two categories:.

- Clausal or long-distance relations:
  - **advcl** Adverbial clause: *I **walk** while **talking***
  - **xcomp** Open clausal complement: *I **love** to **sing***
  - **ccomp** Clausal complement: *he **saw** us **leave***
  - **rcmod** Relative clause modifier: *the **letter** I **wrote** reached*
- Other relations:
  - **nsubj** Subject of verb: ***he threw** the ball*
  - **dobj** Object of verb: *he **threw** the **ball***
  - **amod** Adjective modifier ***red ball***
  - **prep_in** Preposition (in): *a **hole** in a **bucket***
  - **prep_of** Preposition (of): *the **piece** of **cheese***
  - **conj_and** Conjunction (and) ***mind** and **body***
  - **advmod** Adverbial modifier: *we **walk slowly***
  - **nn** Noun compound: ***Mr. Brown***

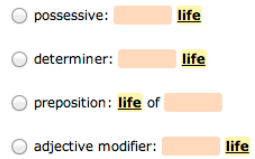
We tested each of the above relations 4 times, with 2 different focus words in each role. For example, the verb-subject relation `nsubj` was tested in the following forms:

- `nsubj(Ahab, ___)`: the sentences each contained 'Ahab', highlighted in yellow, as the subject of different verbs highlighted in peach.
- `nsubj(captain, ___)`
- `nsubj(___, said)`: the sentences all contained the verb 'said', highlighted in yellow, but with different subjects, highlighted in peach.
- `nsubj(___, stood)`

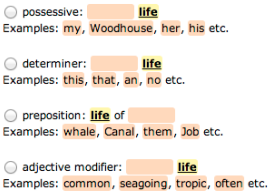
To maximize coverage, yet keep the total task time reasonable (average 6.8 minutes), we divided the relations



(a) An example of an identification task in the *phrases* condition for the relationship `amod(life, ___)` (where different adjectives modify the noun 'life'). The correct answer is 'adjective modifier' (4th option), and the remaining 3 options are distractors.



(b) The same options as they appear in the *baseline* condition.



(c) The same options as they appear in the *words* condition.

Figure 1: The appearance of the choices shown in the three experiment conditions.

above into 4 task sets of 3 relations each. Each relation was tested with 4 different words, making a total of 12 tasks per participant.

Participants

We chose Amazon's Mechanical Turk (MTurk) crowdsourcing platform as a source of study participants. The wide range of backgrounds provided by MTurk is desirable because our goal is to find a representation that is understandable to most people, not just linguistic experts. Participants were paid 50c (U.S.) for completing the task, with an additional 50c bonus if they correctly identified 10 or more of the 12 relationships. They were informed of the possibility of the bonus before starting the task.

To help ensure the quality of effort from participants, we included a multiple-choice screening question, 'What is the third word of this sentence?" Those that answered incorrectly were eliminated. 400 participants completed the study distributed randomly over the 4 task sets and the 3 presentations. To gauge their syntactic familiarity, we also asked them to rate how familiar they were with the terms 'adjective' (88% could define it), 'infinitive' (43%), and 'clausal complement' (18%).

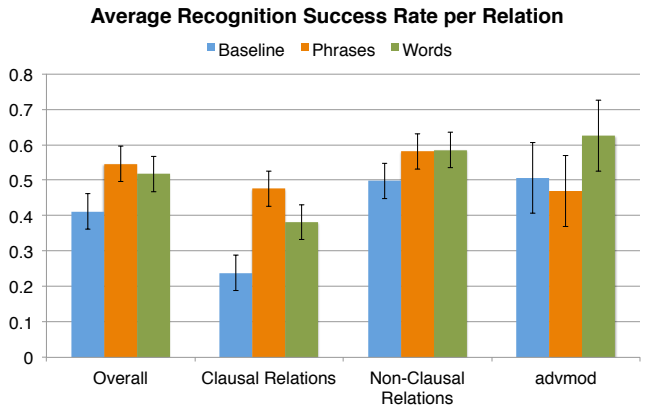


Figure 2: Recognition rates for different types of relations under the 3 experiment conditions.

Results

The results (Figure 2) confirm H1. Participants in conditions that showed examples (**phrases** and **words**) were significantly more accurate at identifying the relations than participants in the **baseline** condition. The average success rate in the **baseline** condition was

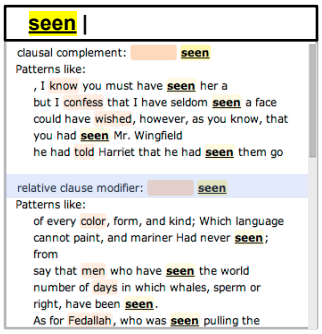


Figure 3: Mockup of auto-suggest for syntactic search on the word 'seen', showing clausal relations with example phrases for each.

41%, which is significantly[1] less accurate than **words**: 52%, ($p = 0.00019$), and **phrases**: 55%, ($p = 0.00013$).

Clausal relations operate over longer distances in sentences, and so it is to be expected that showing longer stretches of context would perform better in these cases; that is indeed what the results showed. Phrases significantly performed better than words and baseline. The average success rate is 48% for **phrases**, which is significantly more than **words**: 38%, (p = 0.017), and **baseline**: 24%, ($p = 1.9 \times 10^{-}9$).

For the non-clausal relations, there was no significant difference between **phrases** and **words**, although they were both overall significantly better than the baseline (words: $p = 0.0063$, phrases: $p = 0.023$). Among these relations, adverb modifiers (`advmod`) stood out (Figure 2), because evidence suggested that **words** (0.63% success) made the relation more recognizable than **phrases** (0.47% success, $p = 0.055$) – but the difference was only almost significant, due to the smaller sample size (only 96 participants encountered this relation). This may be because the words are the most salient piece of information in an adverbial relation – adverbs usually end in 'ly' – and in the phrases condition the additional information distracts from recognition of this pattern.

DISCUSSION

The results imply that auto-suggest interfaces for syntactic search should show candidate relationships augmented with a list of phrases in which they occur. A list of phrases is the most recognizable presentation for clausal relationships, and is as good as a list of words for the other types of relations. A mockup of such a search interface is shown in Figure 3. Selecting the choice will return all sentences that contain the search term and match the relation.

There is a tradeoff between recognizability and space required for scrolling through the choices, although it is important to keep in mind that because the suggestions are populated with phrases from the collection itself, they are informative. Further, the suggestions can be ordered by frequency of occurrence in the collection, or by an interestingness measure given the search word. As the user becomes more familiar with a given relation, it may be expedient to shorten the cues shown, and then re-introduce them if a relation has not been selected after some period of time as elapsed.

Participants exhibited surprisingly strong performance given that most were unfamiliar with the technical definitions of the relationships. Nonetheless, there is room for improvement in their scores, and it may be that additional visual cues, such as some kind of bracketing, will improve results. Furthermore, the current study did not test three-word relationships or more complex combinations of structures, and those may require improvements to the design.

[1]Using the Wilcoxson signed-rank test, an alternative to the standard T-test that does not assume samples are normally distributed.

REFERENCES

1. Androutsopoulos, I., Ritchie, G., and Thanisch, P. Natural language interfaces to databases–an introduction. *Natural Language Engineering 1*, 01 (1995), 29–81.
2. Culotta, A., and McCallum, A. Reducing labeling effort for structured prediction tasks. In *AAAI* (2005), 746–751.
3. De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, vol. 6 (2006), 449–454.
4. Gibbs, F., and Owens, T. Building better digital humanities tools. *DH Quarterly 6*, 2 (2012).
5. Hearst, M. *Search user interfaces*. Cambridge University Press, 2009.
6. Hirschman, L., Yeh, A., Blaschke, C., and Valencia, A. Overview of biocreative: critical assessment of information extraction for biology. *BMC bioinformatics 6*, Suppl 1 (2005), S1.
7. Jakubicek, M., Kilgarriff, A., McCarthy, D., and Rychlỳ, P. Fast syntactic searching in very large corpora for many languages. In *PACLIC*, vol. 24 (2010), 741–747.
8. Kepser, S. Finite structure query: A tool for querying syntactically annotated corpora. In *EACL* (2003), 179–186.
9. Lai, C., and Bird, S. Querying linguistic trees. *Journal of Logic, Language and Information 19*, 1 (2010), 53–73.
10. Levy, R., and Andrew, G. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation* (2006), 2231–2234.

11. MacLean, D. L., and Heer, J. Identifying medical terms in patient-authored text: a crowdsourcing-based approach. *Journal of the American Medical Informatics Association* (2013).

12. Muralidharan, A., and Hearst, M. A. Supporting exploratory text analysis in literature study. *Literary and Linguistic Computing 28*, 2 (2013), 283–295.

13. Resnik, P., Elkiss, A., Lau, E., and Taylor, H. The web in theoretical linguistics research: Two case studies using the linguists search engine. In *Proceedings of the 31st Meeting of the Berkeley Linguistics Society* (2005), 265–276.

14. Shneiderman, B., and Plaisant, C. *Designing The User Interface: Strategies for Effective Human-Computer Interaction, 5/e (Fifth Edition)*. Addison Wesley, 2010.

15. Soehn, J.-P., Zinsmeister, H., and Rehm, G. Requirements of a user-friendly, general-purpose corpus query interface. *Sustainability of Language Resources and Tools for Natural Language Processing 6* (2008), 27.