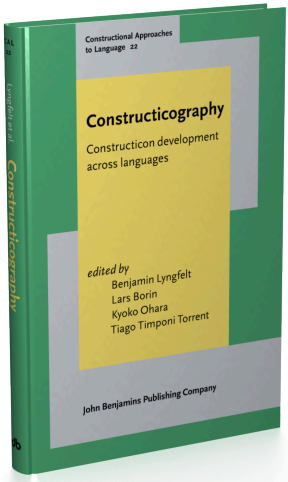
Constructional Approaches to Language 22

CHAPTER 8

Linguistics vs. language technology in constructicon building and use

Lars Borin
Dana Dannélls
Normunds Grūzītis

https://doi.org/10.1075/cal.22.08bor



Pages 229–253 of
Constructicography: Constructicon development across languages
Edited by Benjamin Lyngfelt, Lars Borin, Kyoko Ohara and Tiago Timponi Torrent
[Constructional Approaches to Language, 22]
2018. viii, 313 pp.

CHAPTER 8

Linguistics vs. language technology in constructicon building and use

Lars Borin, Dana Dannélls and Normunds Grūzītis

In this chapter, we describe the close interaction of linguists and language technologists in the Swedish constructicon project. This kind of collaboration is not so common today, because of the way that language technology has developed in recent decades, but in our case the collaboration has been very successful, and constituted a genuine instance of cross-fertilization, where an evolving language technology infrastructure and a computational lexical macroresource described in the chapter has formed an integral part of the Swedish constructicon development environment, while at the same time the structured linguistic knowledge described in the constructicon has informed the language technology making up the infrastructure.

Keywords: language technology, natural language processing, computational linguistics, machine learning, grammatical framework, computational lexical resource, corpus, Swedish, research infrastructure

1. Introduction

It may come as a surprise to most linguists – who we assume form the primary readership of this book – that, despite its name, present-day computational linguistics has a quite tenuous and in many cases nonexistent relationship to the academic discipline of linguistics. According to at least one author, "computational linguistics is not a specialization of linguistics at all; it is a branch of computer science" (Abney, 2011, p. 1).

While it is true that computational linguistics (or language technology/natural language processing/language engineering)[1] encompasses many kinds of activities

1. In this chapter, we will prefer the term "language technology" (LT), as that being most frequently encountered today, and also understood to subsume both text and speech technologies. We will also talk about NLP (natural language processing), as the application of LT systems to empirical language data, typically large volumes of text.

DOI 10.1075/cal.22.08bor

all dealing with language,[2] the fact of the matter is still that it rarely overlaps in its most fundamental concerns with those of linguistics. Rather than striving to describe human language or explain human linguistic competence, current language technology has as its main aim to build systems capable of processing unrestricted natural-language text (or speech), typically for some particular purpose or purposes (e.g., information access, translation, grammar correction, spoken dialogue). Even though such systems could be thought of as being crucially dependent on human-style language understanding, no claim is made on the part of their designers that these systems actually mimic how humans process language. And of course identical external behavior in a system can be caused by an infinitude of internal configurations. The designers of language technology systems are only interested in how to produce the desired behavior, not whether the computational machinery whereby this is achieved has anything in common with the cognitive or neurological mechanisms underlying human linguistic abilities.

For many years now, language technology has been completely dominated by statistical systems based on machine learning, so-called *data-driven* systems; for every grammar-based or rule-based LKB (Copestake, 2002) or Giellatekno (Trosterud, 2006) type system we find at least a dozen data-driven RASP (Briscoe, Carroll, & Watson, 2006) or MALT (Nivre et al., 2007) type systems in the literature.

As part of this development, even though the two fields started out close, research in language technology has become increasingly disassociated from the concerns of linguistics (Reiter, 2007). According to Wintner (2009, p. 642), there are mainly three reasons for this: (1) "applications that were based on explicit linguistic knowledge didn't scale up well"; (2) "[f]unding agencies (mainly in the U.S.) are motivated by short-term practical goals"; and (3) "[linguistics] focused mainly on syntax (and predominantly on English): and its theory became so obscure, so baroque, and so self-centered, that it became virtually impenetrable to researchers from other disciplines". This characterization primarily refers to the kind of linguistics that informed much early research in language technology, much of it being pursued under the label of generative grammar. On the other hand, language technology remained strangely untouched by other vigorous strands of linguistic research, such as typological and contact linguistics, sociolinguistics, psycholinguistics, or lexicography (including lexical semantics), to mention a few that all have generated a substantial body of work simultaneously with – sometimes even well before – the kind of linguistics that Wintner refers to.

Be that as it may, the fact of the matter is that data-driven approaches have made huge progress in recent times, in the wake of increasingly powerful computer

2. See <https://aclanthology.coli.uni-saarland.de/>

hardware and the availability of enormous amounts of training data in the form of billions of words of digital text.[3] The most recent incarnation of data-driven language technology is the so-called "deep learning" paradigm, which – being capable of "learning language" automatically from large amounts of raw textual data – according to some predictions will put language technologists out of business completely. However, others point out that machine learning will not make the domain problems go away, and that instead of focusing on numbers, "[m]ore of the field's effort should go into problems, approaches, and architectures" (Manning, 2015, p. 702).

In a discussion of the role of linguistics versus language technology in constructicon building and use, this particular strand of language technology research must loom large, for at least two reasons: (1) This is arguably the kind of language technology research which at present is most distant from contemporary linguistics, which makes the contrast to the constructicon work described elsewhere in this volume maximal (and the corresponding comparison maximally methodologically and theoretically fruitful); (2) since the data-driven systems seem to be here to stay and in reality today dominate the leading language technology conferences completely, they cannot be ignored – any serious attempt to include the kind of linguistic knowledge encoded in constructicons into state-of-the-art language processing systems must somehow accommodate this fact – which is why in the language technology research community, arguments are now made for the desirability of a new "linguistic turn" in language technology, and for allowing explicit linguistic knowledge to inform the statistical language processing paradigm (Reiter, 2007; Wintner, 2009; Bender, 2011; Manning, 2015). Except at its very beginning, the interaction of computational and other linguistics has been almost exclusively with the kind of linguistics that Wintner characterizes as syntax-focused and English-centered. What is rarely noted is that, even here, the traffic has been largely one-way, since, with some rare exceptions, work in computational linguistics has had very little influence on the development of theory or methodology in general linguistics (Pullum, 2009). However, Reiter (2007), Wintner (2009), Abney (2011), Bender (2011) and Manning (2015) see many opportunities for closer interaction between LT and linguistics, which hopefully would not be one-way

3. But note that the availability of training data is extremely unevenly distributed over the world's languages and further that many of the most popular data-driven methods contain a good deal of hidden language dependence (Bender, 2011). Because of this, grammar-driven language technology maintains a more modest but thriving existence in "ecological niches" which are still out of reach to data-driven techniques: Low-resource languages (Trosterud, 2006; Tyers & Pirinen, 2016), historical language varieties (Bollmann, 2013), controlled languages (Angelov & Ranta, 2009), and some others.

anymore, but a genuinely synergistic undertaking. In our view, too, these two fields stand to benefit considerably from a rapprochement. On the one hand, data-driven LT offers a well-developed and rigorous methodology for conducting large-scale hypothesis-driven empirical linguistic investigations (see, e.g., Abney, 2011), even though it is normally not referred to in these terms, but rather presented under the narrower heading "evaluation". This methodology, together with the analysis tools being developed and refined through it, has the potential to both scale up and widen the scope of corpus-based linguistics (see, e.g., Pullum 2007), which will benefit endeavors such as constructicon building (see Section 3 below), empirical investigation of constructs such as "entrenchment" (Schmid 2010), and others.

On the other hand, because of the Zipfian distribution of linguistic phenomena, even very large corpora, while revealing many things that linguists did not know about a language or language in general, will also almost certainly fail to provide evidence of some things that linguists do know. For this reason, it would be useful if these two sources of knowledge could somehow be combined.

In this connection, highly refined linguistic knowledge such as that encoded in framenets or constructicons is especially interesting, but at the same time raises many methodological challenges. In part this is because of the sometimes very different theoretical assumptions and methodological decisions characteristic of linguistics and language technology, especially those assumed to be self-evident and consequently unexpressed (Borin et al., 2010; Borin, Forsberg, & Lyngfelt, 2013).

2. Some theoretical and methodological observations

There is a more trivial and less interesting form of interaction between linguistics and data-driven language technology. Most data-driven methods learn a classification or labeling, e.g., of text word occurrences for their parts of speech, their syntactic functions, or their lexicon senses. Data-driven methods are conventionally subclassified into *supervised* and *unsupervised* methods. The unsupervised methods learn a classification from being exposed only to the data itself, i.e., the unannotated text in our case,[4] while supervised methods require preclassified training data, i.e., text where each word occurrence has been assigned its correct part of speech, syntactic function, lexicon sense, etc.

Since supervised methods tend to achieve considerably better classification accuracy for most kinds of linguistic annotation, language technology relies on

4. Although note that even "raw" text may come "pre-cooked" with some linguistic analysis, e.g., in the form of (orthographic) word and sentence segmentation, orthographic marking of proper nouns or nouns in general, etc.

linguistics for producing training data labelled with the relevant linguistic analyses. This is a very time-consuming and consequently expensive undertaking, and it furthermore requires that the language variety has been described linguistically to sufficient detail and also that there are trained annotators available, so that sufficient amounts of training data can be produced. In practice, for many of the world's languages we do not have good linguistic descriptions, and even for languages with many speakers and a long history of linguistic description, annotators with suitable training may be hard to come by (Liberman, 2009; Peldszus & Stede, 2013).

Consequently, a more promising avenue would consist in somehow making explicit linguistic descriptions, such as those found in a constructicon, inform data-driven language analysis, e.g., by introducing biases or/and guiding feature selection in the learning algorithms. In the case of grammar-driven systems, constructicon entries can of course be incorporated into the formalism itself, either directly or in modified form (see Section 4 below).

It was mentioned above that language technology has influenced linguistics (at least theoretical linguistics) even less than the opposite. This unfortunate state of affairs may in fact be rectified through a natural development, stemming from the availability of enormous corpora with increasingly sophisticated linguistic annotations and search interfaces, which linguists are more or less forced to use because of the sheer volumes of text available today, which cannot be investigated except with the help of computational tools. As empirical linguistics is becoming increasingly corpus-driven, linguists are exposed to annotations produced by the analysis tools of language technology, which arguably must influence their own thinking about linguistic matters. In the work on the Swedish constructicon, the analysis tools of language technology have been brought to bear on the task of extending the Swedish constructicon – and also the Swedish FrameNet (Dannélls, Friberg Heppin, & Ehrlemark, 2014; Johansson, 2014) – with new entries. This work is described in the next section.

The "turtles-all-the-way-down" notion that language has nothing but constructions may be intellectually satisfying and theoretically enlightening, but not obviously immediately compatible with the mechanics of practical language processing systems, which minimally assume a set of word-level units combining into phrase-level (and/or clause/sentence-level) units, and where the mechanisms for dealing with the two levels normally are quite different. This contradiction may be more imagined than real, however. At the level of their genetic machinery, all organisms on Earth are of the same kind (except maybe prions), but for many (most?) practical purposes we do not want to treat amoebas as being the same as elephants or chihuahuas. Against this background and also because of the many far-from successful attempts to build practical language processing systems based exclusively on linguistic formalisms, we may think of more pragmatically feasible

modes of drawing on construction grammar in such systems. One such mode could be as a way of reducing ambiguity in analyses. Especially computational syntactical analysis is plagued by immense ambiguity. In pure grammar-based systems this manifests itself as many analyses – hundreds of syntax trees even from fairly short sentences – whereas data-driven systems – which typically return only the most probable analysis – will suffer a general drop in performance as the number of theoretically possible alternatives grows.[5] The Swedish analysis system described below in Section 4.1 introduces information from the Swedish constructicon in its grammar in order not to have to distribute (seeming) noncompositionality among the components of constructions, as it were.

3. The role of language technology in constructicon population

There is also an undeniable sociological component to the interaction of language technology and linguistics, in the sense that this presupposes interaction among researchers in the respective fields, or rather interaction between researchers trained in computer science and those trained in linguistics. This seems to be happening increasingly rarely in language technology, as the observations cited above reveal. A happy exception to this trend, the work on the Swedish constructicon has benefitted from a long history of linguistically informed language technology research conducted in Gothenburg in close collaborations involving Swedish and general linguists and computer scientists.

Thus, the Swedish constructicon project was conceived and executed in a context where work on a lexical macroresource for Swedish language technology was well underway and crucially included a framenet for Swedish as a central component. This meant that the new project could draw on the same kind of language technology support for building the constructicon, that was being developed for the other resources, including the Swedish FrameNet. This context is described in the next section, and the general infrastructure developed in the collaboration for building the constructicon and editing constructicon entries is discussed in Section 3.2.

The most central contribution of language technology in the Swedish constructicon project has been the application of tools for automatic language analysis to the problem of finding construction candidates in large text corpora. This work is described in Section 3.3.

---

5. Although supervised data-driven systems will have the advantage that they will not be exposed to the full theoretical range of analyses, since a large share of these will not occur in the training data at all.

3.1 Towards a lexical macroresource for Swedish language technology

The Swedish FrameNet++ (SweFN++) project (Borin et al., 2010) was many things simultaneously. Its main goal was the creation of an integrated lexical macroresource for Swedish to be used as a basic infrastructural component in Swedish language technology research and in the development of natural language processing (NLP) applications for Swedish. A significant result of the project is the Swedish FrameNet, containing almost 40,000 lexical units, making it the world's most extensive framenet on this measure.

The macroresource is topologically a hub-and-spokes structure. There is one primary, central lexical resource, a pivot, to which all other resources are linked. This is SALDO (Borin & Forsberg, 2009; Borin, Forsberg, & Lönngren, 2013), a large (ca. 147K entries and 2M wordforms), freely available (under a Creative Commons Attribution license) morphological and lexical-semantic lexicon for modern Swedish. It has been selected as the pivot partly because of its size and quality, but also because its form and sense units are identified by carefully designed unique *persistent identifiers* (PIDs) to which the lexical information in other resources are linked.

As a semantic lexicon, SALDO is a kind of lexical-semantic network, superficially similar to Princeton WordNet (Fellbaum, 1998), but quite different from it in the principles by which it is structured. The basic organizational principle of SALDO is hierarchical. Every entry in SALDO – representing a word sense – is supplied with one or more semantic descriptors, which are themselves also entries in the dictionary. All entries in SALDO are actually occurring words or conventionalized or lexicalized multi-word expressions (MWEs) of the language.[6] No attempt is made to fill perceived gaps in the lexical network using definition-like paraphrases, as is sometimes done in WordNet (Fellbaum, 1998, 5f). One of the descriptors, called primary, is obligatory. The primary descriptor is the entry which better than any other entry fulfills two requirements: (1) it is a semantic neighbor of the entry to be described; and (2) it is more central than it. Both these aspects need some clarification.

---

6. In order to make SALDO into a single hierarchy, an artificial entry, called PRIM, is used as the primary descriptor of 45 semantically unrelated entries at the top of the hierarchy, making all of SALDO into a single rooted tree. Here we may also add that SALDO contains words of *all* parts of speech, again distinct from WordNet, which only includes the four open word classes nouns (including proper nouns), verbs, adjectives, and adverbs. About 5% of the entries in SALDO are MWEs, representing all parts of speech. An interesting methodological question in this connection – to which no definitive answer has been found – is how Swedish constructicon entries should relate to SALDO MWE entries.

That two entries are semantic neighbors means that there is a direct semantic relationship between them, for instance synonymy, hyponymy, argument–predicate relationship, etc. Centrality is determined by means of several criteria, the most important being frequency: a frequent entry is more central than an infrequent entry.[7] The basic linguistic idea underlying SALDO is in effect that, semantically speaking, the whole vocabulary of a language can be described as having a center – or core – and (consequently) a periphery. The notion of core vocabulary is familiar from several linguistic subdisciplines (Borin, 2012). In SALDO, the higher levels in the hierarchy contain simpler and more basic entries. Contrast this with WordNet, where the higher nodes in the hierarchy contain very abstract vocabulary (e.g. 'entity').

Below, we give a few examples of entries with their primary and secondary descriptor senses:

balkong – hus	'balcony (n)' – 'house (n)'
Berlin – stad + Tyskland	'Berlin (prop)' – 'city (n)' + 'Germany (prop)'
berusa – yr	'intoxicate (v)' – 'dizzy (adj)'
bete – fånga	'bait (n)' – 'catch (v)'
bete^2 *– beta*	'grazing land (n)' – 'graze (v)'
bete^3 *– tand + djur*	'tusk (n)' – 'tooth (n)' + 'animal (n)'
bröd – mat + mjöl	'bread (n)' – 'food (n)' + 'flour (n)'
brödföda – uppehälle	'daily bread (n)' – 'subsistence (n)'
bröllop – gifta sig	'wedding (n)' – 'get married (v)'

The standard scenario for a lexical resource to be integrated into the macroresource is to (partially) link its entries to the sense PIDs of SALDO, via their citation form and part of speech. This typically leads to ambiguity, since most of the resources associate lexical information to part-of-speech-tagged lemmas, and not to word senses. Some of these ambiguities can be resolved automatically – especially if information from several resources is combined – but in the end, manual work is required for complete disambiguation. However, like many other linguistic phenomena, the distribution of senses over citation forms in lexical resources is roughly Zipfian (Moon, 2000; Borin, 2010). Thus, the vast majority of the lemmas

7. The actual work on SALDO relies mainly on the lexicographical experience and linguistic intuition of the compilers, who use clues such as stylistic value, word-formation complexity, the type of semantic relation holding between an entry and its primary descriptor, acquisition order in first-language acquisition, etc. Frequency correlates highly with these, however: It turns out that about 90% of the SALDO entries have primary descriptors which are at least as frequent as the entries themselves in a corpus of more than one billion words of Swedish. A more detailed description and discussion of the semantic organization of SALDO can be found in Borin, Forsberg, and Lönngren (2013, pp. 1196–1200).

are monosemous, reducing the sense mapping problem to the much simpler problem of pairing up forms between lexical resources.

The Swedish constructicon project uses the lexical infrastructure developed in the SweFN++ project and maintained as part of Språkbanken's (the Swedish Language Bank) LT infrastructure (see the next section), including the use of SALDO word-sense identifiers. This means that the constructicon shares all the linguistic information available in the component resources, which opens for exciting opportunities both in NLP and linguistics. A concrete example: The macroresource also includes historical lexical resources (Borin, Forsberg, & Kokkinakis, 2010; Borin & Forsberg, 2011), the oldest describing Old Swedish (13th–16th c.). The hope is that a successful (but possibly partial) linking of SALDO to the historical lexicons will make it possible to project various linguistic information from the modern resources onto the historical resources, allowing, e.g., identification of possible counterparts of modern constructions in historical varieties of Swedish, making it possible to study the diachronic development of selected constructions in empirical corpus data.

3.2 A general lexical infrastructure and a language-aware lexicon editor

The activities described above also have a more technical side, a generalized lexical infrastructure, called Karp (Borin et al., 2012). The heart of the lexical infrastructure is the lexical macroresource described in the previous section, a large network of interconnected lexicons (Borin et al., 2010; Borin, Forsberg, & Lyngfelt, 2013), all encoded in the ISO Lexical Markup Framework (LMF) format (ISO, 2008; Francopoulo, 2013).

Even though the lexical macroresource is primarily intended for use in LT applications, its component lexicons are still very much lexicographical entities. Thus, from a linguistic point of view, the work on individual resources as well as on their integration is at heart a genuinely lexicographical activity, to boot one with considerable potential to make significant theoretical contributions to lexicology, lexical semantics and lexical typology because of the large-scale empirical nature of our endeavor and the diversity of the lexical resources involved. In general, working with large amounts of data as we do, requires good tools. The Karp lexical infrastructure has been designed with this in mind. It supports the work on creating, curating, and integrating the lexical resources, and the maintenance in parallel of online-browsable and downloadable versions of the resources. An important feature of the lexical infrastructure is that we maintain a strong bidirectional connection to our corpus infrastructure Korp (Borin, Forsberg, & Roxendal, 2012). For example, the corpora are annotated with the lexical information available through

Karp, and the language examples for the lexical resources in Karp are retrieved from Korp. Similarly, corpus frequencies for lemmas and wordforms are provided by Korp.

In the Swedish constructicon project a dedicated constructicon editor was developed in Karp, which has contributed greatly to achieving an efficient workflow in the project (see Lyngfelt, Bäckström et al., this volume).

3.3 Mining corpora for construction candidates with language tools

As discussed by Lyngfelt, Bäckström, et al. (this volume), the work on the Swedish constructicon has focused on *partly schematic constructions*, i.e., constructions containing some fixed lexical element or elements, such as the construction `i_adjektivaste_laget` illustrated in their Example (1), repeated here slightly modified as (1). In (1), the construction is set in boldface, and the fixed parts of the construction are underlined.

(1) Jag ska erkänn-a att det här är en kladdkaka **i**
I shall admit-INF that it here be.PRS a sticky.chocolate.cake **in**
söt-ast-e **lag-et**
sweet-SUV-DEF **measure-DEF**
'I must admit that this is a sticky chocolate cake **a bit too sweet**'

There has been a considerable amount of recent work on MWEs in language technology and in corpus linguistics. In the latter field MWEs are sometimes referred to under other terms such as *formulaic language* (Wray 2002) or *lexical bundles* (Biber & Conrad 1999). While linguists have devoted much energy to classifying and characterizing MWEs in general (but as a rule not cross-linguistically informed) terms, much of the language technology research has focused on developing data-driven methods for finding certain subtypes of MWEs in text (see Pecina, 2010, for a good overview). Since some MWE types recognized in the literature are structurally similar to the schematic constructions making up the Swedish constructicon, the Swedish constructicon team has experimented with methods inspired and informed by MWE research in language technology in order to generate construction candidates from linguistically annotated corpora (Bäckström et al., 2013; Forsberg et al., 2014), mentioned by Lyngfelt, Bäckström, et al. (this volume) and explained and motivated in more detail in what follows.

One of the goals of the Swedish constructicon project has been to develop tools for automatic identification of constructions in authentic texts. This is a highly desirable research objective in itself, with potential uses in a number of NLP applications. In addition, the same methods provide the project with a heuristic tool.

By automatically extracting various kinds of regularities in texts, we may discover patterns that might otherwise have been overlooked. This especially concerns seemingly insignificant constructions that do not stand out against the context the way spectacular idioms do. The resulting findings are treated as construction candidates, a subset of which may be considered actual constructions after manual evaluation.

Language technology is both a means and a goal for the Swedish constructicon project. The experiments presented by Bäckström et al. (2013) and Forsberg et al. (2014), turned out to be valuable means for identifying potential constructions; the method used in the experiments provides construction candidates, statistically identified recurring linguistic structures, which are then manually evaluated to filter out material for actual constructicon entries. In a longer perspective, we are also working towards developing methods for automatic identification of particular constructions, which is a desirable research objective in itself, with potential for improving automatic language analysis systems.

The experiments were set up and executed using the resource infrastructure of Språkbanken, a modular and interoperable set of resources and tools in the form of web services for accessing, browsing, editing and automatically annotating resources.

Concentrating on the partly schematic constructions targeted by the Swedish constructicon as well as on purely schematic constructions, the method chosen for suggesting potential constructions consists of two steps: (1) extracting and counting syntactic patterns occurring in a large corpus; (2) ranking those patterns by a relevance measure based on a hypothesis about what characterizes a construction from a statistical point of view.

The aim was to capture constructions which can be described as a sequential pattern – a so-called *n-gram* – of *n* adjacent units: specific words, words with a particular part-of-speech (POS) tag, or phrases. This heuristic covers many important grammatical constructions, in particular the most common basic phrases.

The method works by going through all text in a large linguistically annotated corpus (1 million words in Bäckström et al., 2013 and 19 million words in Forsberg et al., 2014), taking each successive word as the first word in a number of sequences up to a maximum length. For each such sequence, all patterns of which that sequence is an instance are generated and counted. For the two-word sequence *in London*, there are five such patterns. First, there are three ways to form a pattern by replacing words by POS tags:

in [PROPER NOUN]
[PREPOSITION] London
[PREPOSITION] [PROPER NOUN]

Furthermore, since *London* is also an NP, there are two more generalizations in terms of that phrase:

in [NP]
[PREPOSITION] [NP]

These patterns are similar to the "hybrid n-grams" used by Wible & Tsao (2010) but more general since they can involve phrases.

The NLP aspect of this work was provided by the NLP tools available through Språkbanken's infrastructure. POS tags were assigned using HunPos (Halácsy, Kornai, & Oravecz, 2007) trained on the Stockholm–Umeå corpus (Gustafson-Čapková & Hartmann, 2006), and phrases were assigned on the basis of automatic syntactic analyses provided by MaltParser (Nivre et al., 2007) trained on the Swedish Treebank (Nivre, et al., 2008). MaltParser outputs dependency structures, which were converted into phrase structure representations using heuristics, e.g., a dependency subtree dominated by a common or proper noun or a pronoun becomes an NP.

Finally, the extracted patterns were ranked based on a statistical collocation measure, with due consideration of shorter patterns included in longer ones, and also only including patterns which had the same start and end point as some phrase. In the experiment described by Forsberg et al. (2014), the 1,200 most highly ranked patterns were then presented to three linguists for evaluation as to their suitability as construction candidates, together with the (up to) five most frequent instances of each pattern. In total, about 200 promising construction candidates were found, a lot considering that the automated part of the experiment took very little time to set up and execute, thanks to the well-developed LT infrastructure available, and that consequently the manual effort of the linguists was limited to going through a maximum of 7,200 examples (1,200 patterns plus a maximum of five instances of each), rather than 19 million words of text.[8]

There are many directions in which these experiments could be extended, such as:

- Using larger corpora to fight the sparse-data problem
- Using different text types in order to capture domain-specific constructions
- Working with dependency subtrees instead of word sequences

8. Of course, we cannot know what has been missed by this search and filtering procedure, but as far as we can see, there is no realistic alternative way of finding this out, when the input data comprises millions or even billions of words of text.

4. Using constructicons in language technology systems

Whereas frame-semantic representations have seen a fair amount of interest from the language technology community, as a practical, light-weight semantic representation suitable for deployment in NLP systems, the construction grammar formalism sprung from the same intellectual source has received much less attention.

The main computational implementation of construction grammar, Embodied Construction Grammar (e.g., Chang & Maia, 2001; Bergen & Chang, 2005, 2013; Bryant, 2008; Chang, 2008; Schneider, 2010), does not in fact represent an effort to build a practical language-processing system based on construction grammar. Instead, it reflects an older tradition in computational linguistics, where computer simulation is seen as a productive way of investigating human linguistic behavior, by devising "computational implementations of cognitively motivated theories of morphology, metonymy, metaphor, generation, and mental spaces" (Bryant 2008, p. 214).

Just as in the case of frame semantics, in order for it to be considered at all for inclusion in an NLP system, a constructicon must have a high coverage in the domain where the system is to be applied. This, together with a general lack of awareness in the language technology community about construction grammar, makes it hard to believe that we will see construction-based practical NLP systems anytime soon. However, we are already seeing construction information being incorporated in more "conventional" NLP systems, a very exciting and promising development. The next section provides a concrete example of how the Swedish constructicon has been used in exactly this way.

4.1 Using the Swedish constructicon for language analysis

Although the Swedish constructicon (SweCcn) is primarily created for linguistic research, with the help of language technology, language technology itself can benefit considerably from formalized, computational constructions that make the non-compositional parts of the grammar-lexicon interface compositional again. Figure 1 illustrates this with a simple example where the Swedish construction `behöva_något_till_något` 'need *something* for *something*' is recognized as a part of the computed compositional abstract syntax tree of the given clause. Moreover, the construction is fully embedded in the grammatical analysis: it produces a regular verb phrase (VP), and it takes regular noun phrases (NPs) – the variable parts of the construction – as arguments. The fixed parts of the construction (the verb *behöva* 'need' and the preposition *till* 'to/for') are not included in the abstract interlingual analysis; their inclusion and surface realization are left to the language-specific grammar (the Swedish grammar in our case).

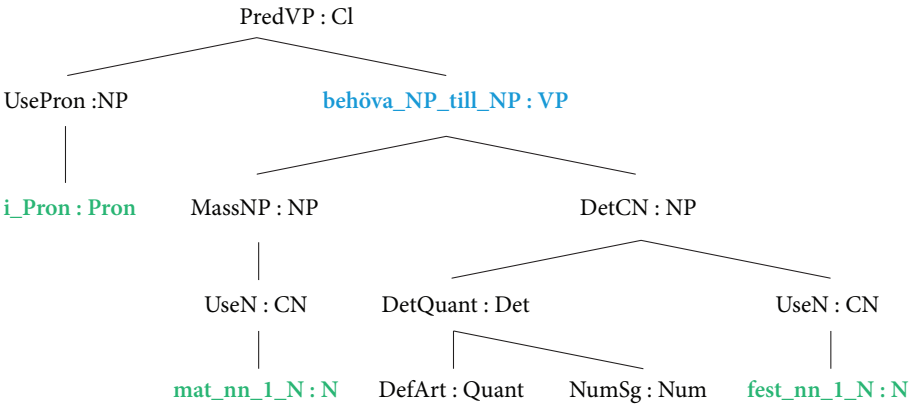


Figure 1. An abstract syntax tree for a clause containing an embedded construction: *jag behöver mat till festen* 'I need food for the party'

The role of computational constructions and potentially multilingual constructicons becomes more apparent in machine translation as illustrated in Figure 2. The preposition *till* in `behöva_något_till_något` literally translates as 'to', but its meaning in this construction is 'for'. In contrast to fixed multi-word expressions, it is impossible to list all instances of this construction in a translation lexicon. Statistical machine translation can theoretically handle such simple constructions, but it often fails because of insufficient training data.

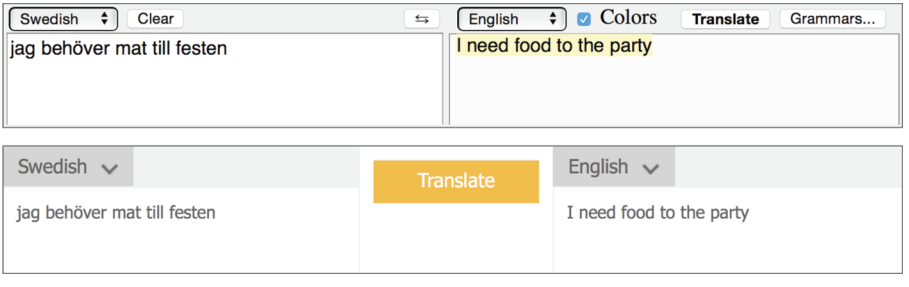


Figure 2. A simple example that illustrates the need for a multilingual constructicon in both rule-based machine translation (the upper screenshot of GF Wide Coverage Translator) and statistical machine translation (the lower screenshot of Bing Translator)

Since the current number of annotated examples per construction is relatively low in the Swedish constructicon, while the level of abstraction in the construction descriptions is relatively high, we have begun by converting the linguistically oriented descriptions into Grammatical Framework (GF), a computational grammar formalism (Ranta, 2004). The aim of formalizing the Swedish constructicon in GF

is twofold: (i) to obtain a more precise and consistent insight into the types and descriptions of Swedish constructions, (ii) to implement and test an extension to an existing grammar-based parser for Swedish, so that the parser would be able to recognize constructions as part of the sentence analysis and thereby reduce ambiguities. This will also facilitate the future development of construction-aware data-driven NLP systems.

In the rest of this section, we outline our methodology on how to systematically formalize the semi-formal representation of the Swedish constructicon in GF, showing that a computational GF construction grammar can be, to a large extent, acquired automatically. A side result of our approach is that it has been helpful for improving the consistency of the Swedish constructicon and for characterizing and annotating construction entries in the database of Swedish constructions.

4.2 The database of Swedish constructions

The Swedish constructicon follows the view that a construction is the basic linguistic unit in language (Goldberg, 1995; Croft, 2001). According to this view there is no strict separation between the lexicon and grammar. As mentioned by Lyngfelt, Bäckström et al., (this volume) and also adhered to here, constructions in the Swedish constructicon are partially schematic multi-word expressions with fixed and variable slots. Information about construction entries is encoded in an ISO LMF database that is integrated into the lexical infrastructure of Språkbanken described in Section 3.1 (see also Lyngfelt et al., 2012). This information includes: a descriptive name of the unit, a grammatical category, a free-text definition, a set of annotated example sentences extracted from Korp, a structure sketch (i.e., a formal description) of both the morphosyntactic structure, and the internal and external construction elements (CEs). Internal CEs are part of the construction proper, and external CEs are valency bound elements of the construction.

In the structure sketches, internal CEs are delimited by brackets, where alternative values and lexical units (LUs) are separated by a bar, for example the structure sketch for the SweCcn entry `behöva_något_till_något` is: [behöva[1] NP_1 till[1] NP_2 | VP]. This sketch combines four internal CEs, two lexically fixed elements and two variable slots of which one contains two alternative values: an NP and a VP. Each CE is further described with a feature matrix containing several attributes, including: (a) the lexical unit's identifier from SALDO (see Section 3.1) for the lexically fixed elements; (b) the semantic role of each element (except preposition elements); (c) the name of the element; and (d) the grammatical category (either part-of-speech tag or phrase type). An example of the feature matrix for the internal CEs of `behöva_något_till_något` taken from the SweCcn database is:

{role = “State” name = “State” cat = “V” lu = “behöva..1”}
{role = “Requirement” name = “Requirement” cat = “NP”}
{name = “P” cat = “P” lu = “till..1”}
{role = “Purpose” name = “Purpose” cat = “NP|VP”

As the above example shows, the set of feature matrices is rather detailed but there is no explicit indication of the corresponding elements which are given in the structure sketch. In many matrices the order of the specifications for each element tends to be diverse; it does not necessarily follow the element specification given in the structure sketch. When we started analyzing the database for language technology analysis, systematic recording of the different specifications and their corresponding elements became very relevant for improving the automatic analysis.

Each construction bears a grammatical category. As of August 2016 there are nine categories specified in the database, viz. AdvP (Adverb Phrase), AP (Adjective Phrase), AP|AdvP (Adverb Phrase or Adjective Phrase), Intj (Interjection), NP (Noun Phrase), PP (Prepositional Phrase), S (Sentence), VP (Verb Phrase), XP (any phrase type). In our work we chose to begin with constructions of category VP mainly because this category predominates in the SweCcn database, with more than 100 constructions available, but also because VP constructions are expressed by complex internal structures.

4.3 Grammatical framework

Grammatical framework (GF; Ranta, 2004) is a categorial grammar formalism and a framework for implementing computational grammars. It provides built-in support for multilingual grammars, which holds great potential for implementing, unifying and interlinking constructions of different languages.

GF is characterized by its two-level approach to natural language representation: abstract syntax which defines the language-independent structure, and concrete syntax which defines the language-specific syntactic and lexical realization of the abstract syntax. The same abstract syntax can correspond to many (multilingual) concrete syntaxes – mappings from abstract syntax trees to feature structures and strings. GF grammars are bi-directional – they can be used for both parsing and language generation. The framework is suitable for implementing general-purpose syntactic grammars as well as domain-specific semantic grammars.

Notably, GF provides a general purpose resource grammar library, RGL (Ranta, 2009), for currently 30 languages that implement the same abstract syntax. The RGL has a high-level interface that provides constructors like `mkVP : V → NP → VP` for building a verb phrase from a verb and a noun phrase without the need of specifying low-level details like inflectional paradigms, syntactic agreement and word

order. These details are handled by the language-specific resource grammars. The coverage of the general-purpose Swedish grammar is one of the largest in the GF RGL comprising a lexicon with over 100,000 lexical entries from SALDO.

4.4 Constructing a computational constructicon

Our method of converting the lexicographic SweCcn entries into a computational GF construction grammar comprises several steps:

1. preprocessing: automatic normalization, consistency checking and rewriting of the structure sketches;
2. automatic generation of the abstract and concrete syntaxes of a GF grammar;
3. semi-automatic verification of the acquired grammar.

Constructions may have optional CEs, alternative types of CEs or alternative LUs, and even alternative word order. In the structure sketches, optional CEs are delimited by parentheses, and alternative types/LUs are separated by a vertical bar, e.g.:

(2) a. `behöva_något_till_något`: [behöva^1 NP_1 till^1 NP_2|VP]
e.g. behöva kvällen till att plugga 'need the evening to study'

(3) a. `verba_av_sig.transitiv`: [V av^1 Pn_{refl} (NP)]
e.g. *ta av mig skorna* 'take off myself the shoes'

(4) a. `snacka_NP`: [snacka^1|prata^1|tala^1 NP_{indef}]
e.g. *prata skolminnen* 'talk school memories'

(5) a. `få_resultativ.agentiv`: [få^1 NP PcP]
e.g. *få gräsmattan klippt* 'get the lawn trimmed'

(6) a. `x-städa`: [N|Adj+städa^1]
e.g. *storstäda* 'bigclean (V)'

The variable CEs may have indices denoting difference, formal identity (repetition), co-reference, etc. In the case of a lexical construction that is realized by a compound word, its internal CEs are delimited by the plus sign indicating concatenation. Suffixing (as in (1) in Section 3) is indicated by a hyphen.

The automatic preprocessing of constructicon entries comprises:

1. Normalization of the structure sketches and attribute values in the feature matrices, fixing a lot of various inconsistencies due to the manual annotation.
2. In case of optional CEs and alternative types of CEs, there are formally several constructions compressed into one. The original structures are rewritten so that for each combination there is a separate alternative structure. This however does not apply to alternative LUs. If a CE is represented by a fixed set of LUs, we

assume that they are interchangeable (synonymous). Otherwise they should be either split into alternative constructions (separate entries), or the CE should be made more general (variable).

3. The rewritten structure sketches are enriched with additional morphosyntactic information from the feature matrices, so that a complete description is at hand.
4. The grammatical categories used in the Swedish constructicon are converted into GF categories.[9] In specific cases, the conversion may lead into a more general or more specific description as well as it may include morphosyntactic tags and may depend on the contextual CEs. This requires a subsequent rewriting of the whole construction. A few categories, however, are not converted at this step; their conversion is postponed to the generation of the GF grammar. For instance, Pc (participle) and PcP (participle phrase) are not converted to V (verb) and VP respectively, as they have to be treated differently in the concrete syntax: PcP is a VP that is eventually converted to AP (adjectival phrase) or Adv (VP-modifying adverb) as illustrated by `få_resultativ.agentiv` (5).

Below in (2b)–(6b) are given rewritten structural descriptions of the sample constructions introduced above in (2a)–(6a). Note that we ignore the SALDO sense identifiers; the word sense information is not used so far in generating the grammar.

(2) b. behöva$_{V}$ NP$_{1}$ till$_{Prep}$ NP$_{2}$ | behöva$_{V}$ NP till$_{Prep}$ VP

(3) b. V av$_{Prep}$ Pron$_{refl}$ NP | V av$_{Prep}$ Pron$_{refl}$

(4) b. snacka|prata|tala$_{V}$ aSg_Det CN | snacka|prata|tala$_{V}$ aPl_Det CN | snacka|prata|tala$_{V}$ CN

(5) b. få$_{V}$ NP PcP$_{perf}$

(6) b. N + städa$_{V}$ | A + städa$_{V}$

The rewritten structural descriptions of constructions provide sufficient information to generate both the abstract and the concrete syntax of a constructicon-based grammar, an extension to the existing wide-coverage Swedish GF resource grammar.

The generation of the abstract syntax is rather straightforward. Each construction is represented by one or more grammar rules (functions) depending on how many alternative structure descriptions are produced (rewritten) in the preprocessing phase. For VP constructions, the average number of alternative functions per construction is 1.4 while the maximum number is 6, produced by `snacka_NP.emfas`: [snacka^{1}|prata^{1} (AP) NP$_{indef}$].

Each function takes one or more arguments that correspond to the variable CEs of the respective alternative construction description. The fixed CEs are not

9. <http://www.grammaticalframework.org/lib/doc/synopsis.html#toc2>

represented by the abstract syntax. The variable CEs are represented only by their grammatical categories; other morphosyntactic constraints (if any) are handled by the language-specific concrete syntax.

The rewritten structure descriptions shown above in (2b)–(6b) are represented by the following abstract functions in the GF construction grammar (2c)–(6c):

(2) c. behöva_något_till_något$_1$: NP → NP → VP
behöva_något_till_något$_2$: NP → VP → VP

(3) c. verba_av_sig_transitiv$_1$: V → NP → VP
verba_av_sig_transitiv$_2$: V → VP

(4) c. snacka_NP$_1$: CN → VP
snacka_NP$_2$: CN → VP
snacka_NP$_3$: CN → VP

(5) c. få_resultativ_agentiv: NP → VP → VP

(6) c. x_städa$_1$: N → VP
x_städa$_2$: A → VP

As for the concrete syntax, many constructions can be implemented in GF by systematically applying the high-level constructors provided by the GF RGL:[10]

mkVP: VP → Adv → VP
mkVP: V2 → NP → VP
mkV2: V → V2
mkV: Str → V
mkAdv: Prep → NP → Adv
mkPerp: Str → Prep
etc.

For instance, `behöva_något_till_något`$_1$ can be implemented by first making a two-place verb (V2) from V, and then combining it with the first NP into a VP; the preposition can be combined with the second NP into a prepositional phrase (Adv) which can then be attached to the VP:

(2) d. behöva_något_till_något$_1$ np$_1$ np$_2$ =
mkVP
(mkVP (mkV2 (mkV "behöver")) np$_1$)
(mkAdv (mkPrep "till") np$_2$)

10. <http://www.grammaticalframework.org/lib/doc/synopsis.html#toc5>

The question is how to make such constructor applications systematically and, thus, automatically, given the various construction descriptions. Essentially, this is a parsing problem in itself: we can look at CEs as words in the formal construction description language for which we need a grammar to combine the lists of CEs into the parse trees of RGL constructors and their arguments. We have defined such an auxiliary GF grammar to generate the implementation of the abstract functions in a GF construction grammar. This approach has emerged from our work on creating a multilingual computational grammar for FrameNet (Grūzītis & Dannélls, 2017). The proposed approach to GF construction grammars can also be reused for other languages, and it is explained in more detail by Grūzītis et al. (2015).

The resulting module of the computational construction grammar yielded 127 GF functions: 1.4 alternative realizations of 93 VP constructions in SweCcn. In most cases, the automatically generated concrete (Swedish) implementation of the abstract GF functions is adequate. In general, however, the implementation requires manual validation and minor or major revisions. An example of a construction that was not generated successfully is `disjunktiv_samordning.korr`[varken^1|vare_sig^1 XP_1 eller^1 XP_2]. Following the definition of the construction, the scope of negation extends over the coordinated sentences, either two coordinated sentences or more. In GF this is a special case of conjunction and therefore requires defining new operations to uncover dimensions of deep linguistic information, something that is very limited in an automatic process such as the one described here.

4.5 Preliminary analysis of the automatically generated computational constructicon

One possible evaluation procedure which we considered in this work was to parse some corpus data with the generated grammar and examine the results.

In the first step we compiled a benchmark collection of 337 example sentences from the annotated sentences of the VP constructions in SweCcn. The second step was to parse each sentence using the generated grammar. Several heuristics were applied before parsing to overcome problems such as: (i) lack of the subject which is necessary for generating a proper clause in GF, and (ii) missing lexical entries for proper nouns, verbs and compounds not covered in the GF lexicon for Swedish. Heuristics were formulated for inserting subjects, replacing compounds, proper nouns and verbs, and for changing the tense or the verb string.

Out of the 337 annotated example sentences, 281 turned out to have a corresponding concrete function in the construction grammar. The parser successfully parsed 80% of these sentences. Cases where the parser could not return any parse

trees were mainly due to annotation inconsistencies or errors in the SweCcn database, for instance, a feature matrix requires the singular form of an NP although the plural form exists among the annotated examples. This turned out to be helpful feedback to the linguists compiling the structural descriptions of the SweCcn constructions.

Most of the words the GF parser failed with were proper nouns and compounds. With respect to the grammar, the parser failed in cases where there existed ill-formed or syntactically complex sentences, often containing coordination and conjunctions and long sentences, containing irrelevant phrases and punctuation that fall outside the construction. In the future, the grammar analysis can be complemented with "inverse testing", i.e., we can use the GF built-in support for automatically generating a treebank and test whether our grammar is able to provide correct syntactic analyses for each construction. The SweCcn developers (the linguists) could in turn further inspect and validate the generated syntactic trees.

5. Conclusion and outlook

In this chapter, we have described the close interaction of linguists and language technologists in the Swedish constructicon project. The collaboration has been very successful, and – importantly – constituted a genuine instance of cross-fertilization, where an evolving LT infrastructure has formed an integral part of the constructicon development environment, while at the same time the structured linguistic knowledge described in the constructicon has informed the language technology making up the infrastructure.

From the point of view of the infrastructure, we have seen an overall quality improvement resulting from corrections of lots of bare syntactic categories incorrectly labeled in the constructicon database. From the point of view of the grammar formalism, we have seen improvements emerging as a result of reduced ambiguity in the parsing analyses.

We believe that the fact that the closely interlinked Swedish lexical macroresource now also includes a constructicon will allow linguists to conduct empirical research into the synchronic and diachronic properties of Swedish constructions on the basis of Språkbanken's vast and varied text corpora.

References

Abney, S. (2011). Data-intensive experimental linguistics. *Linguistic Issues in Language Technology* 6.

Angelov, K., & Ranta, A. (2009). Implementing controlled languages in GF. In N. E. Fuchs (Ed.), *Controlled natural language* (pp. 82–101). Berlin: Springer.

Bäckström, L., Borin, L., Forsberg, M., Lyngfelt, B., Prentice, J., & Sköldberg, E. (2013). Automatic identification of construction candidates for a Swedish constructicon. In *Proceedings of the workshop on lexical semantic resources for NLP at NODALIDA 2013* (pp. 2–11). Oslo: NEALT.

Bender, E. (2011). On achieving and evaluating language-independence in NLP. *Linguistic Issues in Language Technology* 6.

Bergen, B. K., & Chang, N. (2005). Embodied Construction Grammar in simulation-based language understanding. In J. -O. Östman & M. Fried (Eds.), *Construction grammars: Cognitive grounding and thoretical extentions* (pp. 147–190). Amsterdam: John Benjamins. doi:10.1075/cal.3.08ber

Bergen, B. K., & Chang, N. (2013). Embodied Construction Grammar. In T. Hoffman & G. Trousdale (Eds.), *The Oxford handbook of construction grammar* (pp. 168–190). Oxford: Oxford University Press.

Biber, D., & Conrad, S. (1999). Lexical bundles in conversation and academic prose. In H. Hasselgard & S. Oksefjell (Eds.), *Out of corpora: Studies in honor of Stig Johansson* (pp. 181–189). Amsterdam: Rodopi.

Bollmann, M. (2013). POS Tagging for historical texts with sparse training data. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse* (pp. 11–18). Sofia: ACL.

Borin, L. (2010). Med Zipf mot framtiden – en integrerad lexikonresurs för svensk språkteknologi [With Zipf into the future – an integrated lexical resource for Swedish language technology]. *LexicoNordica* 17, 35–54.

Borin, L. (2012). Core vocabulary: A useful but mystical concept in some kinds of linguistics. In D. Santos, K. Lindén, & W. Ng'ang'a (Eds.), *Shall we play the Festschrift game? Essays on the occasion of Lauri Carlson's 60th birthday* (pp. 53–65). Berlin: Springer. doi:10.1007/978-3-642-30773-7_6

Borin, L., Dannélls, D., Forsberg, M., Gronostaj, M. T., & Kokkinakis, D. (2010a). The past meets the present in Swedish FrameNet ++. In *14th EURALEX International Congress* (pp. 269–281). Leeuwarden: EURALEX.

Borin, L., & Forsberg, M. (2009). All in the family: A comparison of SALDO and WordNet. In *Proceedings of the Nodalida 2009 Workshop on WordNets and other Lexical Semantic Resources – between Lexical Semantics, Lexicography, Terminology and Formal Ontologies* (pp. 7–12). Odense. NEALT.

Borin, L., & Forsberg, M. (2011). A diachronic computational lexical resource for 800 years of Swedish. In C. Sporleder, A. Van Den Bosch, & K. Zervanou (Eds.), *Language technology for cultural heritage* (pp. 41–61). Berlin: Springer. doi:10.1007/978-3-642-20227-8_3

Borin, L., Forsberg, M., & Kokkinakis, D. (2010b). Diabase: Towards a diachronic BLARK in support of historical studies. In *Proceedings of LREC 2010* (pp. 35–42). Valletta: LREC.

Borin, L., Forsberg, M., & Lönngren, L. (2013). SALDO: a touch of yin to WordNet's yang. *Language Resources and Evaluation* 47(4), 1191–1211. doi:10.1007/s10579-013-9233-4

Borin, L., Forsberg, M., & Lyngfelt, B. (2013). Close encounters of the fifth kind: Some linguistic and computational aspects of the Swedish FrameNet ++ project. *Veredas* 17(1), 28–43. doi:10.1007/s10579-013-9233-4

Borin, L., Forsberg, M., Olsson, L. -J., & Uppström, J. (2012). The open lexical infrastructure of Språkbanken. In *Proceedings of LREC 2012* (pp. 3598–3602). Istanbul. ELRA.

Borin, L., Forsberg, M., & Roxendal, J. (2012). Korp – the corpus infrastructure of Språkbanken. In *Proceedings of LREC 2012* (pp. 474–478). Istanbul. ELRA.

Briscoe, T., Carroll, J., & Watson, R. (2006). The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 interactive presentation sessions* (pp. 77–80). Sydney: ACL. doi:10.3115/1225403.1225423

Bryant, J. E. (2008). Best-fit constructional analysis. UC Berkeley PhD dissertation.

Chang, N. (2008). Constructing grammar: A computational model of the emergence of early constructions. UC Berkeley PhD dissertation.

Chang, N., & Maia, T. (2001). Learning grammatical constructions. In J. D. Moore & K. Stenning (Eds.), *Proceedings of the twenty-third annual conference of the Cognitive Science Society* (pp. 176–181). Mahwah: Lawrence Erlbaum.

Copestake, A. (2002). *Implementing typed feature structure grammars*. Stanford: CSLI Publications.

Croft, W. A. (2001). *Radical construction grammar: Syntactic theory in typological perspective*. Oxford: Oxford University Press. doi:10.1093/acprof:oso/9780198299554.001.0001

Dannélls, D., Friberg Heppin, K., & Ehrlemark, A. (2014). Using language technology resources and tools to construct Swedish FrameNet. In *Proceedings of the workshop on lexical and grammatical resources for language processing* (pp. 8–17). Dublin: ACL. doi:10.3115/v1/W14-5802

Fellbaum, C. (Ed.). (1998). *WordNet: An electronic lexical database*. Cambridge, Mass.: MIT Press.

Forsberg, M., Johansson, R., Bäckström, L., Borin, L., Lyngfelt, B., Olofsson, J., & Prentice, J. (2014). From construction candidates to constructicon entries: An experiment using semi-automatic methods for identifying constructions in corpora. *Constructions and Frames* 6(1), 114–135. doi:10.1075/cf.6.1.07for

Francopoulo, G. (Ed.). (2013). *LMF: Lexical markup framework*. London/Hoboken, NJ: ISTE/Wiley. doi:10.1002/9781118712696

Goldberg, A. (1995). *Constructions: A construction grammar approach to argument structure*. Chicago: Chicago University Press.

Grūzītis, N., & Dannélls, D. (2017). A multilingual FrameNet-based grammar and lexicon for controlled natural language. *Language Resources and Evaluation* 51(1), 37–66.

Grūzītis, N., Dannélls, D., Lyngfelt, B., & Ranta, A. (2015). Formalising the Swedish Constructicon in Grammatical Framework. In *Proceedings of the grammar engineering across frameworks (GEAF) workshop* (pp. 49–56). Beijing: ACL. doi:10.18653/v1/W15-3307

Gustafson-Čapková, S., & Hartmann, B. (2006). Manual of the Stockholm Umeå Corpus version 2.0. Stockholm University.

Halácsy, P., Kornai, A., & Oravecz, C. (2007). HunPos – an open source trigram tagger. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics companion volume: Proceedings of the demo and poster sessions* (pp. 209–212). Prague: ACL.

ISO (2008). Language resource management–Lexical markup framework (LMF). *International Standard ISO* 24613.

Johansson, R. (2014). Automatic expansion of the Swedish FrameNet lexicon: Comparing and combining lexicon-based and corpus-based methods. *Constructions and Frames* 6(1), 92–113. doi:10.1075/cf.6.1.06joh

Liberman, M. (2009). The annotation conundrum. In *Proceedings of the EACL 2009 workshop on the interaction between linguistics and computational linguistics: Virtuous, vicious or vacuous?* (p. 2). Athens: ACL. doi:10.3115/1642038.1642040

Lyngfelt, B., Bäckström, L., Borin, L., Ehrlemark, A., & Rydstedt R. (this volume). *Constructicography at work: Theory meets practice in the Swedish Constructicon*.

Lyngfelt, B., Borin, L., Forsberg, M., Prentice, J., Rydstedt, R., Sköldberg, E., & Tingsell, S. (2012). Adding a constructicon to the Swedish resource network of Språkbanken. In *Proceedings of KONVENS 2012* (LexSem 2012 workshop) (pp. 452–461). Vienna: ÖGAI.

Manning, C. D. (2015). Last words: Computational linguistics and deep learning. *Computational Linguistics* 41(4), 701–707. doi:10.1162/COLI_a_00239

Moon, R. (2000). Lexicography and disambiguation: The size of the problem. *Computers and the Humanities* 34(1–2), 99–102. doi:10.1023/A:1002605522910

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryiğit, G., Kübler, S., Marinov, S., & Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(2), 95–135.

Nivre, J., Megyesi, B., Gustafson-Čapková, S., Salomonsson, F., & Dahlqvist, B. (2008). Cultivating a Swedish treebank. In J. Nivre, M. Dahllöf & B. Megyesi (Eds.), *Resourceful language technology: Festschrift in honor of Anna Sågvall Hein* (pp. 111–120). Acta Universitatis Upsaliensis: Studia Linguistica Upsaliensia 7.

Pecina, P. (2010). Lexical association measures and collocation extraction. *Language Resources and Evaluation* 44(1–2), 137–158. doi:10.1007/s10579-009-9101-4

Peldszus, A., & Stede, M. (2013). Ranking the annotators: An agreement study on argumentation structure. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse* (pp. 196–204). Sofia: ACL.

Pullum, G. (2007). Ungrammaticality, rarity, and corpus use. *Corpus Linguistics and Linguistic Theory* 3(1), 33–47. doi:10.1515/CLLT.2007.002

Pullum, G. (2009). Computational linguistics and generative linguistics: The triumph of hope over experience. In *Proceedings of the EACL 2009 workshop on the interaction between linguistics and computational linguistics: Virtuous, vicious or vacuous?* (pp. 12–21). Athens: ACL. doi:10.3115/1642038.1642042

Ranta, A. (2004). Grammatical Framework, a type-theoretical grammar formalism. *Journal of Functional Programming* 14(2), 145–189. doi:10.1017/S0956796803004738

Ranta, A. (2009). The GF Resource Grammar Library. *Linguistic Issues in Language Technology (LiLT)* 2(2).

Reiter, E. (2007). The shrinking horizons of computational linguistics. *Computational Linguistics* 33(2), 283–287. doi:10.1162/coli.2007.33.2.283

Schmid, H. -J. (2010). Entrenchment, salience, and basic levels. In D. Geeraerts & H. Cuyckens (Eds.), *The Oxford handbook of cognitive linguistics* (pp. 117–138). Oxford: Oxford University Press.

Schneider, N. (2010). Computational cognitive morphosemantics: Modeling morphological compositionality in Hebrew verbs with embodied construction grammar. In N. Rolle, J. Steman, & J. Sylak-Glassman (Eds.), *Proceedings of the thirty-sixth annual meeting of the Berkeley Linguistics Society* (pp. 353–368). Berkeley: BLS.

Trosterud, T. (2006). Grammatically based language technology for minority languages. In A. Saxena & L. Borin (Eds.), *Lesser-known languages of South Asia: Status and policies, case studies and applications of information technology* (pp. 293–315). Berlin: Mouton de Gruyter.

Tyers, F., & Pirinen, T. A. (2016). Intermediate representation in rule-based machine translation for the Uralic languages. In *Proceedings of the second international workshop on computational linguistics for the Uralic languages* (pp. 92–104). Szeged: University of Szeged.

Wible, D., & Tsao, N. -L. (2010). StringNet as a computational resource for discovering and investigating linguistic constructions. In *Proceedings of the NAACL HLT workshop on extracting and using constructions in computational linguistics* (pp. 25–31). Los Angeles: ACL.

Wintner, S. (2009). What science underlies natural language engineering? *Computational Linguistics* 35(4), 641–644. doi:10.1162/coli.2009.35.4.35409

Wray, A. (2002). *Formulaic language and the lexicon*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511519772