PRIMING THROUGH CONSTRUCTIONAL DEPENDENCIES
A CASE STUDY IN FLUID CONSTRUCTION GRAMMAR

PIETER WELLENS, JOACHIM DEBEULE
AI-Lab,Vrije Universiteit Brussel, Pleinlaan 2,
Brussels, 1050, belgium
pieter@arti.vub.ac.be, joachim@arti.vub.ac.be

According to recent developments in (computational) Construction Grammar, language processing occurs through the incremental buildup of meaning and form according to constructional specifications. If the number of available constructions becomes large however, this results in a search process that quickly becomes cognitively unfeasible without the aid of additional guiding principles. One of the main mechanisms the brain recruits (in all sorts of tasks) to optimize processing efficiency is priming. Priming in turn requires a specific organisation of the constructions. Processing efficiency thus must have been one of the main evolutionary pressures driving the organisation of linguistic constructions. In this paper we show how constructions can be organized in a *constructional dependency network* in which constructions are linked through semantic and syntactic categories. Using Fluid Construction Grammar, we show how such a network can be learned incrementally in a usage-based fashion, and how it can be used to guide processing by priming the suitable constructions.

1. Introduction

According to Construction Grammar, linguistic knowledge is captured in a constructicon, which is an assembly of form-meaning pairings called constructions (Goldberg, 1995). Processing (i.e. producing or parsing) a sentence amounts to the successive application of constructions, gradually augmenting and transforming an initial meaning to a final form or vice versa.

In each step during processing out of tens of thousands of constructions only a few constructions can apply and of these even less will be correct thus pressuring both efficiency and accuracy. Acknowledging this, most flavors of Construction Grammar adopt a taxonomy of constructions capturing relations of schematicity, but other relations like polysemy, meronomy, inheritance have been proposed as well (see Croft and Cruse (2004) for an overview). Such relations are based on intrinsic properties that hold between the form or meaning of the connected constructions. But constructions can also be related by usage-based properties (i.e. properties that follow first and foremost from their actual use in processing). For example in (Saffran, 2001) it was shown that children are capable of tracking co-occurrence relations not only for word boundaries but also for syntactic patterns.

Such co-occurrence relations can also be captured in a network of constructions, linking them according to conventional usage patterns. Obviously learning these relations can be used to guide and optimize later processing. In addition, if these principles are operating continuously (i.e. at every usage event) this will have a major impact not only on the internal organisation of the constructicon but also on the way the language evolves and changes since it gives rise to self-enforcing loops entrenching patterns more quickly than they otherwise would.

In this paper we operationalize two usage-based construction networks, the second being an extension of the first. Operationalizing requires (1) a learning component that gradually builds and shapes the network on every use and (2) optimized language processing in terms of efficiency and accuracy by utilizing this network. First we introduce a network capturing the fact that certain constructions tend to precede others and show how this can be learned from a series of usage events given a pre-defined set of constructions in Fluid Construction Grammar. We also show how the acquired co-occurrence network combined with language processing capable of priming improves processing performance significantly. A second type of network, the *dependency network*, is based on more subtle and often intricate processing dependencies instead of simple co-occurrences. Such networks allow for an even greater improvement in performance by making the priming much more accurate. We thereby provide the basis for further investigating the influence of conventional constructional usage patterns on the further evolution of language in a computational fashion (e.g. in language game experiments).

2. Learning and using Causal Co-occurrences

In order to capture usage-based dependencies between constructions, we first need to specify in more detail how constructions are used during processing. We use Fluid Construction Grammar (FCG) (De Beule & Steels, 2005; Steels & De Beule, 2006), arguably one of the most advanced formalisms presently available for doing computational construction grammar. FCG has been developed primarily for investigating the emergence and evolution of artificial grammars among autonomous robots. Furthermore, FCG is not a theory of any particular language in that it remains neutral towards what kinds of semantic or syntactic features constitute constructions. As such, it provides the ideal skeletal substrate for implementing and testing a wide variety of empirical and theoretical findings in evolutionary linguistics.

In FCG language processing amounts to finding the correct chain of constructions so that applied in that sequence they will lead to a correct interpretation or production. Determining the appropriate sequence of constructions involves searching through a vast space of possible sequences. If however one construction is often observed to trigger another construction, it makes sense to record this information and use it in later processing. It is important to understand that we are

tracking these co-occurrences at the level of such application sequences and *not* for example at the surface form.

We kept track of causal co-occurrences between constructions as they were applied while processing a thousand randomly generated but valid sentences according to an FCG grammar based on the one documented in (Micelli, Trijp, & De Beule, 2009). The grammar contains 64 lexical constructions for 39 nouns, 18 adjectives, 4 verbs and 3 prepositions, and 16 grammatical constructions, in total amounting to a constructicon of 80 constructions suitable for parsing and producing sentences into and from their Frame Semantic meaning according to FrameNet (Fillmore, 1982; Baker, Fillmore, & Lowe, 1998).

After each sentence is processed, co-occurrences are recorded between the constructions involved, and a constructional causal co-occurrence network is gradually built up. A fragment of the resulting network is shown in Figure 1.

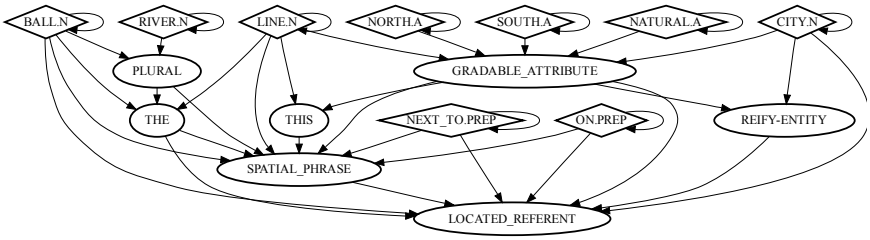

Figure 1. A fragment of the learned causal co-occurrence network. Diamond shaped nodes represent constructions that appeared to be applicable without reliance on other constructions, i.e. only requiring initial meaning and form. Egg-shaped constructions on the other hand were observed to have a causal co-occurrence with other constructions, namely with those connected to them by incoming edges. For example, it was recorded that the plural construction causally co-occurs often with nouns like "ball" and "river", but not yet with other nouns like "line".

As the network is being improved after each new sentence, it is also used to reduce the number of constructions tried for processing the next sentence thus improving efficiency. Figure 2 shows the average number of grammatical constructions considered before an applicable one was found for each sentence.

As the first 300 sentences were processed, the network was being built up, and the number of constructions tried steadily decreased. After this, the average number of tried constructions stabilizes around seven, resulting in a vast improvement compared to the baseline case.

3. Constructional dependencies

Constructional dependencies are a more refined form of causal co-occurrence relationships and build on two related observations. First, in an application chain of constructions most of them can only apply when certain constraints are met. This

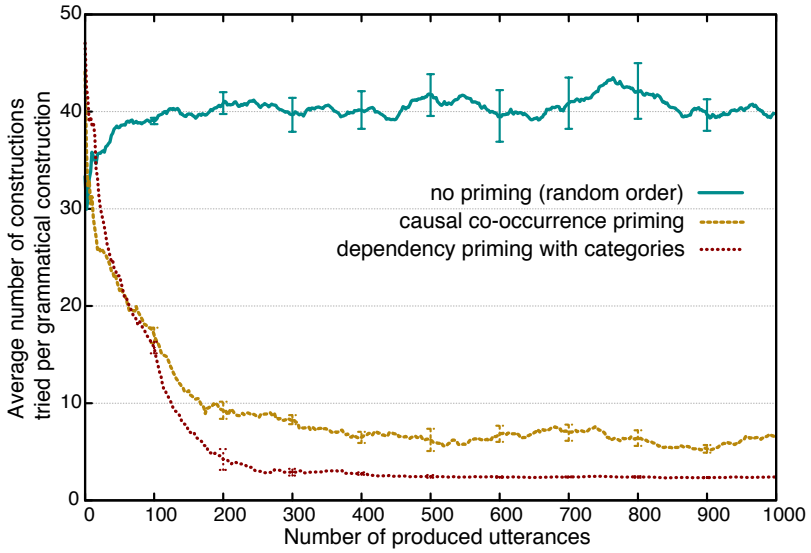

Figure 2. Evolution of the average number of grammatical constructions considered per construction applied while processing a new sentence. The top curve represents the baseline case corresponding to random search, amounting to on average about half of all constructions being tried each time. The middle curve shows how this can be improved by priming constructions according to a causal co-occurrence network. The bottom curve shows another improvement by using dependency based priming, amounting to an additional drop from around seven to around two constructions. This performance is also more stable and learned faster due to generalisation.

holds especially true for grammatical constructions in which these constraints can become quite abstract. For example, in English, the intransitive constructions will, during parsing, only be triggered if a Noun Phrase (NP) is observed directly preceding a Verb. During production, it expresses the fact that the subject NP in a 'Subject Verb' syntactic pattern fills the agentive participant role of the event evoked by the verb. The second and related observation is that these semantic and syntactic constraints are supplied by previously applied constructions. For example whether a constituent is a Noun Phrase or can play the agentive role in an event, and hence whether the intransitive construction will apply, depends on the constructions making up that constituent.

In more general terms, whether construction X should trigger doesn't just depend on what constructions applied before, but more specifically *on the semantic and syntactic categories supplied by these previously applied constructions*. In

this view constructions are thus "communicating" with each other during processing through the categories that they require (from previous constructions) and supply (to later constructions). Categories thus become the main regulators of linguistic processing. This way, dependencies specify a *constructional dependency network* among the constructions in the constructicon.

In FCG the application of a construction involves a *match phase* in which its preconditions are verified and a *merge phase* in which the linguistic feature structure that is being built is modified. It is exactly the interplay of merges of earlier constructions and matches of later constructions that makes the tracking of dependencies possible. Indeed, when a construction matches with the linguistic feature structure only because of an earlier modification by another construction the matching construction is dependent on the earlier one. In pseudocode this is computed as follows:

Function LearnDependencies(applicationChain)

```
Loop
 ForEach laterConstruction in Reverse(applicationChain) do
 // We loop backwards over the applied constructions
  applicationChain ← Remove(laterConstruction, applicationChain);
  dependencyFound ← false;
  ForEach previousConstruction in Reverse(applicationChain) do
    matchedOnByLaterConstruction ← MatchedOn(laterConstruction);
    mergedByPreviousConstruction ← MergedBy(previoudConstruction);
    matchMergeIntersection ← Intersection(matchedOnByLaterConstruction,
                                          mergedByPreviousConstruction);
    If matchMergeIntersection
    then
     // laterConstruction is dependent on previousConstruction
     // either the intersection is already a category or we create it
     category ← FindOrCreateCategory(matchMergeIntersection,
                                     knownCategories);
     AddOrEntrenchDependencyLink(previousConstruction, category);
     AddOrEntrenchDependencyLink(category, laterConstruction);
     dependencyFound ← true;
  End ForEach;
  If dependencyFound == false
  then
   // laterConstruction is INdependent and gets marked as such
   AddOrEntrenchIndependencyLink(laterConstruction);
 End ForEach;
End Loop;
```

End LearnDependencies

The above code loops in reverse order over the applied constructions (i.e. the last one first) and checks for this construction what it matched on (i.e. its preconditions). Then it loops again over all constructions applied before this construction and checks how these construction applications modified the feature structure by their merge. If an intersection is found between this match and merge then a dependency is found. Essentially it means the later construction would not have been able to apply if it was not for the modifications of the earlier construction. When such a dependency is found either a new edge is added between the constructions or if there already is one it entrenches it more by incrementing its score. If no

dependency is found then this is also recorded because it means the construction could be applied without any previous construction modifying the initial feature structure.

Capturing these sort of dependencies requires that not only the constructions are explicitly represented in the network but also the semantic and syntactic categories that constitute the dependencies. This is illustrated in Figure 3.

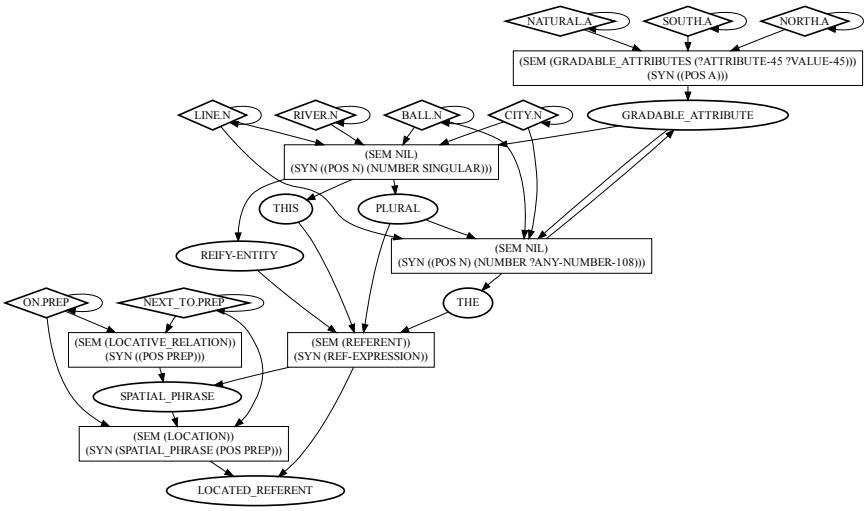

Figure 3. Fragment of the constructional dependency network learned from the exact same usage history as in Figure 1, but this time also capturing more subtle inter-dependencies between constructions besides causal co-occurrences. Square nodes hold semantic and syntactic categories as provided or required by constructions.

Explicitly representing the categories has the advantage that it makes priming of constructions much more accurate and at the same time more general. Consider for example the GRADABLE_ATTRIBUTES construction in Figures 1 and 3. It requires an adjective and a noun and assembles them in an Adjective-Noun phrase. In the simpler network of Figure 1, all nouns and adjectives that occurred together prime the construction, meaning that on the one hand it is primed too often (because it is primed *whenever* one of these nouns is observed, even if no adjective is observed yet), but on the other hand also that it is not primed often enough, because only previously encountered nouns and adjectives prime it. In the augmented network of Figure 3 however, the GRADABLE_ATTRIBUTES construction is primed only if *both* a noun and an adjective are observed, and even if they were not encountered before with this particular construction; as long as they are known in the network to be a noun and an adjective. The same effect is also observed in children when they cue in on syntactic information to produce novel utterances

with nonce words (Tomasello, Akhtar, Dodson, & Rekau, 1997).

In the current model, the effect can also be illustrated by comparing the linkage of the LINE.N construction with the PLURAL construction in both networks. In the simple network, LINE.N and PLURAL are not yet connected, meaning that no sentence was ever encountered containing the plural form "lines". LINE.N is however linked to the THE construction, meaning that the phrase "the line" was observed. In the augmented network, observing "the line" connects LINE.N to THE through the intermediate noun-like category node. This node is in turn connected to PLURAL due to *another* observation *not* involving LINE.N. This way, the dependency between the LINE.N and the PLURAL constructions has also been captured without having observed it.

Dependency based priming allows an additional improvement of processing performance as can be seen in Figure 2. This time, on average only two out of eighty constructions need to be considered before a good one is found. This number is also more stable compared to a co-occurrence network.

4. Discussion and conclusion

There is a growing body of evidence supporting the hypothesis that acquiring a language involves the learning of usage-based dependency patterns among constructions (Tomasello, 1992; Saffran et al., 2008). There is also evidence that acquired patterns of constructional usages influence language processing, for example through the priming of frequently co-occurring constructions (Tomasello et al., 1997; Saffran, 2001).

In this paper we have shown how these two observations can be operationalized by performing a case study in Fluid Construction Grammar involving the learning of constructional dependency networks from randomly generated but valid sentences according to an FCG grammar documented in (Micelli et al., 2009). It was shown that such networks can be learned and allow to reduce the amount of processing required for parsing or producing a sentence. In addition, we have proposed to explicitly include semantic and syntactic categories in the network, providing the glue between constructions, and have shown how this leads to a powerful capacity to generalize from observations and an associated further reduction of processing load.

The processing efficiency and accuracy in humans is nothing short of amazing and the mechanisms regulating our capacity for language must have been under these evolutionary pressures at all times. If not it would have become to slow to use or could not expand to more than a limited lexicon. The model and results presented here show that language does indeed contain structure that allows one to build and shape a constructional dependency network which can be used to optimize both efficiency and accuracy. Moreover there is a non trivial connection between the dependencies we tracked in our networks and the hierarchical structure of language thus hinting that recruitment of these general cognitive ca-

pabilities might be a necessary requirement to learn and process hierarchical large scale language systems (Steels, 2007).

Moreover, if learning a language involves learning dependency patterns, and if using language involves employing the learned patterns, then language transmission, and hence language evolution, will also be influenced by dependency patterns. By operationalizing the learning and usage of dependency patterns, we have therefore provided a basis for investigating the role of dependency patterns in the evolution of language.

References

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on computational linguistics.* Morristown, NJ, USA: Association for Computational Linguistics.

Croft, W., & Cruse, A. (2004). *Cognitive linguistics.* Cambridge: Cambridge University Press.

De Beule, J., & Steels, L. (2005). Hierarchy in Fluid Construction Grammar. In U. Furbach (Ed.), *Ki 2005: Advances in artificial intelligence. proceedings of the 28th german conference on ai* (Vol. 3698, pp. 1–15). Berlin: Springer.

Fillmore, C. J. (1982). Frame semantics. In *Linguistics in the morning calm* (pp. 111–137). Seoul.

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

Micelli, V., Trijp, R. van, & De Beule, J. (2009). Framing fluid construction grammar. In N. Taatgen & H. van Rijn (Eds.), *the 31th annual conference of the cognitive science society* (p. 3023-3027). Cognitive Science Society.

Saffran, J. (2001). The use of predictive dependencies in language learning. *Journal of Memory and Language*, *44*, 493-515.

Saffran, J., Hauser, M., Seibel, R., Kapfhamer, J., Tsao, F., & Cushman, F. (2008). Grammatical pattern learning by human infants and cotton-top tamarin monkeys. *Cognition*, *107*, 479-500.

Steels, L. (2007). The recruitment theory of language origins. In C. Lyon, C. Nehaniv, & A. Cangelosi (Eds.), *Emergence of communication and language* (p. 129-151). Berlin: Springer Verlag.

Steels, L., & De Beule, J. (2006). Unify and merge in Fluid Construction Grammar. In P. Vogt, Y. Sugita, E. Tuci, & C. Nehaniv (Eds.), *Symbol grounding and beyond.* (pp. 197–223). Berlin: Springer.

Tomasello, M. (1992). *First verbs: A case study of early grammatical development.* Cambridge: Cambridge University Press.

Tomasello, M., Akhtar, N., Dodson, K., & Rekau, L. (1997). Differential productivity in young children's use of nouns and verbs. *Journal of Child Language*, *24*, 373-87.