Evaluation Strategies for Computational Construction Grammars

Tânia Marques
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB United Kingdom
tmarques@inf.ed.ac.uk

Katrien Beuls
Artificial Intelligence Lab
Vrije Universiteit Brussel
Pleinlaan 2 B-1050 Brussels Belgium
katrien@ai.vub.ac.be

Abstract

Despite the growing number of Computational Construction Grammar implementations, the field is still lacking evaluation methods to compare grammar fragments across different platforms. Moreover, the hand-crafted nature of most grammars requires profiling tools to understand the complex interactions between constructions of different types. This paper presents a number of evaluation measures, partially based on existing measures in the field of semantic parsing, that are especially relevant for reversible grammar formalisms. The measures are tested on a grammar fragment for European Portuguese clitic placement that is currently under development.

1 Introduction

Computational Construction Grammar allows computational linguists to formalize their hypotheses and intuitions about certain linguistic phenomena and explore how these representational choices affect the processing of natural language utterances (Schneider and Tsarfaty, 2013). In this sense, it follows in the footsteps of Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), Lexical Functional Grammar (Bresnan et al., 2016), Head-Driven Phrase-Structure Grammar (HPSG) (Pollard and Sag, 1994) and Combinatory Categorial Grammar (CCG) (Steedman, 2000). Yet, different from these other approaches, it adheres to the principles of Construction Grammar (CxG) (Goldberg, 2005; Östman and Fried, 2005; Hoffmann and Trousdale, 2013).Therefore, constructions are treated as first-class citizens in the grammatical organisation of a language. They are viewed as learned mappings between form (sounds, morphemes, syntactic categories) and function (semantics, pragmatics, etc.). Because there is no strict separation between the lexicon and the grammar, semi-productive idioms like "X let alone Y" are treated in the same way with lexemes and core syntactic patterns (Fillmore et al., 1988). We can distinguish three main computational frameworks that are currently active within the Construction Grammar community: Embodied Construction Grammar – ECG (Bergen and Chang, 2005), Fluid Construction Grammar – FCG (Steels, 2004; Steels, 2011) and more recently Template Construction Grammar – TCG (Barrès and Lee, 2014).

The evaluation of computational construction grammars is currently not reaching further than proof-of-concept grammar fragments that show how to implement a certain language phenomenon and demonstrate the resulting grammar by means of web demonstrations or its use in a simulation-based robotic environment (Trott et al., 2015). There are two reasons for the lack of widely used evaluation metrics in CxG: (i) Different from data-driven approaches, construction grammars are not built automatically from annotated treebanks and therefore do not reach a wide coverage in the traditional sense. Instead, both ECG and FCG allow the grammar writer to test a number of sentences automatically when loading the grammar fragment and return the average base parse for these (geometric mean of the number of parses per sentence). (ii) Different from syntactic parsers that concentrate on the syntactic accuracy of the syntax trees that their grammars derive, computational construction grammars focus on semantic accuracy as a metric that better meets their objectives. However, semantic accuracy is harder to measure than syntactic accuracy because it requires textual corpora annotated with large formal meaning representations that are agreed upon by different grammar developers.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1137–1146, Osaka, Japan, December 11-17 2016.

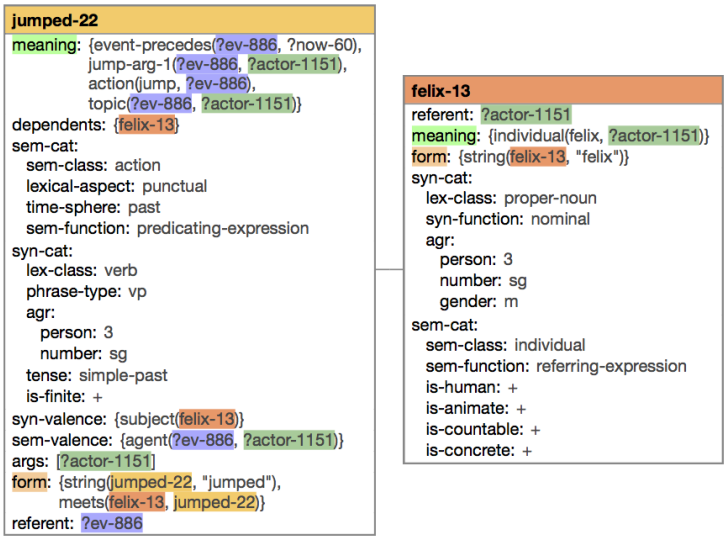

Figure 1: Resulting transient structure after parsing "Felix jumped". Features cut across argument structure, information structure and functional structure.

The field of computational CxG is reaching a certain level of maturity in terms of linguistic phenomena that are treated and publications that accompany grammars (e.g. (van Trijp, 2015) for FCG and (Trott et al., 2015) for ECG), and has clearly demonstrated the feasibility of implementing the constructional approach in a full-fledged computational framework (Schneider and Tsarfaty, 2013). The time has come to evaluate these implementations so that they can be compared within the field and benchmarks can be created. We therefore suggest a number of metrics for parsing (comprehending an utterance into a meaning representation) and production (formulating an utterance from a meaning representation) of a test suite that addresses a specific linguistic phenomenon. Because computational CxG fragments are hand-crafted precision grammars rather than data-driven approaches, broad-coverage newspaper corpora are not interesting test beds for evaluation since they are not constrained towards the phenomena of study. Rather, we suggest to compile a test suite of sentences taken from linguistic research in the area that the grammar is focusing on.

Before explaining the actual metrics we designed, our contribution first argues, in Section 2, for the usefulness of including CxG in computational linguistics. Section 3 then draws parallels between the fields of semantic parsing and computational construction grammar in terms of semantic representations, before Section 4 presents metrics we propose to evaluate grammars in the latter formalism. The case study then shows the use of these metrics in a sample grammar for European Portuguese in Section 5. Finally, Section 6 concludes.

2 Why Constructions?

Computational CxG views production and comprehension in terms of a chain of consecutive operations over a linguistic structure, called the transient structure, a feature structure consisting of units that are made up of non-typed feature-value pairs, which maintains a temporary state of knowledge. A sequence of transient structures on a particular execution chain is called a linguistic pathway (Steels, to appear). One of the characteristics of CxG is its "insistence on simultaneously describing grammatical patterns and the semantic and pragmatic purposes to which they are dedicated" (Fillmore et al., 1988) so transient structures, as well as constructions, need to be able to represent information from a multitude of different perspectives. A construction schema, or in short, a construction, is an abstract schema that can be used to "expand any aspect of a transient structure from any perspective and it can consult any aspect of this transient structure to decide how to do so" (Steels, to appear).

The result of construction application is thus a transient structure from which semantic or formal information can directly be extracted, without additional steps. Figure 1 shows the resulting transient

structure after parsing the intransitive sentence "Felix jumped" with a dependency-style grammar. The meaning features of both units form together the complete understanding of the sentence. Through variable equalities, Felix is linked to the first argument of the jump action and the topic of the sentence. Of course, the meaning predicates in this example are merely illustrative here and are subject to choices that the grammar engineer makes.

Not only the resulting transient structure can be valuable in the evaluation procedure but also the constructions that built the structure should be considered. In the case of "Felix jumped", a two-word utterance, at least five constructions have been at work to construe its interpretation. Two lexical constructions covering the words themselves, one tense construction to situate the event in the past (and indicate that it is not the adjective "jumped"), one argument structure construction linking the Felix to the actor or the jumping event (intransitive) and one information structure construction identifying "Felix" as the topic of the sentence.

3 Semantic Parsing

Semantic parsers – similar to Computational Construction Grammar implementations – are not interested in building well-formed syntactic trees but instead map sentences into formal meaning representations (Mooney, 2007). They are used in domains such as question answering, where natural-language questions are converted into formal queries, and are typically built over databases with unsupervised (e.g. (Poon and Domingos, 2009)) or supervised (e.g. (Berant et al., 2013)) learning algorithms. To go beyond simple query formulation towards complex knowledge extraction, a recent approach by (Parikh et al., 2015) uses distant supervision methods to learn a semantic parser from a database of complex events and unannotated texts.

While testing, the retrieved meanings are compared to a gold standard annotation to calculate the precision and recall of the parser. Three main ways to calculate precision and recall can be distinguished, ranging from less to more fine-grained analyses:

1. The retrieved meaning/formulated query is correct when it matches the gold standard. Recall is then the number of correct meanings divided by the number of sentences. Precision is the number of correct meanings divided by the number of sentences for which the parser produced a meaning. An example of a system that employs this measure is the Cocktail system (Tang and Mooney, 2001).

2. Instead of using a binary measure, one can calculate the overlap in attribute-value pairs between the retrieved meaning and the gold standard. Recall is then the percentage of recovered attribute-value pairs per sentence, averaged over the test set. An example of a semantic parser that calculates this overlap is the (Zettlemoyer and Collins, 2007) parser for the ATIS flight info domain.

3. The Smatch score (Cai and Knight, 2013) measures the overlap between meaning representations while taking into account variable bindings. It is determined by calculating the maximum possible F-scores for alternative bindings. This calculation process can be seen in Table 1 for a small exemplifying sentence "a boy is", with the following gold standard and parsed meanings (in Abstract Meaning Representation):

   gold standard meaning: *instance(?x, boy)* $\wedge$ *attribute(?x, single)* $\wedge$ *instance(?y, be)* $\wedge$ *attribute(?y, currently-being)* $\wedge$ *is(?y, ?x)*

   parsed meaning: *instance(?a, boy)* $\wedge$ *attribute(?b, single)* $\wedge$ *instance(?b, be)* $\wedge$ *is(?a, ?a)*

	Matched	Precision	Recall	F-Score
x = a y = b	2	2/4	2/5	0.44
x = b y = a	1	1/4	1/5	0.22
			S-score:	0.44

Table 1: Calculation of the normal Smatch score.

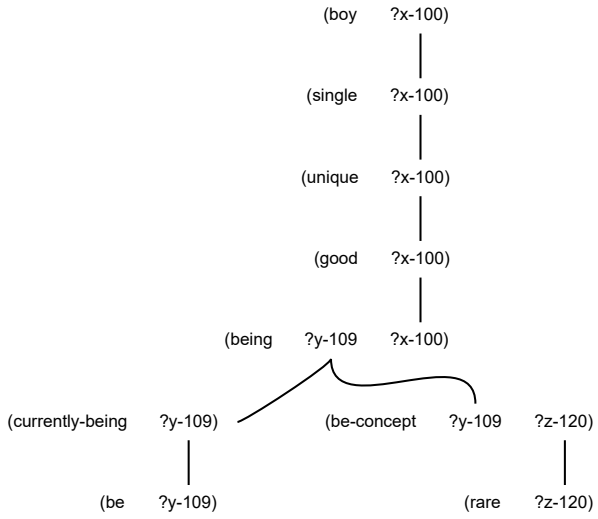

Figure 2: Fully connected meaning after parsing the sentence "A good boy is rare".

Given the existing variables, there are two possible bindings. The *?x* is either bound to *?a* or *?b*, and the same is true for *?y*. There are 5 relations in the gold standard, but only 4 in the generated meaning. Replacing the variables with *?x = ?a; ?y = ?b* gives two matches between the meanings: *instance(?x, boy) = instance(?a, boy)* and *instance(?y, be) = instance(?b, be)*. This makes the precision 2/4 and the recall 2/5, leading to an F-score of 0.44, which is the maximum amongst all the possible bindings. Therefore, 0.44 is also the Smatch score (S-score) for the two meanings.

The S-score is interesting for evaluating construction grammars, because the meaning representation obtained in Fluid Construction Grammar can be directly converted to Abstract Meaning Representation (AMR). AMR has two types of relations: a relation between a concept and a variable (instance and attribute relations); and a relation between two variables (argument relations). The AMR version of the meaning network in Figure 2 would become: *instance(?x, boy)* $\wedge$ *attribute(?x, single)* $\wedge$ *attribute(?x, unique)* $\wedge$ *attribute(?x, good)* $\wedge$ *instance(?y, be)* $\wedge$ *attribute(?y, currently-being)* $\wedge$ *arg(?y, ?x)* $\wedge$ *arg (?y, ?z)* $\wedge$ *instance(?z, rare)*. All the concepts related to the "boy" use the same variable *?x*, and the concept "rare" is connected with the verb "be", which is happening at present and corresponds to the being *?x*, "boy" which is *?z*, "rare". The nodes in the graph are second order logics predications, where properties and relations can be objects as well. For instance, the meaning of "a good boy is rare" could be: *(boy ?x)* $\wedge$ *(single ?x)* $\wedge$ *(unique ?x)* $\wedge$ *(good ?x)* $\wedge$ *(be ?y)* $\wedge$ *(currently-being ?y)* $\wedge$ *(being ?y ?x)* $\wedge$ *(be-concept ?y ?z)* $\wedge$ *(rare ?z)*. All the concepts related to the "boy" use the same variable *?x*, and the concept "rare" is connected with the verb "be", which is happening at present and corresponds to the being *?x*, "boy" which is *?z*, "rare".

4 Proposed Metrics

The remainder of this section explains the two accuracy metrics in more detail. For comprehension, a variant of the Smatch score is proposed that takes into account subparts of the meaning graph. For production, we present the longest common substring measure, with two variants. Finally, a single profiling measure is included to quantify the efficiency of the grammar in comprehension and production.

Reinterpreting Smatch Let us consider that we were evaluating a hypothetical precision grammar with the goal of identifying nouns and their modifiers, but that lacked any constructions for verbs. Then, the S-score of 0.44 (obtained in Table 1), while important for understanding the broad accuracy of the grammar, might not really tell us how the grammar is doing in terms of identifying the nouns. For expressing this in an explicit way, we might also want to have a specific smatch score to calculate the accuracy only for this phenomena. Unfortunately, due to the relational nature of the different parts of the

sentence, it is not possible to separate what is being studied from what is not, since the identification of its relations is also important.

A compromise between what is studied and what we can measure was found for enabling us to calculate a more specific S-score, while keeping the same process of finding the maximum F-score. This is done by annotating the variables in the gold standard that are more important for the phenomena being studied. In this case, it would be *x*. Then, we disregard any relation that does not contain this variable in it, and we accept partial matchings. Even, if *is(?y, ?x)* did not totally match, but the noun was identified in the correct position, it would still be counted. Table 2 shows the calculation for the same sentence ("a boy is"), but considering only the variable *x*. There are three relations to be considered in the gold standard and two in the meaning obtained, regardless of the variable bindings. The matches are still two, but the precision and recall increase, leading to a S-score of 0.8

	Matched	Precision	Recall	F-Score
x = a y = b	2	2/2	2/3	0.8
x = b y = a	1	1/2	1/3	0.4
			S-score:	0.8

Table 2: Calculation of the specific Smatch score.

The S-score obtained for a specific phenomenon should not be presented on its own, because it would lead to an erroneous understanding of the grammar accuracy over the whole corpus. However, together they give a better understanding how the grammar is working. For instance, in this example, the grammar only parses this sentence into its meaning with a 0.44 accuracy, yet the noun identification is mostly correct, which is what actually tell us if the grammar is working for what it was proposed.

Calculating reproducibility The FCG grammar has an additional problem that is usually not faced in semantic parsers: the fact that the full meaning of a sentence is correctly obtained does not necessarily mean that the original utterance can be reproduced. FCG does not work with templates or libraries of sentences, instead the sentence will be reproduced based on the syntactic aspects of the grammar. While word order does not pose any problems in parsing and can indeed help to guide the comprehension process, it can become an issue in production that gets worse when sentence length increases. It is also particularly hard in languages that do not follow a rigid word order, where a different but correct sentence can also be produced without any change in meaning.

Measures that evaluate the correctness of the sentence that is produced are essential to have a complete understanding of the accuracy of FCG due to its bidirectional nature. The most obvious measure would be to compare the sentence obtained with a set of possible acceptable sentences that can be generated back with the same meaning. However, this binary measure does not convey much information. Instead, we propose to use the Longest Common Subsequence (LCS) algorithm to obtain the percentage of the sentence that was correctly generated. LCS is an algorithm that given two sequences, $X = [x_1, x_2, .., x_i]$ and $Y = [y_1, y_2, .., y_j]$, can find the maximum length subsequence between them, defined as a strict increasing sequence of indices of X $[1, 2, .., k]$ such that $x_{ij} = z_j$ (Cormen, 2009). The number of words in the maximum length common subsequence divided by the number of words in the original acceptable sentences, will then gives us an estimation of the percentage of the sentence that the grammar was able to generate back. This measure is not new, and has been previously used to evaluate machine translated texts by (Lin and Och, 2004).

Because the comprehension process might lead to partial meanings, we distinguished between two variants of the LCS measure. One for sentences that were successfully parsed, leading to a fully connected network and one for the unsuccessful ones that might only lead to the production of partial sentences.

Grammar efficiency Construction application may generate a large search. Multiple constructions can expand the same transient structure at a certain time step, or a single construction can expand the transient structure in more than one way. Splits in the search tree can be the result of ambiguities in processing, but

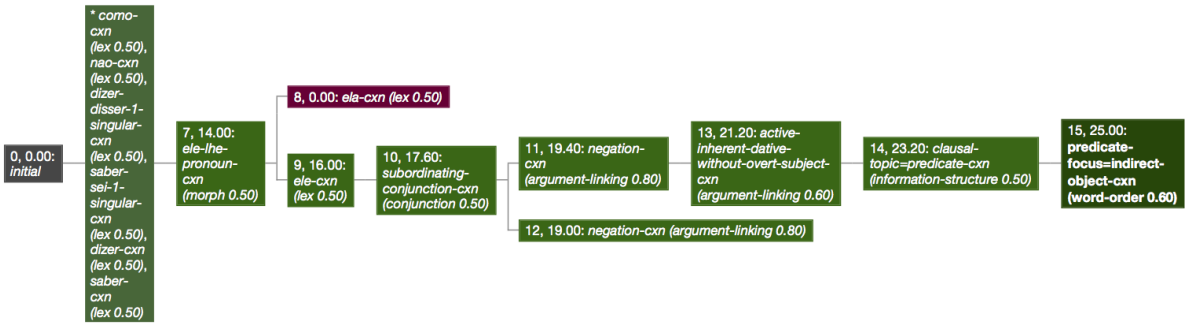

Figure 3: The comprehension process of "não sei como disser-lhe" has an efficiency of 13/14 (not considering the red node, where the second unification step failed).

often originate in sloppy grammar design. Measuring grammar efficiency (at least in terms of additional nodes generated in the search tree) is done by dividing the number of search nodes in the branch that leads to the solution (when all goal tests succeed) by the total number of nodes in the tree. When efficiency equals to 1, no additional nodes are created. The grammar efficiency of the comprehension example in Figure 3 equals 13/15. We see that one of the two additional nodes is red, indicating that it passed the first unification step (conditional part) but failed when unifying the contributing part of the construction. This node do not create additional search since they have no further children. A more informative grammar efficiency measure would perhaps only consider the succeeded nodes. In that case, the example sentence "não sei como disser-lhe" has an efficiency of 13/14 (0.93).

5 Case Study

To better understand how one can interpret the proposed metrics and how they differ from traditional approaches, we include the evaluation of an European Portuguese (EP) grammar fragment that is under development in FCG. It should be noted, however, that this grammar fragment is only here to exemplify how the metrics can be used during grammar developing and evaluation phase. We do not make any claims regarding the grammar itself. The chosen fragment focuses on pronominal clitics. Clitics in EP can be positioned following the verb (enclisis) or preceding the verb (proclisis). Correct clitic placement does not depend on the finiteness of the verb (as in other Romance languages) but is instead determined by the phrasal context. Hence, the coverage of different contexts is an important consideration in the evaluation of such a grammar. Therefore, we built our own test suite by collecting 67 sentences from linguistic research papers dedicated to this phenomenon (Madeira, 1992; Luís et al., 2004; Luís and Otoguro, 2011), and annotated them manually. To create the grammar, lexical constructions were automatically generated from the test suite words based on their grammatical categories, but the core grammatical constructions were hand-crafted considering the linguistic concept being studied, and do not represent necessarily all grammatical intricacies present in the test suite sentences.

The normal S-score gives us an F-score of 0.75 ± 0.21, while the specific S-score returns a slightly higher value of 0.79 ± 0.32. These numbers tell us that the grammar is not covering all necessary aspects to comprehend all the sentences but it is better in the clitics placement than it would seem, as the specific S-score is higher than the general one. To get a better understanding of which parts of the grammar are still not satisfactory, the example sentences were annotated with the proclisis trigger they contain, or else tagged as enclisis. Table 3 shows the average S-scores for seven triggers, together with their standard deviation. Between brackets the number of example sentences is shown. The operator adverb and the relative clauses results exemplify the biggest discrepancies between the two S-scores: 0.62 vs 0.84 and 0.63 vs. 0.80, respectively. This shows that while the grammar fragment is not very good at handling the sentences containing this concepts, the position of the clitics is still correctly identified. Looking into the sentences, we understand that they have more complex verbal forms that are not well processed by the grammar because they were not implemented. The opposite situation does also occur where the general score is higher than the specific. For instance, the enclisis results present a F-score of 0.75 vs 0.73, which

Clitics Position	Triggers	S-score		LCS	
		Normal	Specific	Successful	Unsuccessful
enclisis (28)	none (28)	0.75 ± 0.20	0.73 ± 0.38	0.87 ± 1.43	0.49 ± 0.63
proclisis (39)	negation (6)	0.77 ± 0.25	0.81 ± 0.79	1.00 ± 0.00	N/A
	wh-question (5)	0.88 ± 0.18	0.86 ± 0.28	0.90 ± 0.12	0.33 ± 0.00
	relative clause (7)	0.62 ± 0.08	0.84 ± 0.25	0.43 ± 0.68	0.37 ± 0.10
	fronted focus (8)	0.70 ± 0.22	0.88 ± 0.23	0.84 ± 0.33	0.42 ± 0.01
	operator adverb (3)	0.63 ± 0.23	0.80 ± 0.28	0.40 ± 0.00	0.20 ± 0.00
	undefined-subject (3)	0.89 ± 0.09	0.89 ± 0.16	1.00 ± 0.00	N/A
	downward quantifier (7)	0.78 ± 0.20	0.79 ± 0.30	0.74 ± 0.63	0.14 ± 0.30

Table 3: Accuracy and reproducibility results for the EP grammar case study. All the results are presented by the concepts being studied, the proclisis triggers.

means that although it seems to perform generically well in those sentences, the positioning of the clitics is incorrect more often than predicted by the general score.

Table 3 also includes the Longest Common Substring results. Two scores are kept: the leftmost column includes the LCS for sentences that were produced from meanings that passed all goal tests (and have thus a higher chance of success). The rightmost column shows the scores for productions whose initial meaning networks were extracted from a comprehension process that failed. No score is shown when the case did not happen. The biggest issue in getting the word order right seems to occur in the relative clauses (0.43) and the operator adverbs (0.40). The latter allow multiple grammatically correct word orders in EP, whereas we only include the first solution.

When it comes to grammar efficiency, Figure 4 (on the left) shows the results ordered by sentence length (3–7). We see indeed that the efficiency does not scale well when longer sentences are parsed (and the same goes for production). Less than 20% of all search nodes are used by the branch that leads to the solution. To get an idea of the complexity of the grammar Figure 4 (on the right) plots the number of constructions that is needed to parse a sentence. Sentences of length 3 require on average 9 constructions and the number increases with steps of 2 with every word that is added.

The grammar fragment presented in this section covers a very basic grammatical phenomenon of positioning the clitics correctly. Yet, being basic it has several intrinsic aspects that are fundamental to get right to process it correctly, especially if we want to have grammars that generate grammatically correct sentences. A traditional metric of accuracy or even the normal S-score metric would give an unfair comparison between this fragment grammar and a corpus-based grammar. While, the latter might be better at covering all the sentences provided in a corpus or in a test suite, it might always fail in the processing of this phenomena. While the former grammar not being constructed for a wide coverage

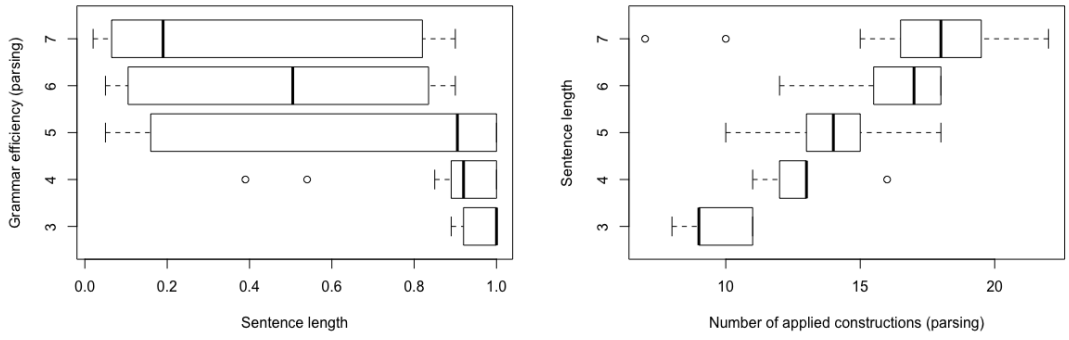

Figure 4: The efficiency and complexity of the EP grammar in function of the sentence length.

might be better at handling this phenomena. The specific s-score when used together provides us with further information to understand how the grammar differs in a more detailed level. Furthermore, this score and the additional profiling metrics for efficiency are very useful for grammar engineers during the grammar developing phase to understand where their grammar falls short and should be further developed. The LCS in FCG provides a further insight into the accuracy of the generated sentences which tells us how well the grammar understands and generates the sentence back.

Using the AMR annotation for evaluating grammar is a relatively costly process. However, it is becoming increasingly necessary to have a semantically annotated corpus in addition to the more traditional syntactical tree-based annotation. Furthermore, there are some strategies that can decrease the burden on the annotators: (1) if only a partial phenomena is being studied, then it might be reasonable to have only partial annotations; (2) it is possible to distribute the phenomena per annotator, thus decreasing the learning curve; (3) it is possible to provide the sentences already generated by the grammars and ask the annotators to fix the errors based on their cognitive understanding of the meaning and/or a set of rules provided.

6 Conclusions

Taking inspiration from existing measures in semantic parsing and machine translation, we proposed two new metrics for evaluating computational Construction Grammar implementations: the S-score, with a variant for focusing on the parts of the meaning graph that are tackled by the grammar fragment; and the LCS score, indicating the reproducibility of the retrieved meaning network. Used in addition to more traditional metrics, these scores give insights about the exact type of phenomena that can be handle by a precision grammar, which is important to distinguish grammars that cover a large number of sentences but invariably fail in processing specific phenomena and grammars that cover a small set of sentences but can deal well with a specific phenomena. Additionally, some profiling measures are also suggested to give an idea of the grammar efficiency and complexity. We hope the proposed metrics help grammar engineers to better understand the complex interactions between the constructions in their grammars and the phenomena being covered by it.

Acknowledgements

The research presented in this paper has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 607062 *ESSENCE: Evolution of Shared Semantics in Computational Environments* (`http://www.essence-network.com/`).

References

Victor Barrès and Jinyong Lee. 2014. Template construction grammar: from visual scene description to language comprehension and agrammatism. *Neuroinformatics*, 12(1):181–208.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In Timothy Baldwin and Anna Korhonen, editors, *Conference on Empirical Methods on Natural Language Processing*, pages 1533–1544, Seattle, Washington. The Association for Computational Linguistics.

Benjamin Bergen and Nancy Chang. 2005. Embodied construction grammar in simulation-based language understanding. In Jan-Ola Östman and Mirjam Fried, editors, *Construction grammars: Cognitive grounding and theoretical extensions*, number 3 in Constructional Approaches to Language, pages 147–190. John Benjamins, Amsterdam.

Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2016. *Lexical-Functional syntax*. Wiley-Blackwell, West-Sussex, England, 2 edition, August.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013*, volume Volume 2: Short papers, pages 748–752, Sofia, Bulgaria, 4-9 August 2013.

Thomas H. Cormen. 2009. *Introduction to algorithms*. MIT Press, Boston, MA.

Charles J Fillmore, Paul Kay, and Mary Catherine O'Connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of let alone. *Language*, 64(3):501–538.

Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Basil Blackwell, Oxford.

Adele Goldberg. 2005. *Constructions at work*. Oxford University Press, Oxford.

Thomas Hoffmann and Graeme Trousdale, editors. 2013. *The Oxford handbook of Construction Grammar*. Oxford University Press, Oxford.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 605–612, Barcelona, Spain, July. ACL.

Ana Luís and Ryo Otoguro. 2011. Inflectional morphology and syntax in correspondence: Evidence from european portuguese. In Glyn Hicks Alexandra Galani and George Tsoulas, editors, *Morphology and Its Interfaces*, number 178 in Linguistik Aktuell/Linguistics Today, pages 97–136. John Benjamins, Amsterdam.

Ana Luís, Ryo Otoguro, Miriam Butt, and Tracy Holloway King. 2004. Proclitic contexts in european portuguese and their effect on clitic placement. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG'04 Conference*, pages 334–352, Stanford, CA. CSLI Publications.

Ana Maria Madeira. 1992. On clitic placement in European Portuguese. *UCL Working Papers in Linguistics*, 4:95–122.

Raymond J. Mooney. 2007. Learning for semantic parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 4394 of *Lecture Notes in Computer Science*, pages 311–324, Mexico City, Mexico, February 18-24. Springer.

Jan-Ola Östman and Mirjam Fried. 2005. *Construction Grammars: Cognitive grounding and theoretical extensions*, volume 3 of *Constructional Approaches to Language*. John Benjamins, Amsterdam.

Ankur P Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 756–766, Denver, Colorado, June 5. Association for Computational Linguistics.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press, Chicago.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 1–10, Singapore, August 6-7. Association for Computational Linguistics.

Nathan Schneider and Reut Tsarfaty. 2013. Book review: Design patterns in Fluid Construction Grammar. *Computational Linguistics*, 39(2):447–453.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Boston, MA.

Luc Steels. 2004. Constructivist development of grounded construction grammars. In Walter Daelemans, editor, *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 9–19, Barcelona. Association for Computational Linguistics.

Luc Steels, editor. 2011. *Design patterns in Fluid Construction Grammar*. Number 11 in Constructional Approaches to Language. John Benjamins, Amsterdam.

Luc Steels. to appear. The basics of Fluid Construction Grammar. *Constructions and Frames*.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477, Freiburg, Germany. Springer.

Sean Trott, Aurélien Appriou, Jerome Feldman, and Adam Janin. 2015. Natural language understanding and communication for multi-agent systems. In *Artificial Intelligence for Human-Robot Interaction Papers from the AAAI 2015 Fall Symposium*. AAAI.

Remi van Trijp. 2015. Cognitive vs. generative construction grammar: The case of coercion and argument structure. *Cognitive Linguistics*, 26(4):613–632.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 678–687, Prague, June.