Inducing grammar from IGT

Lars Hellan  Dorothee Beermann
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
{lars.hellan, dorothee.beermann}@ntnu.no

Abstract
We suggest a strategy for incremental construction of deep parsing grammars from Interlinear Glossed Text (IGT). IGT is a format of representation where standard linguistics and NLP in principle meet, since they are a data-type which is often available for digitally ‘less resourced languages’ (‘LRL’). The IGT database is TypeCraft (Beermann and Mihaylov 2009, www.typecraft.org), and the grammar technology so far employed is that defined in the LKB system of (Copestake 2002), an implementation of the HPSG grammar framework.

Keywords: grammar induction, multilingual deep annotation, IGT

1. Introduction
We suggest a strategy for incremental construction of deep parsing grammars from Interlinear Glossed Text (IGT). IGT is a format of representation where standard linguistics and NLP, and of particular interest is that IGT data exist also for languages with no or little digital resources. The present paper describes a strategy for scaling up grammar development for ‘less resourced languages’ (‘LRL’). Introduced as a method for tackling valency frames for Germanic languages, the general strategy has also been applied to a Kwa and an Ethio-Semitic language, respectively Ga (ISO 639-3 gaa; Hellan and Dakubu 2010), and Kistaninya (ISO 639-3 gru; Wakjira 2010).

The IGT are taken from the TypeCraft database (Beermann and Mihaylov 2009, www.typecraft.org), and the grammar technology employed is that defined in the LKB system of (Copestake 2002), an implementation of the HPSG grammar framework (cf. Pollard and Sag 1994, Sag et al. 2003). The IGT specifications are channeled from the database via XML to pertinent modules of the computational grammar. It requires relatively little effort on the part of the grammar engineer to fully integrate XML structures in the grammar code.

At the point where a grammar gets constructed, we will introduce a notion of ‘meta-grammar’, where the objects treated are strings of analytic symbols, and not expressions of the language itself. This architecture is described in section 4. Sections 2 and 3 describe the IGT tool and the grammar in some detail, to provide a grounding of the basic components.

2. TypeCraft
TypeCraft (TC) consists of a relational database combined with a tabular text editor for the manual creation of text annotations wrapped into a wiki (TC-wiki). This design allows TC to serve in linguistic research that is dedicated to an Open Data approach. TC is loaded by directing a browser to www.typecraft.org. A customised wiki, the TC-wiki, serves as an access point to the TC-database. Standard wiki functionality is used to direct the user to the database via ‘New text’, ‘My texts’, and ‘Text- or Phrase search. ‘My texts’ displays the user's repository of annotated material, called ‘Texts’. The notion of Text in TC refers not only to coherent texts, but to any collection of individual phrases. TypeCraft offers some of the functionality associated with the Linguist's Shoebox/Toolbox (Toolbox) and SIL's new system FLEX. Different from Toolbox, TypeCraft is a relational database and therefore by nature has many advantages over file-based systems like Toolbox. Data management tasks as well as pattern matching are more easily performed when using a database search. Online databases like TypeCraft are multi-user systems, that is, many people can access the data at the same time independent of their physical location. Users administer their own data in their private space at TypeCraft, but they can also make use of other users’ shared data. Distributive applications such as TypeCraft facilitate the flow of research data within research collaborations. Data-sets can be easily exchanged between users and co-edited in user-groups. Interlinear Glosses can be loaded to the TC-wiki where they are displayed as part of a TC-wiki page. Under the printing of wiki pages they export automatically. One additional feature is that wiki-import
from the TC database gets automatically updated when the database changes.

TypeCraft is designed as an Interlinear Glosser with the goal to allow easy creation and management of data not only within a group of researchers but also for individual researchers for example preparing a publication. As a collaborative tool TypeCraft tries to utilise the effect of collaboration for the further standardisation of linguistic glosses. Glossing rules are conventional standards and one way to spread them is to make them easily accessible at the point where they are actively used. Different from Toolbox, TypeCraft insists that annotators use a pre-defined set of glosses. TypeCraft glossing standards are rooted in the Leipzig Glossing Rules, but have been extended to meet the needs of annotation in a more multi-lingual setting. With attention to conceptual transparency, part of speech (POS) glosses are kept separate from the other glosses. All glosses are grouped into annotation classes and mapped to the GOLD (General Ontology for Linguistic Description) ontology. GOLD has been created as a tool (Farrar and Langendoen, 2003) to facilitate a more standardised use of basic grammatical features. As an OWL ontology it presents grammatical features in terms of categories and their relations. By mapping the TC glosses to GOLD it is possible for the users of TC to gain direct access to the information provided by the GOLD system. This allows the users to relate gloss tags to grammatical concepts.

The system distinguishes between translational, functional and POS glosses. Properties can be assigned to linguistic phrases which can either be sentences or parts of sentences, and are represented by ‘Global tags’. Properties can also be assigned to words or morphemes. Each phrase receives a free translation. Global tags are called Construction parameters, and the user can select them from a separate window selecting from eight parameters: Construction kernel, Situation, Frame alternation, Secondary predicates, Discourse function, Modality, Force and Polarity. The annotation interface is an annotation table. Information is ordered horizontally (the phrase and the free translation) and vertically, so that words and morphs are aligned vertically with their Baseform and their POS, as well as their gloss. In addition they can be annotated for the attribute Head as an additional syntactic mark-up. TypeCraft encourages that free class morphemes are annotated for meaning while closed class items receive a gloss. Morphs may be accompanied by null to many glosses.

Each phrase in TypeCraft has a unique identifier, allowing exchange of the data on the net. TC data can be rendered as XML.

TypeCraft is connected to the web version of Ethnologue, a SIL International resource. Ethnologue gives information about the number of speakers and where the languages is spoken together with pointers to SIL publications about the language. In TypeCraft this resource can be directly accessed from the TC Editor.

TypeCraft uses Unicode, every script that the user can produce on the PC can be entered into the browser. As of recent, TypeCraft offers the transliteration of some Indic-to Latin script to make data that has been entered into the database using the original script more widely accessible.

Next to export to the main editors, TypeCraft supports XML export which allows the exchange of data with other applications.

Annotations made by linguists working on their own research reflect differences in interest and linguistic expertise. TypeCraft uses closed tag-sets, yet there is no restriction on how many items in a token sentence need to be annotated, or procedures that need to be followed to ensure that the annotations are linguistically sound. The approach thus presents a range of challenges having to do with inter- and intra-annotator consistency.

3. TypeGram

A well-known tool for multilingual grammar engineering based on the LKB platform is the ‘HPSG Grammar Matrix Matrix (‘Matrix’) (Bender et al. 2002, Bender et. al 2010). The architecture of the Matrix is essentially that of a ‘grammar intersection’ – a set of types and rules common to a set of grammars, where it is assumed that the more closely related that set of languages are, the larger will the intersection be. The opposite perspective is that of a ‘grammar union’ – the set of types and rules formed by the union of the set of types and rules of individual grammars, existing together as a functioning system. The latter is the strategy that we intend to pursue here.1

The system is called TypeGram, as yet unpublished (see http://typecraft.org/tc2wiki/TypeGram). Its feature geometry is comparable to that of Sag et al. 2003. Compared to the Matrix, it lacks the apparatus inducing MRS (Minimal Recursion Semantics; cf. Copestake et al. 2005) representations, which can however be added. Unlike standard HPSG grammars it uses attributes for grammatical functions (‘GF’).

To illustrate the scope of this grammar we give examples of how it analyzes phenomena as diverse as Bantu verb extensions and West-African serial verb constructions.

(1) below (from Kiswahili, based on Vitale,1981:165, quoted in Kroeger, 2004, p.196, ex. (11a)) is a passive form of a morphologically marked causative construction:

(1) mkeweke a-li-pi-ish-w-a uji
wife his S.agr-PAST-cook-CAUS-Pass-Ind gruel
‘His wife was made to cook gruel’

Its mapping between syntax and semantics can be represented as in (2), where the grammatical functions of a transitive construction are introduced by the attribute GF, viz. subject (SUBJ) and direct object (OBJ). Participants in the situation type expressed by the construction, introduced by the attribute ACTNTS (for ‘actants’), distinguished as ‘actant 1’ (ACT1) and ‘actant 2’ (ACT2), are interlinked to GFs through indices (introduced by INDEX).

1 The notion ‘meta-grammar’ might suggest itself for this kind of approach. However, apart from the use of that term which was just introduced, we would prefer using it in the way of Clement and Kinyon 2003, as a formalism distinct from that used in the ‘actual’ grammar(s) in question, whereas TypeGram uses exactly the same formalism as the grammars it subsumes.
As is observed, the *causer* participant lacks a syntactic realization, which is as expected if it is ‘underlyingly’ expressed as subject of the causative verb and then subjected to passivization. Also observed is that while the construction is syntactically a simple transitive structure, it semantically has a more complex structure with a ‘caused’ situation embedded as ACT2 relative to the ‘cause’ relation.

Taking the analysis a bit further, the notion ‘derivation’ suggests a definable ‘input’ and a definable ‘output’ relative to both the causativization and the passive formation. (3) displays how these notions can be further accommodated: a ‘derivation’ is portrayed as an upward ‘pumping’ process whereby the ‘input’ is represented by the attribute DTR (‘daughter’) inside of the ‘output’ structure. As (1) is a construction which has first undergone causativization, and then passivization, (through the agent becoming implicit, and the object being promoted to subject), an AVM displaying this derivational history will have a recursion of derivational ‘daughters’. This is displayed in (3), where the information given in (2) is supplemented by a display of the derivational process conceived as now exposed:

(3) Verb derived by morphological causative and subsequently passive:

\[
\begin{align*}
\text{HEAD } & \text{verb} \\
\text{GF } & \text{SUBJ } \text{INDEX } \text{1} \\
\text{OBJ } & \text{INDEX } \text{2} \\
\text{ACTNTS } & \text{PRED } \text{cause - rel} \\
\text{ACT1 } & \text{INDEX } \text{1} \\
\text{ACT2 } & \text{INDEX } \text{2} \\
\text{TENSE } & \text{past}
\end{align*}
\]

\[
\begin{align*}
\text{DTR } & \text{ACTNTS } \text{PRED } \text{cause - rel} \\
\text{ACT1 } & \text{INDEX } \text{1} \\
\text{ACT2 } & \text{INDEX } \text{2} \\
\text{TENSE } & \text{past}
\end{align*}
\]

(We here assume that the situational content of the passive construction is the same as that of its ‘active’ input, hence the values of ACTNTS and DTR\text{ACTNTS} can be set identical, but not those of DTR\text{DTRACTNTS} and DTR\text{DTRACTNTS}, since this step introduces causativization).

As part of the grammar mechanism by which the parser will produce (3) when applied to (1), the lexical specification type of the verb in this case is:

\[
\begin{align*}
\text{(4) } & \text{v-trPsCs-suPobCsu-obPob2Cob}
\end{align*}
\]

where \(v\) means ‘headed by verb’, \(trPsCs\) refers to a transitive argument structure derived through morphological causativization followed by passive formation. The sequence \(suPobCsu-obPob2Cob\) means that Subj is derived by passive from underlying Obj, in turn derived from underlying Subj, and that Obj is derived by passive from underlying Obj2, in turn derived from underlying Obj. The sequence \(suPobCsu\) thus corresponds to the AVM segment

\[
\begin{align*}
\text{GF.SBJ.INDX } & \text{2} \\
\text{DTR.GF.OBJ.INDX } & \text{2} \\
\text{DTR.DTR.GF.SBJ.INDX } & \text{2}
\end{align*}
\]

and could in principle be derived from it, as a compact declaration of this part of the overall AVM. A large set of possible combinations of ‘extensions’ of this kind (also including ‘applicative’, ‘reflexive’, and others) are defined in this code.

Turning to *Serial verb constructions* (SVCs), they have as basic distinctive properties that

(i) there is no head-complement relationship between the verbs involved;

(ii) there are pervasive uniformity requirements between the verbs, both in their shapes by themselves and regarding their arguments.

(5) is an SVC from the Kwa language Ga, which may conceivably be associated with the AVM (6) (which expresses no stand to what the exact syntactic relation between the two VPs is, and lacks a representation of the conventionalized meaning of this expression indicated in the free translation, which however is clearly grounded in the ‘literal’ parts of the semantics rendered here):

\[
\begin{align*}
\text{(5) } & \text{ Á-gbele gbɛ á-ha bo} \\
3.\text{PRF-open road 3.\text{PRF-give} 2S} \\
V & N V \text{ Pron}
\end{align*}
\]

‘You have been granted permission.’

\[
\begin{align*}
\text{(6) } & \text{ V1 PRED open - rel} \\
\text{ACTNTS } & \text{ACT1 } \text{perf} \\
\text{V2 PRED give - rel} \\
\text{ACTNTS } & \text{ACT1 } \text{perf}
\end{align*}
\]

\[
\begin{align*}
\text{DTR } & \text{DTRACTNTS} \\
\text{V1 } & \text{DTRACTNTS} \\
\text{V2 } & \text{DTRACTNTS}
\end{align*}
\]
This SVC has two verbs, both with an expressed object; their subjects are identical, and likewise their aspects. A grammar internal construction code is in this case svSuAspIDALL-v1tr-v2tr, again as an expression which can be derived from the AVM. Note that such code expressions are not defined one by one in toto, but built up from parts each representing a typical segment of an AVM, and thereby composed out of a finite set of ‘minimal labels’ (cf. Hellan and Dakubu 2009, 2010, Hellan 2008).

For enumerating the possible argument structures that can be realized (in active voice) in a language, around 200 such expressions seem necessary, based on the three languages so far investigated. Such an assembly we call a c-profile of the language in question (for ‘construction profile’, focusing on verbal argument structures, whereby the notion comes close to that of subcategorization frames).

For indexing of test-suites cross-linguistically, such codes are useful, and for comparison of valence frames across languages, related as well as non-related, the compactness of their format allows for reasonably large scale systematic comparison.

TypeCraft provides a ‘global’ annotation level where from 8 parameters as mentioned in section 2 of construction specification, the annotator can select one instantiation of each, such as ‘transitive’, ‘active+causative’, ‘svc’, etc. Each of these is linked to those grammar internal codes like (4) where the property in question is represented, such that in principle even the lexical subcategorization of the verb in an annotated string can be indicated through the IGT and automatically projected into the parsing-grammar’s grammar-internal code.

4. From TypeGram to individual meta-grammars

We now consider how single-language meta-grammars may be constructed from IGT annotation sets. A sentence like (5), instead of a constituent analysis like (7),

(7)  S | V P VP V NP V NP | | | Á-gbele gbɛ á-ha bo

will have the analysis tree (8),

(8)  S | V P VP V NP V NP | | | 3PRFopen road 3PRFgive 2S

where words and morphemes of the actual language are replaced by analytic symbols used in the IGT annotation of the string in question (what we may call a ‘meta-string’). This is accomplished by XML-conversion where an XML such as (9)

(9)  <word id="30409" text="etee" citation="etee"> <pos>V</pos> <morpheme id="46593" text="eI ” > <gloss>PERF</gloss> </morpheme> <morpheme id="46594" text="tee" meaning="go"/> </word>

will give rise to the Meta-Lexicon entry (10),

(10)  go_v := v-lxm & [ STEM "go" ].

and is filtered into the grammar resources as a meta-inflection rule of the type (11):

(11)  verb-Perf_irule := %suffix (* Perf) word & [ARGS < v-lxm > ].

The XML excerpt (12)

(12)  <word id="30491" text="lÉ›" citation="lÉ›"> <pos>DET</pos> <morpheme id="46710" text="lÉ› " > <gloss>DEF</gloss> </morpheme> </word>

similarly will induce the function-word entry (13):

(13)  DEF-DET := det-word & [ STEM "DEF" ].

Automatizing the construction of meta-strings can also be done on the basis of XMLs of annotated sentences, whereby the appropriate ‘gloss’ and ‘meaning’ parts of XMLs are extracted to form meta-strings for the sentences in question, such as ‘3PRFopen road 3PRFgive 2S’ for the example used above.

The mapping procedure in question will incrementally fill up the lexicon and inflection resources of the grammar.

A pilot performing these steps is described in Bruland (2011). The text annotations on which the construction of the three meta-grammars mentioned is based are partly done in TypeCraft, partly in standard publication format.
5. Conclusion

The novelty of the approach lies in the way IGT are made directly operative in the construction of deep parsers in the form of meta-grammars, which allow a hierarchical representation of grammatical functions and underlying argument structures, thus serving as an additional source of information for the linguist and the grammar writer alike.

The general strategy implementing this idea is of course not restricted to the particular tools described; thus, in principle not only data supplied originally to TypeCraft but also data annotations embedded in linguistic publications and elsewhere are relevant. (A project like ODIN (Lewis 2006, Lewis and Xia 2010), thus, can potentially supply data from such sources.) Given the growing IGT resources, the approach thus seems scalable.

References


Bruland, T. (2011). Creating TypeGram data from TypeCraft. Presentation at India 2011, NTNU.


