Proceedings of the CLTW 4 @ LREC2022 , pages 40–46
Marseille, 20-25 June 2022

Iterated Dependencies in a Breton treebank and implications for a Categorial Dependency Grammar

Annie Foret, Denis Béchet, Valérie Bellynck
IRISA& Univ. Rennes 1, Nantes University, Univ. Grenoble
Annie.Foret@irisa.fr, Denis.Bechet@univ-nantes.fr, Valerie.bellynck@imag.fr

Abstract

Categorial Dependency Grammars (CDG) are computational grammars for natural language processing, defining dependency structures. They can be viewed as a formal system, where types are attached to words, combining the classical categorial grammars' elimination rules with valency pairing rules able to define discontinuous (non-projective) dependencies. Algorithms have been proposed to infer grammars in this class from treebanks, with respect to Mel'čuk principles. We consider this approach with experiments on Breton. We focus in particular on "repeatable dependencies" (iterated) and their patterns. A dependency d is iterated in a dependency structure if some word in this structure governs several other words through dependency d. We illustrate this approach with data in the universal dependencies format and dependency patterns written in Grew (a graph rewriting tool dedicated to applications in natural Language Processing).

Keywords: Formal Grammar, Categorial Grammar, Treebank, Universal Dependencies, Breton, Repeatable Dependencies, Grammatical Inference, Graph Rewriting

1. Introduction

This paper discusses how a formal grammar in the class of categorial dependency grammars can be applied to under-resourced languages.

Previous works have proposed the categorial dependency framework for natural language modelling and processing, with nice formal and pratical properties: polynomial parsing complexity and algorithms to infer such grammars from dependency treebanks.

We conducted experiments in that direction on Breton. Using Grew with both CDG grammars and a treebank provides quickly specific views of the linguistic data: in our case views related to the interpretations of the Mel'čuk repeatable dependency principle. This helps to validate this repeatable principle for a language such as Breton and annotation guidelines.

Several dependency treebanks are developed for Celtic languages (Lynn and Foster, 2016; Batchelor, 2019; Heinecke and Tyers, 2019). In this work we consider the UD_Breton-KEB corpus [1] (Tyers and Ravishankar, 2018). We wrote programs with reproducible experiments on Breton annotated sentences following the Universal Dependencies scheme [2].

We focus in particular on iterated dependencies. A dependency d is iterated in a dependency structure if some word in this structure governs several other words through dependency d. The iterated dependencies are due to the basic principles of dependency syntax, on optional repeatable dependencies (Mel'čuk, 1988): All modifiers of a noun n share n as their governor and, similarly, all modifiers of a verb v share v as their governor. At the same time, the iterated dependencies have been a challenge for grammatical inference (Béchet and Foret, 2021): the class of k-valued CDG (at most k types per word) is not learnable (in the sense of Gold's (Gold, 1967) "identification in the limit"), while the class of k-valued "iteration-free" CDG is learnable.

The paper is organized as follows. Section 2 introduces Categorial Dependency Grammars (CDG). Section 3 provides an inference algorithm for CDG when we interprete the notion of iterated dependencies as consecutive outgoing edges separately on the left and on the right of a governor. We also discuss different possible interpretations of the notion of iterated dependencies, handled in extended CDG. Section 4 reports on experiments on a Breton corpus. Section 5 concludes. We provide code on CDG and UD available at the *cdg-ud* page[3].

2. Categorial Dependency Grammars

A CDG (Dekhtyar et al., 2015) is a formal grammar that defines a language of surface dependency structures. A surface dependency structure is a list of words linked together by dependencies. Each dependency has a name, a starting point called the governor and an ending point called the subordinate.

Figure 1 shows a surface dependency structure for the string "*This deal brought more problems than profits.*". The structure contains eight words (or punctuation symbols) and seven dependencies. The arrow between *brought* and *problems* defines a dependency of name $a-obj$ where *brought* is the governor and *problems* is the subordinate (this dependency indicates that *problems* is the object of *brought*). The root of the structure is the word *brought* (this word isn't the subordinate of any dependency). The CDG dependency

[1] V1.0 available at https://universaldependencies.org/treebanks/br_keb/index.html

[2] https://universaldependencies.org/

[3] https://gitlab.inria.fr/foret/cdg-ud

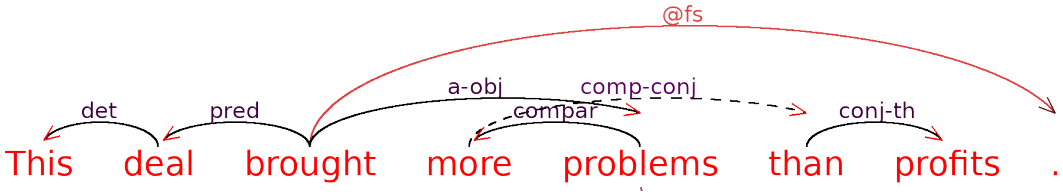

Figure 1: A Dependency Structure.

$\mathbf{L^l}$	$[C]^P[C\backslash\beta]^Q \vdash [\beta]^{PQ}$	($\backslash$ elimination)
$\mathbf{I^l}$	$[C]^P[C^*\backslash\beta]^Q \vdash [C^*\backslash\beta]^{PQ}$	($\backslash$ repetition)
Ω^l	$[C^*\backslash\beta]^P \vdash [\beta]^P$	($\backslash$ option)
$\mathbf{D^l}$	$\alpha^{P_1(\swarrow V)P(\nwarrow V)P_2} \vdash \alpha^{P_1PP_2}$	P *without* $\swarrow V \nwarrow V$

Table 1: The CDG Type Calculus (Left Rules)

structures are not necessarily dependency trees because certain dependencies called discontinuous dependencies are usually introduced together with an auxiliary dependency called an anchor [4]. In the example, there is a discontinuous dependency $comp{-}conj$ but its anchor dependency is not shown here.

A CDG is defined mainly by a lexicon that associates types to words and punctuation symbols. The following lexicon shows a lexicon for the previous dependency structure (the anchor sub-types for the discontinuous dependency $comp{-}conj$ are presented as $\#\searrow comp{-}conj$ in the types of $problems$ and $than$):

$this$	$\mapsto$	$[det]$
$deal$	$\mapsto$	$[det\backslash pred]$
$brought$	$\mapsto$	$[pred\backslash S/@fs/a{-}obj]$
$problems$	$\mapsto$	$[compar\backslash a{-}obj/\#\searrow comp{-}conj]$
$profits$	$\mapsto$	$[conj{-}th]$
$more$	$\mapsto$	$[compar]^{\nearrow comp\text{-}conj}$
$than$	$\mapsto$	$[\#\searrow comp{-}conj/conj{-}th]^{\searrow comp\text{-}conj}$
.	$\mapsto$	$[@fs]$

in this CDG, S is for sentences, $@fs$ is for the full stop.

CDG languages are defined by a dependency types calculus showed on Table 1 which constructs Dependency Structures. Figure 2 shows a proof tree for a simple sentence and typing (en, br). Figure 3 shows a subproof where labels are abreviated (en, br).

In comparison with CCG or Lambek grammars, CDG are written using flat types without type-raising mechanism. From a practical point of view, CDGLab (Béchet et al., 2014) implements a parser for CDG. The lab can also help to define a CDG together with corpora for a specific language. For instance, a large scale grammar and a corpus for French have been developed with this tool (Béchet and Lacroix, 2015).

[4]Formally a token could have several heads, but practically, token have one head or one main head and auxiliary heads (for anchors)

3. CDG Learning and Subclasses

The notion of K-star has been introduced to define learnable subclasses of CDG grammars allowing iterated dependencies. This constraint differs from the k-valued bound and does not impose a bound on the number of types associated to a word. A K-star constraint (for a number K) reflects an indiscernability principle between K repetitions of a same dependency d and its iterated form d^*. Different K-star criterions have been proposed, that enable grammatical inference in the presence of iterated dependencies; the first one "K-star revealing" is a complex non-constructive criterion, the two later proposals "Simple K-star" (Béchet and Foret, 2016b) and "Global Simple K-star" (Béchet and Foret, 2021) (the global variant does not impose the repetitions to be consecutive in a type) are both syntactic and easy to check on a given grammar. The inference of CDG with these properties is possible from a corpus using the algorithm in Figure 5 (where the set of atomic types depends from the corpus labels). The algorithm can be used to complete an existing CDG as well.

3.1. An Inference Algorithm from a Treebank

The algorithm we proposed in Béchet et al. (2010) first computes a "pre-type" for each word from a dependency structure, called a vicinity, following the outgoing dependencies in sentence order, but without marked iteration. This type is then generalized before expanding the grammar. This kind of algorithm is termed TGE-like for "Type-Generalize-Expand" (Béchet and Foret, 2021). The algorithm is shown in Figure 5.

3.2. Vicinity of a word on a dependency structure

Vicinity. The TGE method involves a first set of types without iteration, called vicinity, that can be directly obtained from a dependency structure. The *vicinity* $V(w, D)$ of a word w in a (labelled) dependency struc-

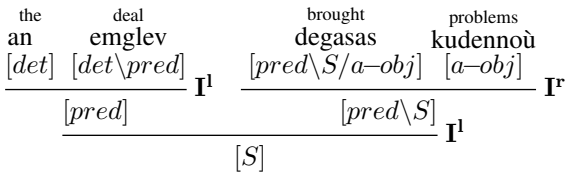

Figure 2: A Proof Tree.

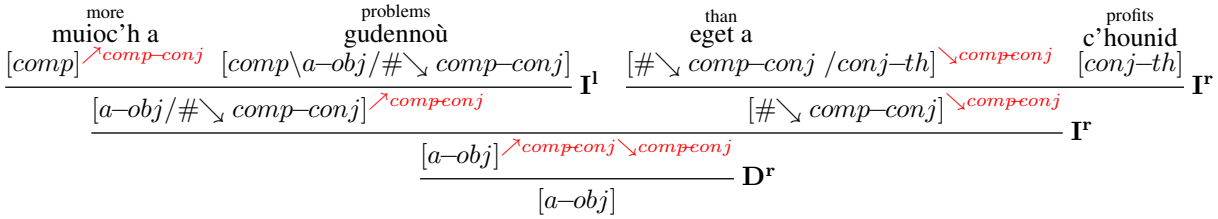

Figure 3: A Simplified Subproof Tree.

ture D, is the type

$$V(w, D) = [l_1 \backslash \ldots \backslash l_k \backslash h / r_m / \ldots / r_1]^P,$$

such that D has:
- the incoming projective dependency or anchor h (or the axiom S for sentences),
- the left projective dependencies or anchors $l_k, \ldots, l_1$ (in this order),
- the right projective dependencies or anchors $r_1, \ldots, r_m$ (in this order),
- the discontinuous dependencies $d_1, \ldots, d_n$ with their respective polarities P to handle their start, their end and their orientation.[5]

3.3. $\mathrm{TGE}^{(\mathbf{K})}$ Algorithm and Example.

Our learning algorithm is provided on Figure 5. From the dependency structure D (fragment) of Figure 4 for a sentence in French[6], we get these two vicinity types:
$V(partition, D) = [det\backslash a-obj/modif/attr/attr/modif]$,
$V(de, D) = [attr/prepos-g]$
If the "2-star repetition principle" applies to the $attr$ dependency, in the sense that if $attr$ occurs consecutively two times then it can occur consecutively any number of times, the previous vicinity of *partition* would be generalized by this CDG type:
$partition \mapsto [det\backslash a-obj/modif/attr^*/modif]$

3.4. Repetition Patterns

Grammar classes and TGE algorithm. The same TGE^K algorithm can be run to learn the class of simple K-star grammars. It can be adjusted to learn global K-star grammars by adding a final step replacing each type t of the output of TGE^K by its *global simple K-star generalization* $gs^{(K)}(t)$ obtained as follows (Béchet and Foret, 2021):

- for each d on the left, where $d \backslash$ occurs at least K times or if $d^* \backslash$ is present, then replace each $d \backslash$ with its starred version $d^* \backslash$
- for each d on the right, proceed similarly.

Variants and extended types. More flexible interpretations than the strict reading of repeatable optional dependencies as "consecutive repetitions" have been proposed.
- "Dispersed iteration" (Pogodalla and Prost, 2011), $\{d_1^*, \ldots, d_p^*\}$ represents the case where the subordinates through a repeatable dependency may occur in any position on the left (respectively, on the right) of the governor.
- "Choice iteration" (Pogodalla and Prost, 2011), $(d_1|\ldots|d_k)^*$ represents the case where the subordinates through one of several repeatable dependencies may occur in one and the same argument position. Using a similar approach in the dispersed case, an algorithm TGE^K_{disp} has been shown to learn K-star dispersed revealing grammars. A similar learning algorithm TGE^K_{ch} is provided for "choice iteration".
- CDGs with "sequence iteration" have later been proposed in Béchet and Foret (2016a) as a generalization of d^*: repeating $/\ d_2\ /\ d_1\ /\ d_2\ /\ d_1$, etc. as $/\ (d_1 \bullet d_2)^*$. An extended CDG-calculus and a TGE-like algorithm for sequences of length 2 is provided in Béchet and Foret (2016a)[7]. This extension seems relevant for treebanks.

4. Experiments

Experiments are reported in Béchet and Foret (2016a) to process vicinities from a French treebank and view patterns in a concept analysis tool[8]. In this paper, we

[5] P is a sequence of elements of the form: $\nwarrow d$ (start left) $\searrow d$ (end right), $\swarrow d$ (end left), $\nearrow d$ (start right).

[6] "On y trouve aussi une partition récente à récupérer de l'ONPL signée par lui." , meaning "There is also a recent score to recover from the ONPL signed by him."

[7] Sequence iteration does not introduce new string languages

[8] Camelis available at `http://www.irisa.fr/LIS/ferre/camelis`

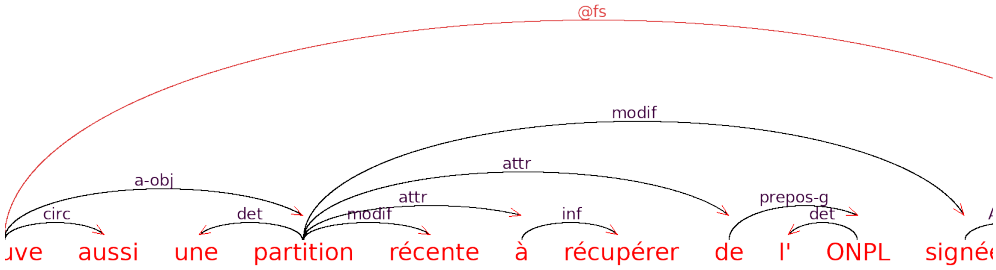

Figure 4: Part of a Dependency Structure.

Algorithm $\mathbf{TGE}^{(K)}$ (type-generalize-expand):
Input: σ, a training sequence of length N.
Output: CDG $\mathbf{TGE}^{(K)}(\sigma)$.

let $G_H = (W_H, \mathbf{C}_H, S, \lambda_H)$
  where $W_H := \emptyset$; $\mathbf{C}_H := \{S\}$; $\lambda_H := \emptyset$;
(loop) **for** $i = 1$ **to** N //loop on σ
  let D such that $\sigma[i] = \sigma[i-1] \cdot D$;
    // the i-th dependency structure of σ
  let $(X, E) = D$;
  (loop) **for every** $w \in X$
    // the order of this loop is not important
    $W_H := W_H \cup \{w\}$;
    let $t_w = V(w, D)$
      // the vicinity of w in D
    (loop) **while** $t_w = [\alpha\backslash l\backslash \mathbf{d}\backslash \cdots \backslash \mathbf{d}\backslash r\backslash \beta]^P$
      with at least K consecutive occurrences of d,
      $l \neq d$ (or $\alpha\backslash l\backslash$ not present)
      and $r \neq d$ (or $r\backslash$ not present)
        $t_w := [\alpha\backslash l\backslash \mathbf{d}^*\backslash r\backslash \beta]^P$
    (loop) **while** $t_w = [\alpha/l/\mathbf{d}/\cdots/\mathbf{d}/r/\beta]^P$
      with at least K consecutive occurrences of d,
      $l \neq d$ (or $/l$ not present)
      and $r \neq d$ (or $/r/\beta$ not present)
        $t_w := [\alpha/l/\mathbf{d}^*/r/\beta]^P$
    $\lambda_H(w) := \lambda_H(w) \cup \{t_w\}$;
      // lexicon expansion
    end end
return G_H

Figure 5: Algorithm $TGE^{(K)}$

use Grew[9] (Guillaume, 2021) to search[10] for patterns corresponding to the CDG vicinities and their possible generalizations; in other words, these express patterns on the successive dependencies outgoing from a given word.

4.1. Edge patterns on a corpus

We wrote patterns, in the Grew syntax, to select graphs (sentences) containing dependency name repetitions, depending on these parameters: a number K of repetitions, a repetition mode (anywhere/flex or consecutive/cons), a side (left, right, or both). We give some of them below, then an occurrence table on Breton data (we provide more patterns at the *cdg-ud* page[3]):

- 2 repetitions, anywhere, left or right (2rep flex l/r)

```
pattern { e: GOV -> DEP1;
f: GOV -> DEP2;
e.label = f.label ;
DEP1 << DEP2 }
```

- 3 repetitions, anywhere, left or right (3rep flex l/r)

```
pattern { e: GOV -> DEP1;
f: GOV -> DEP2;  g : GOV -> DEP3;
e.label = f.label ;
e.label = g.label ;
DEP1 << DEP2 ; DEP2 << DEP3}
```

- 2 repetitions, consecutive, right (2rep cons r)

```
pattern { e: GOV -> DEP1;
f: GOV -> DEP2;
e.label = f.label ;
DEP1 << DEP2 ;  GOV << DEP1 }
without { g: GOV -> DEP12 ;
DEP1 << DEP12 ; DEP12 << DEP2 }
```

- 2 repetitions, consecutive, left (2rep cons l)

```
pattern { e: GOV -> DEP1;
f: GOV -> DEP2;
e.label = f.label ;
DEP1 << DEP2 ;  DEP2 << GOV }
without { g: GOV -> DEP12 ;
DEP1 << DEP12 ; DEP12 << DEP2 }
```

[9] https://grew.fr

[10] We wrote other patterns related to CDG, we also wrote Grew rules that extend such patterns, to transform the corpus with http://transform.grew.fr/, with several outcomes compatible with the UD format: for marking repeatable dependencies on relevant edges, for adding the projective vicinities as node features, for the inference algorithm ; these files can be directly tested on the Grew site and can be provided on demand, see also the *cdg-ud* site[3].

UD e.label	2rep flex l/r 2 repetitions anywhere left or right	3rep flex l/r 3 repetitions anywhere left or right	2rep cons r 2 consecutive repetitions right	2rep cons l 2 consecutive repetitions left	3resp cons r 3 consecutive repetitions right
aux	268	35	107	105	1
advmod	133	22	5	8	
obl	119	16	53	2	7
punct	83	3	3	1	
conj	75	59	42		18
dep	16			11	
nmod:gen	16	1	14		1
det	13			13	
nmod	12	1	8		1
amod	10	1	7		1
flat:name	4		4		
case	3			2	
parataxis	3		2	1	
nsubj	2				
fixed	2		2		
acl	2		2		
advcl	2				
list	1		1		
SUD e.label					
mod	214	27	59	8	5
udep	88	8	64	2	8
punct	78	2	2	1	
unk	16			12	
mod@gen	16	1	14		1
det	13			13	
parataxis	3		2	1	
subj	2				
list	1		1		

Table 2: Occurrences of edge labels w.r.t. repetition patterns on Breton data, in UD and SUD formats

These results in table 2 apply to two versions of the Breton corpus, UD format (de Marneffe et al., 2021) and SUD format (Gerdes et al., 2018)[11], where we ask for e.label in the above patterns.
We can also check the amount of discontinuous (non-projective) dependencies in these formats, with:

```
global { is_not_projective }
pattern { e:GOV -> DEP }
without {f:X -> GOV}
```

We get few (19) in the UD format, and much more (296) in the SUD format. The SUD format seems relevant for further developments. This raises this question: what is the best amount of non-projectivity needed in Breton. CDG is a good formalism for discontinuity, this is not developed here.

4.2. Selected sentences and dependencies

We select here two sentences, to illustrate different repetition status.

[11]SUD stands for "Surface Syntactic Universal Dependencies", the SUD scheme is a recent alternative to the UD format, with possible automatic conversion main differences are that SUD favors functional heads, and has an more economical set of labels; a comparison summary can be found at `https://surfacesyntacticud.github.io/conversions/`

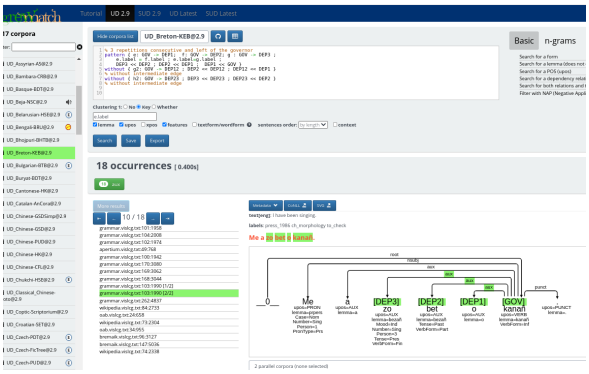

Figure 6: "Me a zo bet o kanañ" in UD format

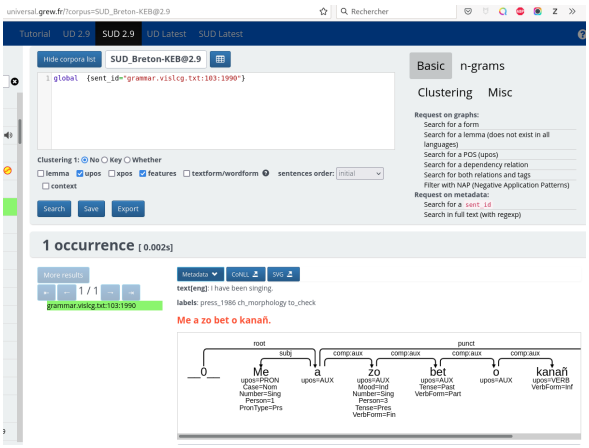

Figure 7: "Me a zo bet o kanañ" in SUD format

In the first sentence "Me a zo bet o kanañ." (meaning "I have been singing.", with sent_id="grammar.vislcg.txt:103:1990"), in the original UD format (Figure 6), 3 consecutive edges on the left of the same governor have the same label (*aux*); this does not happen in the SUD format (Figure 7). This dependency (*aux*) may preferably be kept non-repetitive on the left (in the consecutive or in a flexible reading).
In another sentence "Gant ur c'hresk a 35% eus ar veajourien dindan pemp bloaz, emañ Breizh e penn rannvroioù Frañs evit an TER" (with sent_id="oab.vislcg.txt:163:4313" , meaning "With a 35% increase in TER use in five years, Brittany ranks first among the regions of France."), 3 consecutive edges on the right of a same governor have the same label: *nmod* in UD as in Figure 8, *udep* in SUD as in Figure 9. This dependency (*nmod*) is preferably considered as repetitive on the right (in the consecutive reading).

5. Conclusion and further work

In this study we tried to answer the question: how to identify iterated dependencies on a Breton corpus and translate them into iterated types, to design a Categorial Dependency Grammar (CDG).

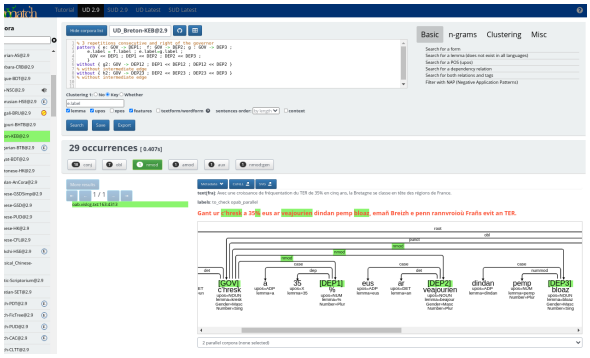

Figure 8: “Gant ur c’hresk a 35% eus ... ” in UD

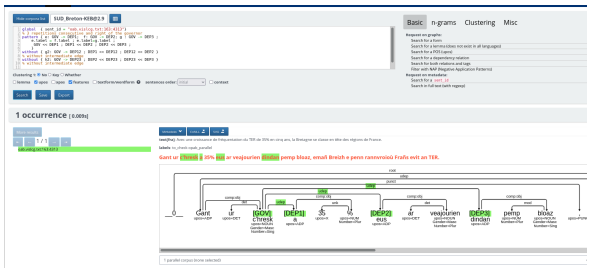

Figure 9: “Gant ur c’hresk a 35% eus ... ” in SUD

One of the issues in grammatical design or in corpus annotation is to determine the good level of generalization and of automation. Through explorations and experiments, we also aim to provide some answers and recommendations both formally and practically.

Here are some other questions we wish to address for Breton:

- In the case of discontinuous dependencies in the corpus, how to treat them (for CDGs, how can we manage the introduction of polarized valencies and anchors); in case of absence of discontinuous dependencies, is it a weakness of the annotated corpus?
- How to use the information from a site such as Arbres (Jouitteau, 2009 2022), an online site describing the grammar of Breton?
- In documents, what are the levels and sources of ambiguity and how to deal with them?
- What processing chain should we develop to go from a text to a targeted semantics (case of the French-Breton language pair)?

6. Acknowledgements

This study has benefited from funds from CNRS and DGLFLF (LangNum-br-fr project) enabling a user study and several student internships (K. Kechis, P. Morvan, P. Martinet). We thank the reviewers for their helpful comments and M. Jouitteau, E. Hupel for useful discussions on Breton.

7. Bibliographical References

Batchelor, C. (2019). Universal dependencies for Scottish Gaelic: syntax. In *Proceedings of the Celtic Language Technology Workshop*, pages 7–15, Dublin, Ireland, August.

Béchet, D. and Foret, A. (2016a). Categorial dependency grammars with iterated sequences. In *Logical Aspects of Computational Linguistics, Nancy, France, December 5-7, 2016*, pages 34–51.

Béchet, D. and Foret, A. (2016b). Simple k-star categorial dependency grammars and their inference. In *Proceedings of the 13th International Conference on Grammatical Inference, ICGI 2016*, pages 3–14.

Béchet, D. and Foret, A. (2021). Incremental learning of iterated dependencies. *Machine Learning*, March.

Béchet, D. and Lacroix, O. (2015). CDGFr, un corpus en dépendances non-projectives pour le français. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles, June 2015, Caen, France*, pages 522–528. Association pour le Traitement Automatique des Langues. Short paper in French.

Béchet, D., Dikovsky, A., and Foret, A. (2010). Two models of learning iterated dependencies. In Markus Egg, et al., editors, *Proceedings of the 15th International Conference on Formal Grammar (FG10), Copenhagen, Denmark, August 7-8, 2010*, pages 1–16.

Béchet, D., Dikovsky, A., and Lacroix, O. (2014). “CDG Lab”: an integrated environment for categorial dependency grammar and dependency treebank development. In Kim Gerdes, et al., editors, *Computational Dependency Theory*, volume 258 of *Frontiers in Artificial Intelligence and Applications*, pages 153–169. IOS Press.

de Marneffe, M.-C., Manning, C. D., Nivre, J., and Zeman, D. (2021). Universal Dependencies. *Computational Linguistics*, 47(2):255–308, 07.

Dekhtyar, M. I., Dikovsky, A., and Karlov, B. (2015). Categorial dependency grammars. *Theor. Comput. Sci.*, 579:33–63.

Gerdes, K., Guillaume, B., Kahane, S., and Perrier, G. (2018). SUD or surface-syntactic universal dependencies: An annotation scheme near-isomorphic to UD. In Marie-Catherine de Marneffe, et al., editors, *Proceedings of the Second Workshop on Universal Dependencies, UDW@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 66–74. Association for Computational Linguistics.

Gold, E. M. (1967). Language identification in the limit. *Information and control*, 10:447–474.

Guillaume, B. (2021). Graph Matching and Graph Rewriting: GREW tools for corpus exploration, maintenance and conversion. In *EACL 2021 - 16th conference of the European Chapter of the Association for Computational Linguistics*, Kiev/Online.

Heinecke, J. and Tyers, F. M. (2019). Development of a Universal Dependencies treebank for Welsh. In *Proceedings of the Celtic Language Technology Workshop*, pages 21–31, Dublin, Ireland, August. European Association for Machine Translation.

Jouitteau, M. (2009-2022). ARBRES, wikigrammaire des dialectes du breton et centre de ressources pour son étude linguistique formelle, IKER, CNRS. http://arbres.iker.cnrs.fr. Licence Creative Commons BY-NC-SA.

Lynn, T. and Foster, J. (2016). Universal dependencies for irish. In *Proceedings of the Celtic Language Technology Workshop*, Paris, France.

Mel'čuk, I. (1988). *Dependency Syntax*. SUNY Press, Albany, NY.

Sylvain Pogodalla et al., editors. (2011). *Logical Aspects of Computational Linguistics, 6th International Conference, LACL 2011, Montpellier, France, June 29 – July 1, 2011. Proceedings*, volume 6736 of *Lecture Notes in Computer Science (LNCS)*. Springer.

Tyers, F. M. and Ravishankar, V. (2018). A prototype dependency treebank for breton. In *Actes de la 25e conférence sur le Traitement Automatique des Langues Naturelles (TALN)*.