IJCoL

Italian Journal
of Computational Linguistics

Rivista Italiana
di Linguistica Computazionale

Volume 2, Number 2
december 2016

Special Issue:
Digital Humanities and Computational Linguistics

Guest Editors:
John Nerbonne, Sara Tonelli

aA
ccademia
university
press

Registrazione presso il Tribunale di Trento n. 14/16 del 6 luglio 2016

Rivista Semestrale dell'Associazione Italiana di Linguistica Computazionale (AILC)

direttore responsabile
Michele Arnese

isbn 978-88-99982-26-3

Accademia University Press
via Carlo Alberto 55
I-10123 Torino
info@aAccademia.it
www.aAccademia.it/IJCoL_2_2

Accademia University Press è un marchio registrato di proprietà
di LEXIS Compagnia Editoriale in Torino srl

IJCoL

Volume 2, Number 2
december 2016

Special Issue:
Digital Humanities and Computational Linguistics

Guest Editors:
John Nerbonne, Sara Tonelli

CONTENTS

PaCQL: A new type of treebank search for the digital humanities

Anton Karl Ingason[*]
University of Iceland

This article describes PaCQL (Parsed Corpus Query Language), a novel query language for carrying out research on parsed historical corpora, an important task for the digital humanities. PaCQL implements and enhances many of the most important features of earlier software that is designed for computational research in historical syntax and combines such functionality with a search engine which employs a fast in-memory index that cuts down waiting time in many realistic research scenarios. A web interface is provided with an automatically created summary of the main quantitative findings. The primary goal of this project is to contribute to the development of software tools which are designed from the ground up specifically with the needs of the digital humanities in mind.

1. Introduction

The ability of Computational Linguistics to analyze and process the syntactic structure of natural language was transformed with the emergence of parsed corpora such as the Penn Treebank (Marcus, Marcinkiewicz, and Santorini 1993). Subsequently, the digital humanities have gradually caught on to the possibilities offered by such resources and today, theoretical and historical syntax increasingly makes use of treebanks. One major milestone in bringing the power of treebanks to the humanities is the development of the Penn Parsed Corpora of Historical English (Kroch and Taylor 2000b; Kroch, Santorini, and Delfs 2004) and their annotation scheme which has been adapted to historical corpora for other languages, including French (Martineau 2011), Portuguese (Britto and Galves 2009), and Icelandic (Wallenberg et al. 2011).

Although treebanks are to an extent generic enough to be studied using a variety of software tools, the needs of historical syntacticians are not necessarily the same as the needs of computational linguists and therefore a demand has arisen for treebank software which is appropriate for use in the digital humanities. For example, ease of use is important and this has motivated the development of example-based search as in the GrETEL system (Augustinus, Vandeghinste, and Van Eynde 2012) and the Linguist's Search Engine (LSE) (Resnik and Elkiss 2005). Output requirements may also be different in the humanities. While a computational linguist may write a program which trains a statistical parser on a treebank or use Tgrep (Rohde 2004) to extract certain patterns from a treebank, a historical syntactician generally needs to extract a dependent syntactic variable from the corpus along with several independent variables, preferably without a heavy dependence on programming skills. Furthermore,

[*] Department of Icelandic and Comparative Cultural Studies - Árnagarði v. Suðurgötu, IS-101 Reykjavík, Iceland. E-mail: antoni@hi.is

the historical syntactician will need her results in a format which facilitates subsequent statistical analysis and visualization of quantitative patterns.

For example, a classic topic in the study of historical syntax of languages like English and Icelandic is the evolution from object-verb (OV) word order to the verb-object (VO) order (Kemenade 1987; Lightfoot 1991; Rögnvaldsson 1994/1995, 1996; Lightfoot 1999; Kroch and Taylor 2000a; Hróarsdóttir 2000). A sentence with an auxilary verb like *will* followed by a main verb (e.g., *eat*) and a direct object (e.g., *the bread*) in either of the two orders can be taken to be a diagnostic of the relevant contrast. This is demonstrated in the examples below which are given in Modern English orthography even though the OV pattern is in fact associated with older periods in history.

(1) a. She will [the bread eat]. (OV diagnostic)

b. She will [eat the bread]. (VO diagnostic)

The researcher typically wants to know the rate at which the OV order is used in each century of the written record and it is important to be able to break down the results by the type of direct object involved (e.g., full noun phrase, pronoun, quantifier), as well as stylistic factors like genre (e.g., narratives vs. religious text). Results of this type are shown for the proportion of OV word order in Icelandic in Figure 1. The graphs show that usage of the OV order declines over time in all contexts. While it is logically possible to use a program like Tgrep in order to extract the required data, this would be a tedious task and it would probably involve some time consuming development of custom scripts to manage the independent variables. Therefore, many studies in historical syntax currently make use of CorpusSearch (Randall 2005) which allows for a tabulated output of the required type. However, while extremely useful, CorpusSearch leaves some room for improvement. Its query syntax is rather verbose, complex studies often still require a bit of shell script manipulation of the results, the program has a rather limited ability to summarize its results and it can take a while to run a query because no indexing of the treebank is used. This paper describes PaCQL, a new query language which addresses these issues.

PaCQL, the Parsed Corpus Query Language (pronounced *pack-well*), uses a compact expressive syntax and it can be used via an online web interface with syntax highlighting and various features which automate common tasks in the everyday work of a historical syntactician. A preview version of PaCQL search for the Icelandic treebank, IcePaHC, is currently available at www.treebankstudio.org. Note that the software is under active development and more features will be added in future versions. An example PaCQL query is given below.

(2)
```
IP-(MAT|SUB) idoms MDPI

ov:1
MDPI sprec NP-OB[12]
NP-OB[12] sprec VB

ov:0
MDPI sprec VB
VB sprec NP-OB[12]

meta:
text century
```

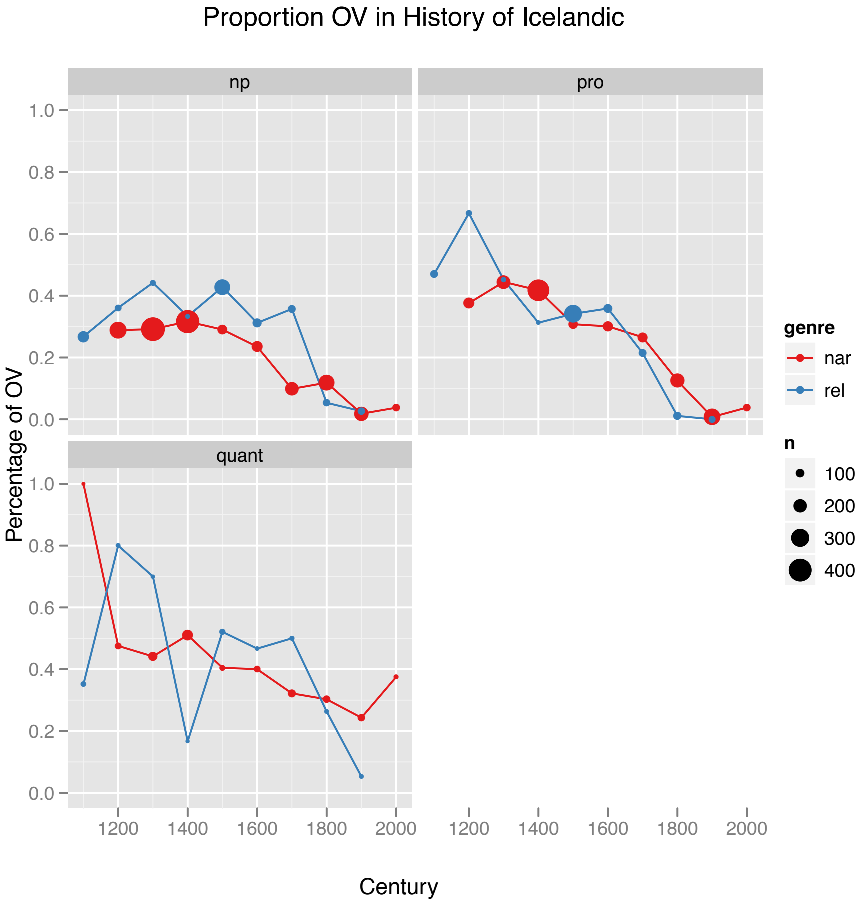


Figure 1
Rate of object-verb word order in the history of Icelandic.

This query involves four sections separated by empty lines: A main section (unlabeled at the top), two coding sections for the dependent variable (`ov:1`) and (`ov:0`), and a meta section (`meta:`) which indicates that the century of writing should be included with each result. The main section and the coding sections specify syntactic relationships; `idoms` means 'immediately dominates' and `sprec` stands for 'sisterwise precedence'. Elements like `IP-(MAT|SUB)` and `MDPI` are Regular Expressions which match the labels of nodes in a tree structure. Here, `IP-(MAT|SUB)` matches the labels `IP-MAT` (matrix clause) and `IP-SUB` (finite subordinate clause) in a treebank like IcePaHC whereas `MDPI` matches the part of speech tag for modal verbs (`MD`) in the present (`P`) indicative (`I`). `VB` is an infinitival main verb. If the exact same label is specified twice in a query, PaCQL assumes that it refers to the same node in both cases. The query language is described in more detail below. The following is an example result from IcePaHC of the OV type.

(3)

```
(IP-SUB
  (NEG eigi)
  (MDPI mun)
  (NP-SBJ (NPR-N guð))
  (NP-OB1
    (PRO-A mig))
  (NP-SPR (N-A helvítismann))
  (VB dæma))
```

The file format of IcePaHC (as well as other corpora that are annotated in the tradition of the Penn historical treebanks; see references above), is the same type of labeled bracketing as used for the Penn treebank but the structure is more flat.[1] The main reason for the relatively flat structure is that it often makes it more convenient to work with highly variable word order during periods of ongoing syntactic change. The query language does not depend on this property of the annotation scheme. The example result above is only a partial tree because it only shows the subordinate clause that matched the query. However, it is clear from the example that various clausal elements are attached directly to the `IP-SUB` node which defines a finite subordinate clause.

The remainder of the paper is organized as follows. Section 2 describes the PaCQL query language, Section 3 describes the output of the system, and Section 4 discusses some current shortcomings of the system and plans for future work. Section 5 concludes.

2. PaCQL

2.1 Syntactic relationships

The syntactic relationships which PaCQL can query for are inspired by CorpusSearch and most of them are available in both systems. The basic syntactic relationships are shown in (4) and every query must contain at least one of those.

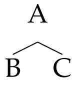
(4) `idoms`: immediately dominates
`idomsonly`: immediately dominates x and nothing else
`idomsfirst`: immediately dominates the leftmost child x
`idomslast`: immediately dominates the rightmost child x
`doms`: dominates at an arbitrary depth
`sprec`: sisterwise precedence
`precedes`: precedence regardless of embedding
`hassister`: sisterhood
`sameindex`: A has the same index as B

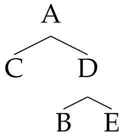
Although these relationships are fairly self-explanatory, we will offer a brief overview of the types of structures that each of them matches. Consider first the `idoms` relationship which stands for for 'immediately dominates'. In the following example, the relationship only yields a match if B is a direct child node of A.

[1] Certain aspects of the annotation will be explained briefly as they are encountered below. To make full use of PaCQL, the user must learn about the annotation guidelines of a given treebank in more detail than we can present here in a paper which focuses on the query language.

(5) **Relationship:** **Matches:** **Does not match:**

`A idoms B`

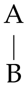


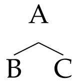


The relationship `idomsonly` also involves immediate dominance but it is restricted to unary branching structures. The following only matches structures where B is the only child node of A in the tree.

(6) **Relationship:** **Matches:** **Does not match:**

`A idomsonly B`

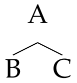


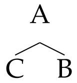


A match is returned from the following use of `idomsfirst` only if the node with the label B is the leftmost child of the node with the label A.

(7) **Relationship:** **Matches:** **Does not match:**

`A idomsfirst B`

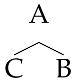


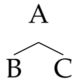


In a parallel manner, `idomslast` returns a match if the node matched as B is the rightmost child of node A.

(8) **Relationship:** **Matches:** **Does not match:**

`A idomslast B`

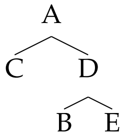


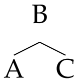


The relationship `doms` stands for 'dominates' and it matches structures where the second node is contained within the first node at an arbitrary depth.

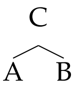
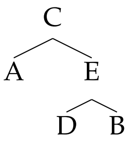
(9) **Relationship:** **Matches:** **Does not match:**

`A doms B`

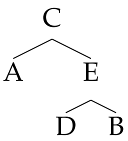
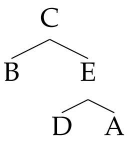




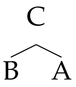
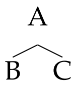
One can use `sprec` to match cases of sisterwise precedence. Here, `A sprec B` yields a result if A and B are sisters and B is also linearly to the right of A.

(10) **Relationship:**	**Matches:**	**Does not match:**
`A sprec B`		

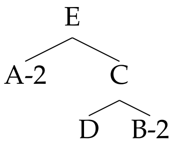
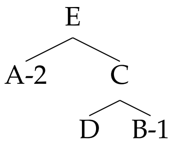
In contrast to `sprec`, we can use `precedes` to query for cases where A linearly precedes B irrespective of their hierarchical status.

(11) **Relationship:**	**Matches:**	**Does not match:**
`A precedes B`		

The relationship `hassister` finds instances where two nodes are sisters and it is not sensitive to linear order.

(12) **Relationship:**	**Matches:**	**Does not match:**
`A hassister B`		

The final basic relationship is `sameindex` which matches nodes with the same numerical index. This feature is sensitive to the way in which co-indexed nodes are handled in the annotation guidelines for the Penn Parsed Corpora of Historical English and other treebanks which adhere to the same type of annotation. Numerical indices are for example used to indicate a syntactic movement relationship between a displaced phrase and its trace.

(13) **Relationship:**	**Matches:**	**Does not match:**
`A sameindex B`		

In addition to the basic relationships, an additional set of special relationships can be used to narrow down results which have been found using the basic ones. Although this is not a logical necessity, the current implementation of the PaCQL search engine requires at least one basic relationship in order to allow for the use of these additional ones. The list in (14) gives an overview of those features.

(14) `haslabel`: match node label
`domswords`: match nodes dominating N orthographic words
`domswords<`: match nodes dominating less than N words
`domswords>`: match nodes dominating more than N words
`idomslemma`: POS-tag has child that has a specific lemma

The special relationship `haslabel` is useful to distinguish labels which have been matched by a Regular Expression. In the annotation scheme for the historical corpora under discussion, the label for a direct object is `NP-OB1` and the label for an indirect object is `NP-OB2`. Both are matched by the Regular Expression `NP-OB[12]`. We can add a coding column `direct` to the results by including the coding sections `direct:1` and `direct:0` shown below. This query finds all quantified objects (`Q-.*` matches quantifiers) and assigns the code 1 to the direct objects and 0 to the indirect objects.

(15)
```
NP-OB[12] idoms Q-.*

direct:1
NP-OB[12] haslabel NP-OB1

direct:0
NP-OB[12] haslabel NP-OB2
```

We can use `domswords` to match nodes that dominate a certain number of orthographic words. For example, the following query finds all one word subjects in passives in the annotation scheme of the Penn historical corpora. Here, `NP-SBJ` matches a subject and `VAN` matches a passive participle. The `domswords` condition narrows down the results to only include those where the subject is one word.

(16)
```
NP-SBJ hassister VAN
NP-SBJ domswords 1
```

Because PaCQL is designed specifically for the historical corpora, it is sensitive to the way in which orthographic words are represented in their annotation scheme. Sometimes a word is split into two terminal nodes if the syntax demands it. For example, the Icelandic suffixed article is split off from its noun in order to include a `D` node to encode definiteness. The article and the noun count as one word in PaCQL, even if they are separate nodes in the tree. Mismatches between the number of terminal nodes and the number of orthographic words also appear when we have traces of syntactic movement. For example, if the NP complement of a preposition has undergone A-bar movement (typically wh-movement; see textbooks on theoretical syntax for details), the annotation in the treebank is as follows:

(17)
```
(PP
   (P í)
   (NP *T*-1))
```

It is possible to find such structures by querying for NP complements of prepositions (`PP` = prepositional phrase) in which the NP does not dominate any words.

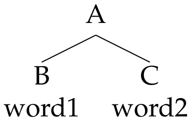
(18)
```
PP idoms NP
NP domswords 0
```

One can also query for nodes that dominate less than a certain number of words using `<domswords` in a query.

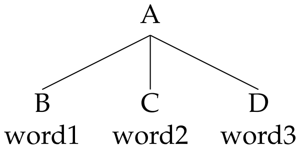
(19) **Relationship:**

`A domswords< 3`

Matches:



Does not match:

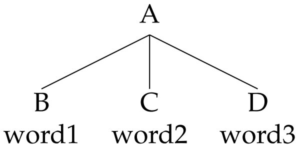


A parallel feature `domswords>` allows for finding nodes that dominate more than a certain number of words.

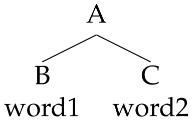
(20) **Relationship:**

`A domswords> 2`

Matches:



Does not match:



Finally, `idomslemma` can be used to search for a specific lemma (dictionary look-up form) of a word immediately dominated by a part-of-speech tag. This is particularly useful for a morphologically rich language like Icelandic. The following query finds noun phrases with the noun *hestur* 'horse' and it returns all inflected forms of the noun.

(21)
```
NP-.* idoms N-.*
N-.* idomslemma hestur
```

For example, the following result contains the noun *hestinn* which is in the accusative case with a suffixed article (split off as a separate `D-A` node as discussed above). This is possible because the Icelandic treebank is lemmatized. Note that in the Icelandic treebank, nominal elements which are inflected for case are tagged for case: `-N`=nominative, `-A`=accusative, `-D`=dative, `-G`=genitive. For example an accusative adjective is tagged `ADJ-A`, an accusative (singular) noun `N-A`, and an accusative definite article is `D-A`.

(22)
```
(NP-OB1
  (ADJ-A hálfan)
  (N-A hest$)
  (D-A $inn))
```

Before concluding this discussion of the syntactic relationships which are supported by PaCQL, it is worth noting that label matching in PaCQL uses standard Regular Expressions (as implemented in Python). This means that users of CorpusSearch will have to use slightly different labels from those that they are used to because CorpusSearch implements a non-standard version of RegularExpressions. We believe the change is justifiable because of the wide availability of documentation and tutorials for standard Regular Expressions.

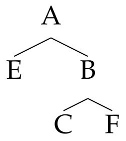
2.2 Coding queries

One of the reasons for the popularity of CorpusSearch is that most realistic research scenarios in historical syntax are not about searching for a pattern but rather about coding a number of results for a dependent variable and various independent variables which are to be analyzed. In PaCQL, every query is a coding query in the sense of CorpusSearch coding queries. The coding functionality is most powerful when coding sections are included in the query. This is demonstrated abstractly below:

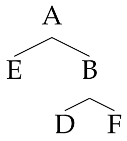
(23) **Query:**

```
A idoms B

foo:1
B idoms C

foo:0
B idoms D
```

Code foo:1



Code foo:2



This kind of a coding query is effectively equivalent to running the following two queries, combining the results, and labeling each individual result with 1 or 0 depending on the matched pattern. Recall that when the exact same label is used twice, it refers to the same node.

(24)
```
A idoms B
B idoms C
```

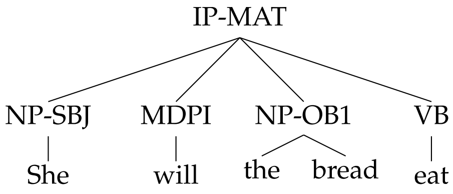
(25)
```
A idoms B
B idoms D
```

Above, we considered the contrast between object-verb (OV) and verb-object (VO) word order. The following examples demonstrate the two orders.

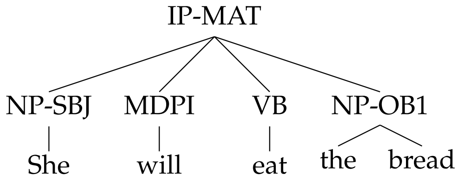
(26) a. She will [the bread eat]. (OV diagnostic)
b. She will [eat the bread]. (VO diagnostic)

In the historical treebanks, the analysis of the OV order is as in (27) and the VO order has the structure in (28) – omitting some irrelevant details for simplicity.

(27)



(28)



A simple coding query which distinguishes the two patterns is shown in (29). Note that for an actual study a more complex query would be needed but this version will suffice for the present purpose.

(29)
```
IP-(MAT|SUB) idoms MDPI

ov:1
MDPI sprec NP-OB[12]
NP-OB[12] sprec VB

ov:0
MDPI sprec VB
VB sprec NP-OB[12]
```

The main top section of the query finds finite clauses, including both main and subordinate clauses. The `ov:1` coding section uses `spred`, sisterwise precedence, in order to match sentences where a finite modal verb (`MDPI`) precedes a noun phrase object (`NP-OB[12]`) which in turn precedes an infinitival main verb `VB`. The second coding section `ov:0` matches the reverse order of the noun phrase object and the main verb.

In addition to the `ov` variable, we are perhaps interested in several independent variables which may be related to word order in the verb phrase. PaCQL includes a useful feature for this which we refer to as meta coding. It is possible to add a meta section to any query and this section allows us to specify various types of additional information to include in the output with each result. The following adds a simple meta section to our OV/VO query.

Recall that sections of a PaCQL query are separated by an empty line and this also applies to the meta section. Here, the meta section adds one property of the text in which the result was found and one property of a node which was matched. Including `text century` makes sure that an output value is added with the century in which the text was written and `node nodewords NP-OB[12]` states that the number of words dominated by the noun phrase object should be included in the result. Information about the date of writing is of course useful for studying and plotting the diachronic development. The weight of the object, measured in words, is interesting because heavy objects are more likely to undergo rightward extraposition, an independent type of a syntactic transformation which affects our VO/OV diagnostics (Wasow 1997; Stallings, MacDonald, and O'Seaghdha 1998; Thráinsson 2007; Stallings and MacDonald 2011).

For the Icelandic treebank which is currently hosted at www.treebankstudio.org, several options are available for meta coding. The codes which can be added are either specific to the text, the tree, or a particular node. The following is an overview of the supported text level coding:

(30) **Text level meta coding:**
`text textid` - id of the text
`text year` - (estimated) year the text was written
`text century` - century the text was written
`text genre` - main genre of the text
`text subgenre` - subgenre of the text
`text postnt` - 0 if written before New Testament translation, 1 otherwise
`text texttrees` - total number of trees in the text
`text meantreewords` - mean number of words per tree in the text
`text mediantreewords` - median number of words per tree in the text
`text meanwordletters` - mean number of letters per word in the text
`text lexicaldiversity` - type frequency of word forms divided by the total number of words in the text

Most of the above are fairly self-explanatory. It should be noted that `text texttrees` yields the number of trees in the sample included in the treebank. It is not a measure of the size of the entire book or manuscript. Some of the text level meta coding options reflect directly the purpose of PaCQL to be a tool for the digital humanities rather than a generic search program for tree structures. PaCQL has built-in functionality to analyze stylistic aspects of texts such as genre and lexical diversity. The lexical diversity metric which is used is described and discussed in the NLTK book (Bird, Klein, and Loper 2009).

Tree level meta coding includes a unique identifier for the tree in which a result was found as well as the number of orthographic words in the full tree. This is useful if one has the hypothesis that a certain linguistic pattern correlates with syntactic complexity.

(31) **Tree level meta coding:**
`tree treeid` - unique id for the tree
`tree treewords` - number of words in the tree

The final type of meta coding is node-specific and there are three types of such coding that can be added to the output of a query.

(32) **Node level meta coding:**
`node label A` - the label matched by A
`node nodestring A` - the string of leafs dominated by A
`node nodewords A` - the number of words dominated by A

It is useful to include *node label A* if `A` is a regular expression which matches a number of labels and the analysis of the results benefits from knowing which label it was in each case. The actual text which is dominated by a node `A` can be output using `node nodestring A` and this is particularly useful when some property of a phrase needs to be manually coded for some variable. For example, if we want to code a particular noun phrase for animacy, a linguistically interesting variable which is not annotated in the treebank, it is convenient to have a column in the output which lists just the noun phrases to be coded. We have seen `node nodewords A` above. This feature can be used to study heaviness effects in natural language which often involve rightward extraposition.

Coding queries sometimes become relatively complex and this is especially true when a regular expression is used several times in the query. It can make the query more readable to define variables in a definition section at the top of the query. In the following query, every instance of `object` is replaced with `NP-OB[12]` when the query runs.

(33)
```
define:
object NP-OB[12]

IP-(MAT|SUB) idoms MDPI

ov:1
MDPI sprec object
object sprec VB

ov:0
MDPI sprec VB
VB sprec object
```

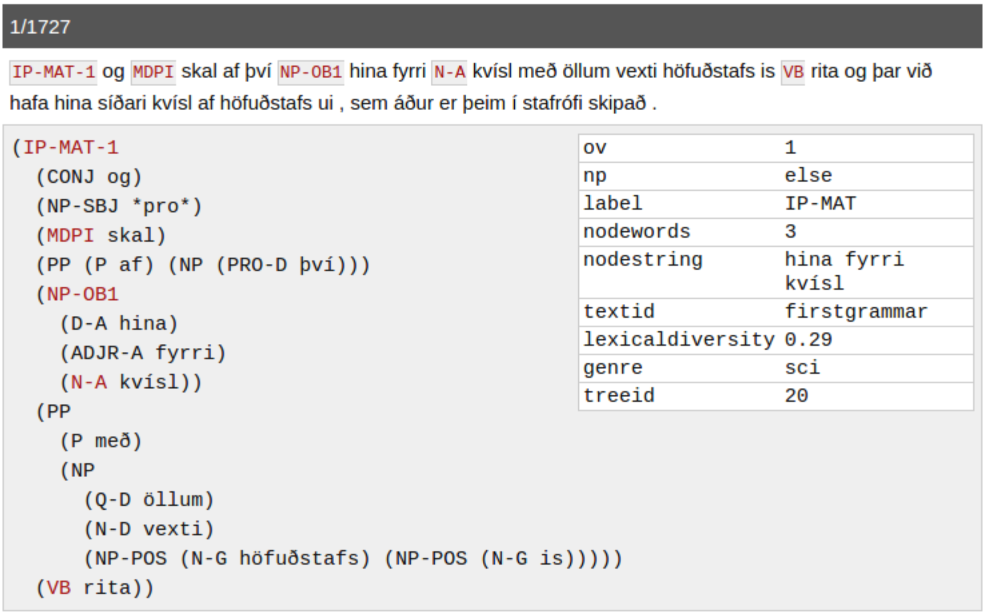
```
meta:
text century
node nodewords object
```

Putting together some of the techniques that have been discussed, let us consider a slightly more complex query. We are still focusing on the OV/VO variable.

(34)

```
define:
modal MD[PD][IS]
object NP-OB[12]

IP-(MAT|SUB) idoms modal

ov:1
modal sprec object
object sprec VB

ov:0
modal sprec VB
VB sprec object

np:pro
object idomsonly PRO-.*

np:quant
object idoms Q[RS]?-.*

np:else
object idoms .*

meta:
node nodewords object
node nodestring object
node label IP-(MAT|SUB)
text genre
text lexicaldiversity
```

The query in (34) defines shorthand variables for finite modals and noun phrase objects. Its coding sections define two variables, `ov` which indicates the word order under investigation and `np` which indicates which type of a noun phrase the object is. The value of `np` is `pro` when we have a pronoun object, `quant` when it is a quantified object, and `else` otherwise. The meta section extracts the number of words in the object and the exact string dominated by the object phrase. Here, *node label IP-(MAT|SUB)* writes out whether the Regular Expression for the clause matched a matrix clause or a subordinate clause and we also have two stylistic variables for genre and lexical diversity. A beginner may not find it easy to construct a query at this level of complexity. However, the query language has been employed successfully in a linguistics class at the master's level at the University of Iceland and by working through a series of examples, the students gained good knowledge of the language, suggesting that researchers in the humanities should be able to do so in general.



1/1727

IP-MAT-1 og MDPI skal af því NP-OB1 hina fyrri N-A kvísl með öllum vexti höfuðstafs is VB rita og þar við hafa hina síðari kvísl af höfuðstafs ui , sem áður er þeim í stafrófi skipað .

```
(IP-MAT-1
  (CONJ og)
  (NP-SBJ *pro*)
  (MDPI skal)
  (PP (P af) (NP (PRO-D því)))
  (NP-OB1
    (D-A hina)
    (ADJR-A fyrri)
    (N-A kvísl))
  (PP
    (P með)
    (NP
      (Q-D öllum)
      (N-D vexti)
      (NP-POS (N-G höfuðstafs) (NP-POS (N-G is)))))
  (VB rita))
```

ov	1
np	else
label	IP-MAT
nodewords	3
nodestring	hina fyrri kvísl
textid	firstgrammar
lexicaldiversity	0.29
genre	sci
treeid	20

Figure 2
An example PaCQL result.

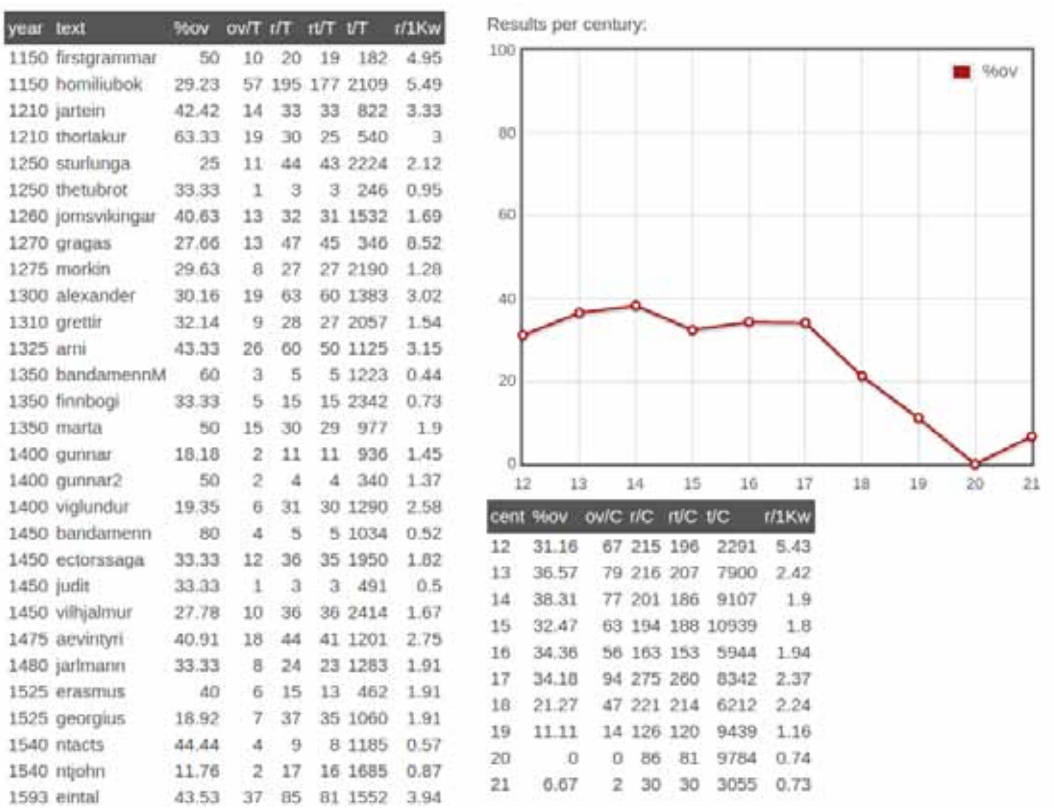
3. Output

Having discussed the PaCQL query language, we will now turn to the way in which the search engine displays its output. As in other aspects of this software, the design is intended to meet the demands of the digital humanities. The example results which we discuss in this section are all based on running the query in (34) on the Icelandic treebank as it is configured at www.treebankstudio.org.

In most cases, users will first want to explore the results in the "web output" mode which is the default setting. An example of a result which is displayed this way is shown in Figure 2.

The text of the entire tree which contains the match is displayed on top and below that the part of the tree which matched the query is shown in the labeled bracketing format which is familiar to users of the Penn treebank and other similar corpora. The top node of this partial tree is referred to as the query's anchor and the anchor is simply the first label entered in the main section of the query. It is possible to display the entire tree in the results by unchecking the "Anchors only" checkbox in the user interface. The nodes matched by the query are highlighted in the text as well as the tree. Next to the tree, all the columns associated with the current result are displayed. This includes the output of the coding sections as well as the meta section.

In almost all actual studies of quantitative diachronic syntax, the user will want to move on to tabulated results when the query has been configured to return the appropriate results. Changing the output mode to "TSV" results in a list of tab separated values of the following type. This kind of an output can be imported into R, SPSS, Excel, or any other software which supports further analysis.



year	text	%ov	ov/T	r/T	rt/T	t/T	r/1Kw
1150	firstgrammar	50	10	20	19	182	4.95
1150	homiliubok	29.23	57	195	177	2109	5.49
1210	jartein	42.42	14	33	33	822	3.33
1210	thorlakur	63.33	19	30	25	540	3
1250	sturlunga	25	11	44	43	2224	2.12
1250	thetubrot	33.33	1	3	3	246	0.95
1260	jomsvikingar	40.63	13	32	31	1532	1.69
1270	gragas	27.66	13	47	45	346	8.52
1275	morkin	29.63	8	27	27	2190	1.28
1300	alexander	30.16	19	63	60	1383	3.02
1310	grettir	32.14	9	28	27	2057	1.54
1325	arni	43.33	26	60	50	1125	3.15
1350	bandamennM	60	3	5	5	1223	0.44
1350	finnbogi	33.33	5	15	15	2342	0.73
1350	marta	50	15	30	29	977	1.9
1400	gunnar	18.18	2	11	11	936	1.45
1400	gunnar2	50	2	4	4	340	1.37
1400	viglundur	19.35	6	31	30	1290	2.58
1450	bandamenn	80	4	5	5	1034	0.52
1450	ectorssaga	33.33	12	36	35	1950	1.82
1450	judit	33.33	1	3	3	491	0.5
1450	vilhjalmur	27.78	10	36	36	2414	1.67
1475	aevintyri	40.91	18	44	41	1201	2.75
1480	jarlmann	33.33	8	24	23	1283	1.91
1525	erasmus	40	6	15	13	462	1.91
1525	georgius	18.92	7	37	35	1060	1.91
1540	ntacts	44.44	4	9	8	1185	0.57
1540	ntjohn	11.76	2	17	16	1685	0.87
1593	eintal	43.53	37	85	81	1552	3.94

cent	%ov	ov/C	r/C	rt/C	t/C	r/1Kw
12	31.16	67	215	196	2291	5.43
13	36.57	79	216	207	7900	2.42
14	38.31	77	201	186	9107	1.9
15	32.47	63	194	188	10939	1.8
16	34.36	56	163	153	5944	1.94
17	34.18	94	275	260	8342	2.37
18	21.27	47	221	214	6212	2.24
19	11.11	14	126	120	9439	1.16
20	0	0	86	81	9784	0.74
21	6.67	2	30	30	3055	0.73

Figure 3
An automatically generated PaCQL summary.

ov	np	label	nodewords	nodestring	textid	lexicaldiversity	genre
1	else	IP-MAT	3	hina fyrri kvísl	firstgrammar	0.29	sci
1	else	IP-MAT	3	þessa stafi átta	firstgrammar	0.29	sci
1	else	IP-SUB	1	máli	firstgrammar	0.29	sci

Going back to the "Web output" mode, the software is designed to prepare an automatic summary of the results which can often serve as an initial analysis. This summary is opened by selecting the "Display summary" option from the user interface. The summary for our sample query is shown in Figure 3.

The first coding variable which only has the values 1 and 0 is automatically detected as the dependent variable in the summary. In this case, it is our OV/VO variable. A historical syntactician will often want to visualize the diachronic development of the dependent variable and the summary includes a plot of the proportion of object-verb word order in each century which is represented in the corpus. The summary also contains tabulated results showing results per text (r/T), result trees per text (rt/T), and total trees per text (t/T). The summary tables include the rate at which the dependent variable has the value 1 (if one is defined) as well as the number of results per 1000 words for the relevant text (to evaluate result density). Those metrics are output in two tables. The first tables breaks the results down by text and the second table by century.

Of course, more sophisticated analysis as shown in Figure 1 requires additional software like R, but this is what the TSV output is designed for. Even so, the automatically generated summary is often sufficient to get a good initial idea of important trends in the data.

Although this article focuses on the side of PaCQL which is most important for a linguist rather than underlying technical issues, it is worth noting that the output of many typical PaCQL queries is returned relatively fast. Unlike CorpusSearch, which is commonly used for searching this type of a treebank, PaCQL employs indexing to speed up access to the corpus and the index remains in memory on the server in order to improve the user experience. Various common types of queries run in less than one second or in just a few seconds when searching and coding the IcePaHC corpus, which is close to 1 million words in size. Naturally, the execution time can be longer if the query is particularly complex.

4. Some current shortcomings and future work

Although PaCQL is already useful for various kinds of studies, the system is fairly new and thus it can be expected that some issues will be revealed by its use. Those will have to be addressed in future updates to the software. Also, while PaCQL presentely supports several features that are available in alternative treebank search software, and adds some of its own, there are still some features that have not been implemented. For example, there is little support for negation in the query language so it is not possible to enter a simple negation operator to ask for all cases where label A, for example, does not immediately dominate label B. For the time being, the relevant type of a result can often be retrieved by a coding query where the positive complement of the desired negation is assigned one code and the remaining results are assigned another code – the latter capturing the negation.

There are plans to make queries with negation more streamlined in the future and expand the range of available features in general, including support for equality testing across matched nodes and quantification. Further development will aim to support a variety of useful features which have been developed in other systems like TIGERSearch (Lezius, Biesinger, and Gerstenberger 2002), INESS-search (Meurer, Butt, and King 2012), and PML Tree Query (Pajas and Štěpánek 2009), while maintaining our focus on providing output that is optimized for use in linguistic research, notably in historical syntax. Those future steps in the development will be informed by a systematic evaluation of PaCQL's expressiveness (Lai and Bird 2010).

5. Conclusion

This article has described PaCQL (Parsed Corpus Query Language), a research tool which is designed to address various specific needs of the digital humanities. We described the way in which queries can be constructed to answer questions about historical syntax and we explained how the output of a PaCQL query can be interpreted and exported to a format that is appropriate for the type of software that is generally used in the analysis of historical syntacticians.

PaCQL is currently only configured to search IcePaHC, the Icelandic Parsed Historical Corpus, but the system is under active development and one of the primary goals for the future will be to make the system available to the users of other treebanks. We furthermore aim to release the PaCQL search engine under a free and open source software license in the near future.

References

Augustinus, Liesbeth, Vincent Vandeghinste, and Frank Van Eynde. 2012. Example-based treebank querying. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3161–3167. ELRA.

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Sebastopol, CA.

Britto, Helena and Charlotte Galves. 2009. The Tycho Brahe Parsed Corpus of Historical Portuguese (TBCHP). University of Campinas.

Hróarsdóttir, Þorbjörg. 2000. *Word order change in Icelandic: from OV to VO*. John Benjamins Publishing Company, Amsterdam.

Kemenade, Ans van. 1987. *Syntactic case and morphological case in the history of English*. Foris, Dordrecht.

Kroch, Anthony, Beatrice Santorini, and Lauren Delfs. 2004. Penn-Helsinki Parsed Corpus of Early Modern English. CD-ROM. First Edition. Size: 1.8 million words.

Kroch, Anthony and Ann Taylor. 2000a. Diachronic syntax: models and mechanisms. In S. Pintzuk, G. Tsoulas, and A. Warner, editors, *Verb-Object Order in Early Middle English*. Oxford University Press, Oxford, pages 132–163.

Kroch, Anthony and Ann Taylor. 2000b. Penn-Helsinki Parsed Corpus of Middle English. CD-ROM. Second Edition. Size: 1.3 million words.

Lai, Catherine and Steven Bird. 2010. Querying linguistic trees. *Journal of Logic, Language and Information*, 19(1):53–73.

Lezius, Wolfgang, Hannes Biesinger, and Ciprian Gerstenberger. 2002. TIGERSearch Manual.

Lightfoot, David. 1991. *How to set parameters: Arguments from language change*. MIT Press, Cambridge, MA.

Lightfoot, David. 1999. *The development of language: Acquisition, change, and evolution*. Blackwell, Malden, MA.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Martineau, France. 2011. MCVF corpus of historical French.

Meurer, Paul, Miriam Butt, and Tracy Holloway King. 2012. INESS-Search: A search system for LFG (and other) treebanks. In *Proceedings of the LFG'12 Conference*, pages 404–421.

Pajas, Petr and Jan Štěpánek. 2009. System for querying syntactically annotated corpora. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 33–36. Association for Computational Linguistics.

Randall, Beth, 2005. *CorpusSearch 2 User's Guide*. University of Pennsylvania.

Resnik, Philip and Aaron Elkiss. 2005. The linguist's search engine: an overview. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 33–36. Association for Computational Linguistics.

Rohde, Douglas LT. 2004. Tgrep2 user manual.

Rögnvaldsson, Eiríkur. 1994/1995. Breytileg orðaröð í sagnlið. *Íslenskt mál og almenn málfræði*, 16–17:27–66.

Rögnvaldsson, Eiríkur. 1996. Word order variation in the VP in Old Icelandic. *Working Papers in Scandinavian Syntax*, 58:55–86.

Stallings, Lynne M. and Maryellen C. MacDonald. 2011. It's not just the "heavy NP": Relative phrase length modulates the production of heavy-NP shift. *Journal of Psycholinguistic Research*, 40(3):177–187.

Stallings, Lynne M., Maryellen C. MacDonald, and Padraig G. O'Seaghdha. 1998. Phrasal ordering constraints in sentence production: Phrase length and verb disposition in heavy-NP shift. *Journal of Memory and Language*, 39(3):392 – 417.

Thráinsson, Höskuldur. 2007. *The syntax of Icelandic*. Cambridge University Press, Cambridge.

Wallenberg, Joel, Anton Karl Ingason, Einar Freyr Sigurðsson, and Eiríkur Rögnvaldsson. 2011. Icelandic Parsed Historical Corpus (IcePaHC). Version 0.9.

Wasow, Thomas. 1997. End-weight from the speaker's perspective. *Journal of Psycholinguistic Research*, 26(3):347–361.