The Linear Arrangement Library. A new tool for research on syntactic dependency structures.

Lluís Alemany-Puig[1,*] Juan Luis Esteban[2,†] Ramon Ferrer-i-Cancho[3,*]

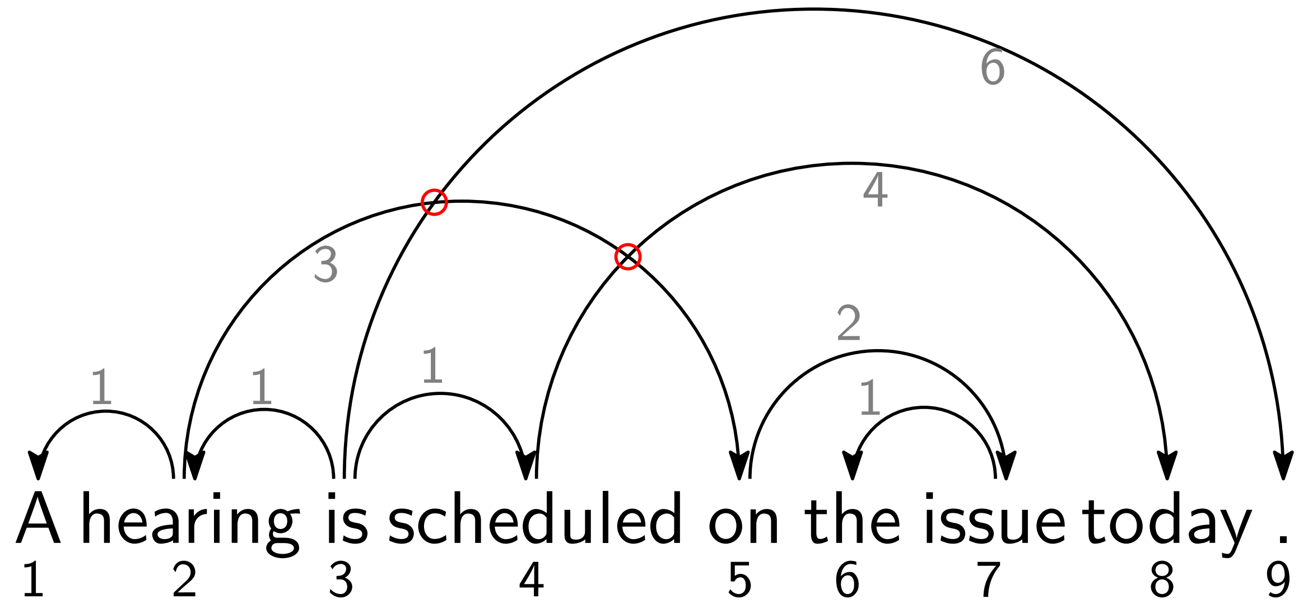
Quantitative Dependency Syntax: Quantitative Linguistics (QL) + Dependency Syntax (DS)

Study of metrics inherent of syntactic dependency structures to unveil statistical laws and understand the patterns in these structures.

Dependency Distance minimization principle (DDm) can be shown to have a strong presence in sufficiently long sentences by statistical analysis (QL) of syntactic measurements of sentences (DS).

Year-long studies have produced dozens of metrics in pursue of said goals. However, code is not usually shared among researchers, so researchers without (or with little) programming skills cannot reproduce previous results or reuse the metrics.

Open source an global tool to which everyone can contribute and share their code with other researchers.



```
import lal
rt = lal.graphs.from_head_vector_to_rooted_tree([2,3,0,3,2,7,5,4,3])
```

Structural measures

Calculate metrics on individual syntactic dependency trees that do not depend on a the linear ordering of the words.

```
# expected sum of syntactic dependency distances
e_sum = lal.properties.exp_sum_edge_lengths(rt)
e_sum_plan = lal.properties.exp_sum_edge_lengths_planar(rt)
e_sum_proj = lal.properties.exp_sum_edge_lengths_projective(rt)
# expected number of crossings
e_cross = lal.properties.exp_number_crossings(rt)
# variance of the number of crossings
v_cross = lal.properties.var_number_crossings_tree(rt)
# others
mhd = lal.properties.mean_hierarchical_distance(rt)
hub = lal.properties.hubiness(rt)
center = lal.properties.tree_centre(rt)
centroid = lal.properties.tree_centroid(rt)
diameter = lal.properties.tree_diameter(rt)
```

Statistical testing: need for generation of trees and arrangements of trees

Hypothesis testing:

- Is metric X significantly larger when observed in the natural word order in a syntactic dependency structure significantly larger than when it is observed in random permutations of the words of the same structure?
- Is metric X significantly larger when observed in the given structure than when observed in other syntactic dependency structures of the same size?

Arrangement generation combinations in LAL: $\left\{ \begin{matrix} exhaustive \\ random \end{matrix} \right\} \times \left\{ \begin{matrix} unconstrained \\ planar \\ projective \end{matrix} \right\} \times \left\{ \begin{matrix} free\ trees \\ rooted\ trees \end{matrix} \right\}$

Tree generation combinations in LAL: $\left\{ \begin{matrix} exhaustive \\ random \end{matrix} \right\} \times \left\{ \begin{matrix} labeled \\ unlabeled \end{matrix} \right\} \times \left\{ \begin{matrix} free \\ rooted \end{matrix} \right\}$

Generating trees and arrangements

Generate structures of a given fixed size both uniformly at random and exhaustively, and arrangements of these structures both uniformly at random and exhaustively.

```
arr_gen = lal.generate.all_projective_arrangements(tree)
while not arr_gen.end():
    arr = arr_gen.yield_arrangement()
    # ...
arr_gen = lal.generate.rand_planar_arrangements(tree)
for i in range(0, 10000):
    arr = arr_gen.yield_arrangement()
    # ...
tree_gen = lal.generate.all_ulab_free_trees(10)
tree_gen = lal.generate.all_lab_rooted_trees(10)
while not tree_gen.end():
    tree = tree_gen.yield_tree()
    # ...
tree_gen = lal.generate.rand_ulab_rooted_trees(10)
for i in range(0, 10000):
    tree = tree_gen.yield_tree()
    # ...
```

Linear ordering measures

Calculate metrics on individual syntactic dependency trees that depend on a linear ordering of the words: sum of syntactic dependency distances, number of edge crossings...

```
D = lal.linarr.sum_edge_lengths(rt)
C = lal.linarr.num_crossings(rt)
dep_flux = lal.linarr.compute_flux(rt)
h = lal.linarr.head_initial(rt)
```

Produce minimum linear arrangements of syntactic dependency trees

```
Dmin, arr_min = lal.linarr.min_sum_edge_lengths(rt)
Dmin_plan, arr_min_plan = lal.linarr.min_sum_edge_lengths_planar(rt)
Dmin_proj, arr_min_proj = lal.linarr.min_sum_edge_lengths_projective(rt)
```

And many more...

Process collections of treebanks

Easily process individual treebanks and collections of treebanks. For every tree in a treebank LAL can calculate several metrics and output them in a .csv file (or other). The resulting data can be used to test statistical laws in the treebank/collection.

LAL processes treebanks in head vector format, therefore it can process treebanks including, but not limited to, Universal Dependencies, Surface Universal Dependencies, treebanks in Stanford/Prague notation.

There exist tools that transform and preprocess treebanks into the head vector format.

Processing collection of treebanks with LAL

Make sure that there are no errors in a collection

```
errlist = lal.io.check_correctness_treebank_collection("main.txt")
```

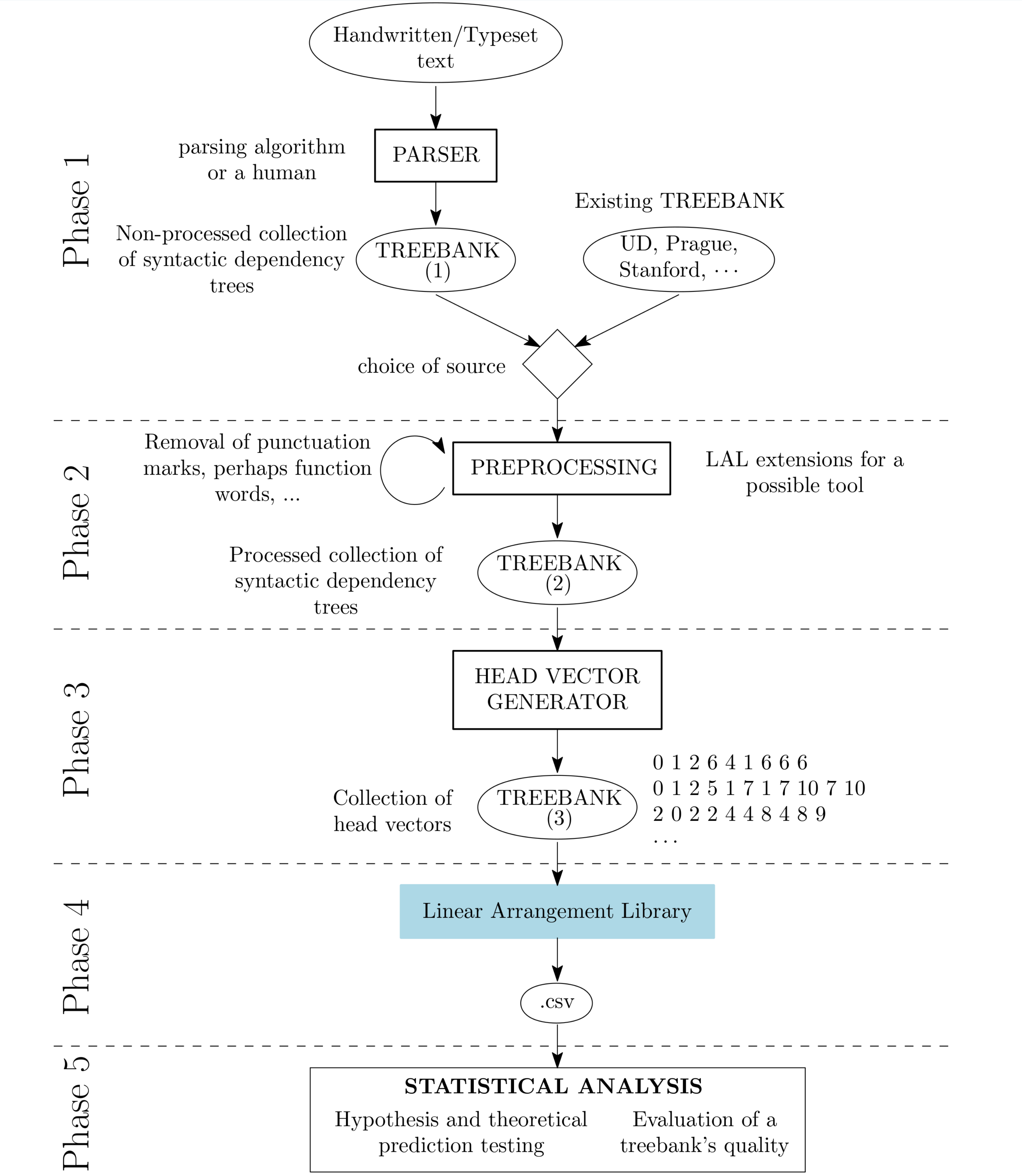
Easily process a collection in a single line

```
err = lal.io.process_treebank_collection("main.txt", "output_directory")
```

Customize the data produced during processing

```
tbcolproc = lal.io.treebank_collection_processor()
# The object will compute all features by default
err = tbcolproc.init("main.txt", "output_directory")
if err == lal.io.treebank_error_type.no_error:
    # Remove all the features
    tbcolproc.clear_features()
    # Now we add some metrics
    tbcolproc.add_feature(lal.io.treebank_feature.num_nodes)
    tbcolproc.add_feature(lal.io.treebank_feature.sum_edge_lengths)
    tbcolproc.add_feature(lal.io.treebank_feature.exp_sum_edge_lengths)
    tbcolproc.add_feature(lal.io.treebank_feature.min_sum_edge_lengths)
    # Process the treebank file...
    err = tbcolproc.process()
```

Standard research pipeline


Phase 1
Handwritten/Typeset text
parsing algorithm or a human
PARSER
Non-processed collection of syntactic dependency trees
TREEBANK (1)
Existing TREEBANK
UD, Prague, Stanford, · · ·
choice of source
Phase 2
Removal of punctuation marks, perhaps function words, ...
PREPROCESSING
LAL extensions for a possible tool
Processed collection of syntactic dependency trees
TREEBANK (2)
Phase 3
HEAD VECTOR GENERATOR
Collection of head vectors
TREEBANK (3)
0 1 2 6 4 1 6 6 6
0 1 2 5 1 7 1 7 10 7 10
2 0 2 2 4 4 8 4 8 9
...
Phase 4
Linear Arrangement Library
.csv
Phase 5
STATISTICAL ANALYSIS
Hypothesis and theoretical prediction testing
Evaluation of a treebank's quality

State-of-the art algorithms to calculate baselines on a tree

The baselines on D, the sum of dependency distances, and C, the number of syntactic dependency crossings, that can be calculated using LAL. 'Unconstrained', 'Planar' and 'Projective' are the different constraints under which the 'Minimum', 'Expected' and 'Maximum' values can be calculated. *: available in LAL but article not published yet; †: the minimum value of C is trivially 0 for every tree.

	D			C
	Unconstrained	Planar	Projective	Unconstrained
Minimum	(Shiloach, 1979), (Chung, 1984)	(Hochberg & Stallmann, 2003), (Alemany-Puig et al., 2022)	(Gildea & Temperley, 2007), (Alemany-Puig et al., 2022)	†
Complexity	$O(n^{2.2}), O(n^2)$	$O(n)$	$O(n)$	
Expected	(Ferrer-i-Cancho, 2004)	*	(Alemany-Puig & Ferrer-i-Cancho, 2021)	(Verbitsky, 2008)
Complexity	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Maximum	Under study	In progress	In progress	Under study
Complexity		$O(n)$	$O(n)$	

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament de Ciències de la Computació

IMTech

[1] lluis.alemany.puig@upc.edu
[2] esteban@cs.upc.edu
[3] ramon.ferrer@upc.edu

[*] Complexity and Quantitative Linguistics Lab, LARCA Research Group, Computer Science Department, Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034, Barcelona, Spain
[†] Computer Science Department, Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034, Barcelona, Spain

webpage https://cqllab.upc.edu/lal *source code* https://github.com/LAL-project Syntax Fest 2021 (conference date: 21st - 24th March 2022)

Presentation made with IPE 7.2.24 (ipe.otfried.org)