CLSINFRA COMPUTATIONAL LITERARY STUDIES INFRASTRUCTURE

D8.1

REPORT OF THE TOOLS FOR THE BASIC NATURAL LANGUAGE PROCESSING (NLP) TASKS IN THE CLS CONTEXT

Authors: Silvie Cinková, Julie M. Birkholz, Ingo Börner, Tess Dejaeghere, Serge Heiden, Maarten Janssen, Michal Křen, Alvaro Perez Pozo

Date: March 09, 2023

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004984

Project Acronym: CLS INFRA
Project Full Title: Computational Literary Studies Infrastructure
Grant Agreement No.: 101004984

Deliverable/Document Information

Deliverable No.: D8.1
Deliverable Title: REPORT OF THE TOOLS FOR THE BASIC NLP TASKS IN THE CLS CONTEXT
Authors: Silvie Cinková, Julie M. Birkholz, Ingo Börner, Tess Dejaeghere, Serge Heiden, Maarten Janssen, Michal Křen, Alvaro Perez Pozo
Dissemination Level: PUBLIC

Document History

Version/Date	Changes/Approval	Approved by
1.0.0.		Maciej Eder

Table of Contents

List of Terms and Abbreviations/Acronyms

Term	Explanation
API	Application Programming Interface - how computers interact with web services. Most web services offer an API access.
authorship attribution	Determining the author of a text
bag-of-words	Bag-of-words methods and approaches do not consider words in their context, and typically they do not use any additional markup.
CoNLL-U, conllu, conll-u	A common format of morphological and syntactic annotation, resembling a comma-separated-values file with commented text rows
corpus	Machine-readable collection of texts, typically containing metadata about the texts, and some additional markup within the texts.
Feature Engineering	Identification of features that help the computer to learn to mimic human judgments or natural phenomena in Machine Learning
HTR	Hand-written Text Recognition. Automatic transcription of facsimile images to machine-readable text for hand-written documents.
IAA	Inter-Annotator Agreement (aka inter-rater/inter-coder agreement) in Machine Learning is the extent to which different people, given the same annotation instructions, come independently to the same result when creating training data to train the computer for a given task.
in-line (annotation)	Annotation (aka markup) that is located in the same file as the text it annotates. E.g., the tags in a html page are in-line annotation.
JSON	JavaScript Object Notation - a both machine and human-readable text format capable of storing multiple types of information and their mutual relations (e.g., it is possible to transform a table in JSON). Visually it is characterized by square brackets and curly braces.
lemma	Basic dictionary form of a word, e.g., verb in infinitive or noun in nominative singular
lemmatization	providing each token in a corpus with its lemma
Machine Learning	Statistical techniques to train computers to mimic human judgments or natural phenomena. It is typically used with complex decisions that cannot be captured by rules. The most recent Machine-Learning systems use neural networks rather than traditional statistical models.
markup (aka annotation)	Tags and labels added to a text, carrying some interpretation of the text. E. g. the part of speech of each word, or a span of speech attributed to a given person.
metadata	Data about the data, e.g., the information about author and year of publication accompanying a text.

NER	Named-Entity Recognition. Extraction of proper names, and, in a broader scope, also of other interesting or important entities, such as dates, currencies, or personal descriptors like academic titles or professions.
NLP	Natural Language Processing
OCR	Optical Character Recognition. Automatic transcription of facsimile images to machine-readable text for printed documents.
parser	A computer program that analyzes character strings or symbols according to a formal grammar. With natural languages, it is supposed to help extract meaning from text by analyzing syntactic and morphological rules, which can be enriched with additional information (e.g., named entity recognition or some semantic labels).
part of speech	Fundamental morphological characteristics of a word, e.g., noun, verb, adjective
regular expressions	Computational text search technique consisting of a "language" to create templates and match text against them; cf. wild card search
Sentiment Analysis	A technique to assess whether a text is expressing something positive or negative.
stand-off (annotation)	Annotation (aka markup) that is located in a different file than the text it annotates. The annotation file relates to the original file with character indices (offsets).
tag	Usually morphological tag - a label on each token in a corpus stating its part of speech and its relevant morphological features, such as case, tense, or number
tagger	A computer program that assigns labels (tags) to character strings or symbols; a typical example is a morphological tagger that labels each word with its part of speech and similar information.
tagging	Providing each token in a corpus with a morphological tag
Text Modeling	Quantitative assessment of diverse characteristics of texts, relating either to content or style
token	Aka *running word* - each occurrence of a word form in a corpus
treebank	A text corpus enriched with syntactic parsing
type	Unique word form
wild card search	A technique to underspecify a text search. E.g., "ca.*" would match against all words starting with "ca", no matter their length.

Executive Summary

This report lists and describes a selection of Natural Language Processing (NLP) tools which are considered to form a Corpus-Enrichment and NLP toolchain for common CLS research tasks. The tools were selected to be:

- safely positioned in their life cycle, i.e., state-of-the art, and mature as well as continuously maintained, or in development and promised as CLS Infra Deliverables by March 2025
- as multilingual as possible (beyond English and several major European languages)
- as interoperable as possible with other tools and texts in other languages.

1. Introduction

Historical-literary textual materials are the most important sources for literary scholars. The increasing provision and availability of digitized and machine-readable optical-character recognized sources affords the use of computational techniques to assist in corpus building and analysis. Natural Language Processing is the computational task of processing natural language or textual information. This encompasses a wide range of tasks and thus methods and tools. This report aims to guide the reader through a selection of state-of-the-art NLP tools, which CLS INFRA considers particularly relevant for the CLS community, given its extraordinary diversity of languages, language varieties, and text domains. The list is far from exhaustive, but what sets it apart from larger lists (e. g. TAPoR[1]) is its multilingual, deliberately non-English language-centric focus, and the fact that these tools have already been chosen or are being considered to become part of a multilingual tool chain developed within the CLS INFRA project, in coordination with the Inventory of existing data sources and formats (D6.1).

This technical report attentively draws on the insights of D4.1 (Rossum and Šeļa, 2022), according to which there is a considerable gap in the distribution of skills between beginning DH researchers with a traditional scholarly background and very advanced DH researchers with very solid competences in data science and programming. The D4.1 survey reflects a lack of education and guiding for intermediate DH researchers, who are motivated to learn some programming and data science but feel intimidated by the vastness of these disciplines[2].

Relating to D4.1, the tool selection mainly addresses two research phases: Collection and Analysis, and it addresses the following skills (D4.1, pp. 9-14):

- Corpus Building (C2),
- Text Annotation (C6),
- Text / Corpus Analysis (A1),

[1] https://tapor.ca/

[2] This technical report addresses these researchers, providing orientation in NLP tools that are ready to use with no or very basic programming skills, pointing the reader to D3.2 for workflow examples (to appear simultaneously with this technical report) and especially to D3.3, D3.4, and D3.5 (to appear in 2024).

- Text Modeling and Feature Engineering (A3).

Text Modeling and Feature Engineering, along with text annotation and corpus building, were all among the most demanded skills in the D4.1 survey.

The technical report pinpoints the linguistic markup as the cornerstone of Text Annotation, Text / Corpus Analysis, and Text Modeling and Feature Engineering.

Other major markup topics, such as Named Entity Recognition, Sentiment Analysis, as well as the standards of poetry or drama processing are tackled in the individual descriptions of the corresponding tools.

Section 8 of this report presents the logic of the tool descriptions in connection with the current (preliminary) data model for data and tools within CLS Infra.

2. The Importance of Linguistic Markup in NLP/Text Mining

The object of inquiry in CLS is texts and their content or/and style. Research questions are solved by various *text mining* techniques. Text mining, also called *text analytics*, is an overarching term for extracting patterns and non-obvious relationships from text. In the industry, it is employed in chatbots, automatic summarization of online reviews, customer profiling, and many other applications. The major text-mining techniques are, among others, concept/entity extraction, sentiment analysis, text categorization, text clustering, topic modeling, and authorship attribution.

These techniques can, to some extent, be performed without any linguistic markup and regardless of word context (aka *bag-of-words* approach). Especially in languages with little inflection and fixed word order, such as English, they can bring about reasonably good results. Nevertheless, many languages and many research questions require linguistically informed text mining, which needs *linguistic markup* (aka *linguistic annotation*).

3. The Origins of Linguistic Markup

The concept of linguistic markup arose from diverse linguistic theories (e.g. Bloomfield and Hockett, 1984; Chomsky and Lightfoot, 2002; Gerdes et al., 2013; Gréciano and Schumacher, 2015; Halliday and Matthiessen, 2013; Hjelmslev and Whitfield, 1961; Mel'cuk, 2021) as well as from canonic language-specific grammars. Before big textual data became available, most language material hailed from introspection or field work.

The onset of big text corpora since the early 1990s (e.g. John Sinclair's Svartvik, 2011, with earlier landmarks such as Kučera and Francis, 1967) was originally driven by the needs of lexicographers rather than those of formal linguists (Kermes and Evert, 2001). However, in their quest to "know a word by the company it keeps" (Firth, 1957, p. 11), the lexicographers had always been aware of the importance of structural linguistic information for lexical semantics, so *part-of-speech tagging* and even *syntactic parsing* was aiding the search from the earliest corpus releases (Leech et al., 1994; Santorini, 1990).

Automatic linguistic markup was first provided with rule-based scripts, whose performance was limited. At the turn of the millennium, statistical machine-learning methods were introduced and widely embraced, and with them came a demand for big gold-standard manually annotated corpora to train the systems on. One of the first and most influential projects was the Penn Treebank (Marcus et al., 1994), followed by numerous similar projects in other languages.

Although the annotation schemes were drawing on the contemporary linguistic theories, their theoretical purity was soon jeopardized by the mundane requirements of annotation consistency (measured as inter-annotator agreement), annotation speed, and the resulting performance of the trained tools. Hence, the creation of linguistic resources as backing evidence for linguistic theories was gradually giving way to practical applications of language professionals outside the realm of formal linguistics (Hovy et al., 2006; Lavid, 2010; Teufel et al., 1996).

4. Cross-lingual Standardization of Linguistic markup

In the 2010s, many large-scale projects to build language resources were running in many language communities, with intermittent releases not exceptionally spanning a decade. These projects started from different theoretical backgrounds and were making different theoretical concessions to arising practical obstacles throughout the years. There were early significant community efforts, e.g., the European projects Eagles, then Multext, followed by Multext-East (Dimitrova et al, 1998), resulting in a common annotation scheme (morpho-syntax) for a dozen European languages, as well as PAROLE (Vignaux, 2000), but most language resources (as well as algorithms) were hardly interoperable, with competing annotation schemes (tagsets, tag definitions) even within one language.

The change came with the *Universal Dependencies* (Agić et al., 2015), drawing on (Marneffe et al., 2014; Zeman, 2008). This corpus-annotation project was the first to succeed in building up a wide multilingual community of corpus users and creators willing to use a universal annotation scheme and adapt it to their language. The community has been active and expanding since the first multilingual release in 2015. An update is released every six months. The current release (2.11) provides 243 corpora to 138 languages (Zeman et al., 2022). For many of these languages these corpora are the first NLP resources at all. The corpora are used as ground truth for training modern neural-network based syntactic parsers (e.g. spaCy (Montani et al., 2023) and UDPipe (Straka and Straková, 2022)).

The benefit of a cross-lingually universal annotation scheme is enormous:

1) The users do not have to study individual tag sets and syntactic theories to extract information from a new corpus.
2) Comparative studies are possible across languages.
3) When creating a corpus for a new language, sometimes a UD-trained parser can pre-annotate the texts using the language model of a closely related language, saving time compared to annotation from scratch.
4) The annotation scheme is as theory neutral as possible, optimized for information extraction. The annotators do not have to be formal linguists by training.

5. Information Extraction Exploiting Linguistic Markup: Query Languages

The linguistic markup adds linguistic structure to texts. It usually consists of three components:

1) lemmatization
2) morphological tagging
3) syntactic parsing

The lemma is the basic dictionary form of a word form, for instance the infinitive of a verb or nominative singular of a noun. Querying lemmas abstracts from the variance of the possible word forms, as well as from letter case.

Morphological tagging helps abstract from concrete words. It is particularly suitable to find typical collocates (words combining with the word in question). For instance, when researching how writers described women, a good information extraction strategy is to extract adjectives preceding a set of lemmas of nouns denoting women (e. g. *female, wife, mother, girl, hag, beauty, chick*...). An extension to the list of nouns could be the morphological tag of a proper noun with the female gender. Another use case could be extracting familial relationships from unstructured text by looking for constructions containing a set of lemmas of nouns denoting familial relationships (e.g., *father*) preceded by a noun in the genitive case or followed by the preposition and a noun (*father of Hamlet*).

Syntactic parsing indicates syntactic relations between words within one sentence. Query-wise it can be considered a powerful enhancement of the morphological tagging, which abstracts from the word order – an invaluable asset when dealing with free-word order languages! With the syntactic parsing available, the aforementioned queries could be rephrased as adjectival attributes of a word with lemma matching a set of nouns denoting women and nominal and adjectival attributes of a word with lemma matching a set of nouns denoting familial relationships.

Corpus search relevant for CLS does not have to be limited to content extraction, as shows the seminal work of Douglas Biber on style or genre registers (Biber, 2004; Biber and Conrad, 2009)

Although corpora could certainly be searched with regular expressions, dedicated *query languages* make corpora search much more convenient and accessible for researchers without programming skills. Since the 1990, the dominant standard of linear corpus search (lemmas and morphological tags) has been the Corpus Query Language (CQL) working in the Corpus Query Processor, which was developed for the Corpus Workbench and is still maintained and regularly updated (Evert and CWB Development Team, 2022). CQL is a query language on simultaneously two different linguistic levels a) regular expressions on word occurrences and b) regular expressions on word property values. Many *corpus managers* nowadays use various implementations and flavors of the CQL query language.

The situation around the syntactic search is somewhat more complicated, as there does not seem to be one established standard.

This does not prevent some of the tools described in this report from using the GREW (graphs), TIGER Search (constituent graphs) or even CQL (dependency syntax) search engines to perform syntactic extractions. The syntactic description of sentences uses different graph formalisms, typically *oriented tree graphs* (hence the term *treebanks* for syntactically annotated

corpora). Querying such a complicated structure with regular expressions would not be suitable. Therefore, tree graphs are queried with *tree query languages*. Tree query languages are based on a range of algorithms to search such graphs, which affect how a query language would be designed. The most fundamental distinction goes between *declarative* and *procedural* queries. Declarative queries draw a template of the desired structure (a subtree), while procedural queries are small scripts traversing the tree graphs and testing conditions on their way through.

A tree graph consists of nodes (words) and edges (syntactic relations). The nodes have their own internal structure consisting of attributes and their values. The edges are *oriented*; that is, each edge points in a direction from its source node to a target node. Usually, the source node is the syntactically *governing* node (*parent*) and the target node is the syntactically *dependent* node (*child*). Each child can only have one parent. The syntactic relations must be defined in the documentation to the given corpus.

6. Analysis of Corpus Query Results

Most corpus managers allow for two types of output:

1. raw query matches
2. aggregations from the raw query results

Raw query matches. The raw query matches are strings matched by the query. They come surrounded by a context. Some corpus managers allow opening the whole source document, depending on how the corpus is licensed. Typically, context restrictions are determined in the number of surrounding characters, tokens, sentences, paragraphs, and other structural units. These matches can be used to build sub-corpora. In the most typical use case, the user views the matches on the screen or exports them to a plain text file or feeds them to additional modules or other tools to extract information from them, varying from very simple computations such as frequency lists, to highly complex analysis (such as topic modeling or stylometry). These analytical steps can result in numeric and tabular outputs, as well as in visualizations (diagrams).

The exact structure of the plain text file varies, but the most usual are comma-separated values files, along with the selected metadata for each match. Outputs of treebanks also come in a table-like format, with edges being rendered as reference to the ID of the governing node. The most common standard is currently CoNLL-U used by the Universal Dependencies (Figure 1). The CoNLL-U file is close to a comma-separated values file, with a sequence of tables interrupted by lines introduced by a hash (#), which is a common convention for comments in the code. Each table contains the linguistic markup of one sentence and each “comment” contains the text of that sentence along with additional metadata. The CoNLL-U format enables various user-friendly visualizations of the tree diagrams in a GUI (Figure 2).
Many corpus managers also have an API, so that they can be queried and return the output programmatically, often with several output format options (e.g., JSON).

Id	Form	Lemma	UPosTag	XPosTag	Feats	Head	DepRel	Deps	Misc
# generator = UDPipe 2, https://lindat.mff.cuni.cz/services/udpipe									
# udpipe_model = english-ewt-ud-2.10-220711									
# udpipe_model_licence = CC BY-NC-SA									
# newdoc									
# newpar									
# sent_id = 1									
# text = Corpus search is easy.									
1	Corpus	corpus	NOUN	NNP	Number=Sing	2	compound	_	TokenRange=0:6
2	search	search	NOUN	NN	Number=Sing	4	nsubj	_	TokenRange=7:13
3	is	be	AUX	VBZ	Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin	4	cop	_	TokenRange=14:16
4	easy	easy	ADJ	JJ	Degree=Pos	0	root	_	SpaceAfter=No|TokenRange=17:21
5	.	.	PUNCT	.	_	4	punct	_	TokenRange=21:22

Figure 1: A sentence annotated with the Universal Dependencies, in the CoNLL-U format

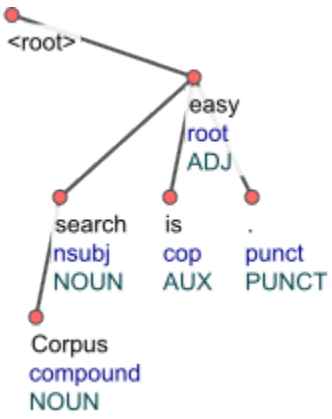

Figure 2: A tree diagram in Universal Dependencies, from the CoNLL-U format

Aggregations from the raw query results. Exported raw query results can certainly be further processed with other tools, depending on the research task. Also, corpus managers themselves offer built-in analytical methods, such as frequency counts, computations of diverse collocation

scores, distributions across different metadata categories (e.g., genre), and sometimes even simple plotting. Again, the aggregated results should be possible to download, typically in a tabular format. The range of available analytical methods greatly varies. When selecting the corpus manager, its built-in analytical methods should not be more important a criterion than the expressivity of its query language, its robustness with large data, API access, and the export options.

7. Domain adaptation of Language Models for CLS

When linguistic-markup tools are missing or underperforming, the automatic processing of the affected language is impossible, with all implications for the education, research, industry, and culture of that language community. This danger, expressively called *digital extinction,* has been long known. For most European languages, it was comprehensively examined by the META-NET Language White Paper series (e.g. Hernáez et al., 2012).

Since 2012, many more languages have built enough linguistic resources and trained NLP tools performing well enough to protect them from the digital extinction in the everyday communication. However, most NLP tools have been trained on texts from newswires, social networks and web, and Wikipedia. The resulting language models are heavily biased towards these modern text domains, and therefore their performance on literary data often turns out much lower than declared.

It is one of the major challenges for the CLS community to assess the performance of the current language models and create reliable language resources for the underperforming domains: archaic language varieties and dialects, spontaneous speech transcripts, poetry, drama, and their combinations. So far, no CLS-specific support infrastructure has been established; only some scattered efforts have been going on: CLS INFRA has hosted two scholars on 3-month fellowships to create CLS-relevant linguistic resources for NLP tools and is ready to host more until the end of the project in March 2025. DARIAH has established a Working Group dedicated to Multilingual DH[3], but at the time of writing this technical report it has not started its activities yet. At the moment, scholars are welcome to directly join the Universal Dependencies community, where they will be instructed how to build their annotated corpus[4].

Twice a year, the UD team releases an update of the UD treebanks and the developers of the UDPipe parser[5] use the update release to retrain their models. This is how new models get included.

The minimum size for a corpus to be included in the Universal Dependencies treebank family on GitHub is more than 20 sentences and more than 100 words at the same time, but it is not enough to train a model on. For instance, the corpus of a new language should comprise of at least 20,000 words to be eligible for the regular semi-annual model training round with the

[3] https://www.dariah.eu/activities/working-groups/multilingual-dh/
[4] https://universaldependencies.org/release_checklist.html
[5] https://lindat.mff.cuni.cz/services/udpipe/

UDPipe parser team. However, models of smaller corpora around 5,000 tokens can also be included on request and after an assessment, especially when large corpora of the same language already exist.

8. The Logic and Organization of the NLP tools list for CLS

8.1 Key concepts

In an effort to contribute to a list of applicable NLP tools for the CLS scholar, a standard ontology has been developed for the characteristics of these tools. The tool list is conceptualized in accordance with the data model proposed in D6.1. The D6.1 data model is conceived from the perspective of the data: *Tools* use *methods* to add *features* to the *data*. This technical report pivots on the opposite perspective of *tools* that provide the *data* with *features* or filter the data matching a feature condition.

Especially with annotation, these pairs are very straightforward: methods and features map 1:1 and are denoted by the same or derived term, e. g. *part-of-speech tagging* (method) and *part-of-speech tags* (feature). Things get more complicated with tools that perform computations over the (possibly annotated) data. For instance, a stylometric tool such as *stylo* applies the method of computing cosine distances between all texts in a corpus along with a clustering algorithm to present a list of all text pairs, their distances, and their association with a given cluster, which can come as a table as well as a plot. Both the table and the plot are features of the text, although the text does not contain them.

The tools perspective introduces one additional concept – *task* – and tentatively describes the outcome of the method with the terms *features*, *metric*, *formalism* and *tagset*. In this tentative data model, the researcher selects a tool to solve a task. That tool approaches this task with methods to provide some features or compute values of some metrics for the given text. Features serve to enrich the input text with annotation (either in-line or stand-off) or filter the text according to a query. Assigned labels typically belong to a given tagset (e. g. Universal Dependencies or PennTreebank) or/and pertain to a formalism (such as HPSG or minimalist syntax). Filters typically pertain to a query language, which also is a formalism.

Another important feature is formats and schemata of the data on input as well as on output (cf. D6.1, Section 5.5.5). The formats and schemas of the known corpora are extremely diverse. It is well known that one of the most established formats, TEI-XML, is so flexible that it does not ensure full interoperability of different data sets. Hence, a really helpful description of a corpus format is solely a full formal definition (such as a DTD) to which its document pertains. This definition cannot be further summarized and requires good XML proficiency from the reader. D 6.1 has so far left this issue open. Nevertheless, this does not prevent some of the tools described in this technical report from being specialized in the adaptation to many common flavors of XML-TEI encoding to be able to process the texts. The makeshift approach of this technical report is giving the reader at least a raw estimate whether a tool can work with any TEI-XML at all, based on several test cases, and a basic presentation of the tool's assumptions

about the data format and structure beyond TEI-XML. Besides, most tool entries provide a narrative section about how to use a given tool and what for.

D 6.1 also mentions the issue of timespan, time issues, temporal coverage (D.6.1, Section 5.5.7). This issue is also relevant to tools, since languages are constantly evolving and tools relying on a language model perform the worse the more the data divert from what the tools were trained on.

8.2 Scope

This technical report focuses on providing information on corpus management, data annotation and feature extraction within a future tool chain. It deliberately does not detail a number of very important tools that come in the workflow before and after. Many of these tools have been reflected upon in D3.1, thus please see that report for a list of all digital tools, regardless of the scope. For example, tools such as Transkribus[6] or the fully open-source Kraken[7], the OCR and HTR software, would be an example of a most prominent tool used before corpus management in the workflow to generate machine readable text from a manuscript. In addition, this report does not consider how the output of information extraction can be used with numerous model-building tasks, such as authorship attribution (*stylo* Eder et al., 2016) or data modeling (*mallet* - McCallum, 2002) or great general-purpose visualization and reporting software such as Gephi (Bastian et al., 2009), not to speak about R and Python data science libraries.

[6] https://readcoop.eu/transkribus/?sc=Transkribus

[7] https://github.com/mittagessen/kraken

9. Included Tools in the Alphabetical Order of Their Primary Application

Tool name	Primary purpose
Arborator Grew	Annotation
conll-u editor	Annotation
INCEpTION	Annotation
DraCor	Corpus management
IMS Corpus Workbench	Corpus management
Kontext	Corpus management
SimpleCorp	Corpus management
TEITOK	Corpus management
TXM desktop	Corpus management
TXM portal	Corpus management
Calc	NLP
Flair	NLP
NameTag	NLP
QuitaUp	NLP
spaCy	NLP
UDapi	NLP
udpipe R library	NLP
UDPipe1 parser	NLP
UDPipe2 parser	NLP
Universal Dependencies Models for UDPipe	NLP
Alberti	poetry processing
Alberti-stanzas	poetry processing
Averell	poetry processing
Horace	poetry processing
Poetrylab	poetry processing
Poetrylab-API	poetry processing
rhymetagger	poetry processing

ANNOTATION TOOLS

Arborator Grew

Quick description	NLP
Task it solves	Collaborative dependency annotation
Method it uses for that task	query language + writing functions
Features (text enrichment)	syntactic dependencies, morphological features
Metric	
Formalism	Grew syntax
Tagset	

What can this tool do for you?

Arborator Grew is an interactive and user-friendly on-line tool for annotating treebanks. It provides annotation management over several users with different editing rights. It can also be used in the classroom. The teacher can provide correct solutions and make them visible for the students to a varying extent.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Any (it is language-agnostic)	

Technical details

Version	unknown
Version Date:	unknown
Works on Operating Systems:	all
License:	Fully free
Distribution:	web service, local server with client integrated in the web browser
Download, installation, documentation, user guides:	https://github.com/Arborator https://arborator.github.io/arborator-documentation/#/
User interface:	web browser
Docker instance:	No or not known.
How does the tool process your text:	It displays
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	This tool does not support XML-TEI at all.

Required data input format:	CoNLL-U

Related papers

Guibon, G., Courtin, M., Gerdes, K., & Guillaume, B. (2020). When Collaborative Treebank Curation Meets Graph Grammars. Proceedings of The 12th Language Resources and Evaluation Conference, 5293–5302. https://www.aclweb.org/anthology/2020.lrec-1.651

conll-u editor

Quick description	Annotation
Task it solves	manually provide labels to text
Method it uses for that task	manual annotation
Features (text enrichment)	edit tokenization, sentence splitting, word form, lemma, part of speech, morphological features, and syntactic relations; validate your annotation against a scheme;
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies

What can this tool do for you?

This tool allows you to manually create or edit corpora annotated with Universal Dependencies. It provides a particularly user-friendly tokenization editing; that is, it is very easy to delete, split or merge tokens or sentences – a feature that is not automatically provided with annotation tools! Also, the sentences can be visualized as trees or flat graphs or tables.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Any (it is language-agnostic)	

Technical details

Version	V2.20.0
Version Date:	2023-01-22
Works on Operating Systems:	the server runs on Linux, but the annotation runs in web browsers across operating systems.
License:	Fully free (BSD 3-Clause License)
Distribution:	local client-server system
Download, installation, documentation, user guides:	https://github.com/Orange-OpenSource/conllueditor
User interface:	web browser frontend
Docker instance:	Yes
How does the tool process your text:	It loads a CoNLL-U file and allows you to edit it.
Tool exports results:	Yes.
Statistical models:	This tool is not a statistical model a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	This tool does not support XML-TEI at all.

Required data input format:	CoNLL-U

Related papers

Heinecke, J. (2019). ConlluEditor: A fully graphical editor for Universal Dependencies treebank files. Universal Dependencies Workshop 2019. Paris. https://syntaxfest.github.io/syntaxfest19/proceedings/papers/paper_55.pdf

INCEpTION

Quick description	Annotation
Task it solves	manual annotation
Method it uses for that task	
Features (text enrichment)	labels of different kinds, some pre-defined (e. g. Universal Dependencies)
Metric	
Formalism	user dependent
Tagset	user dependent

What can this tool do for you?

This tool lets you to manually add markup to your documents. It can even learn from your supplied annotation and pre-annotate. Mind that the performance depends on data size and annotation complexity (e. g. the number of labels). However, it is a high-quality tool even without considering the pre-annotation option at all.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Any (it is language-agnostic)	

Technical details

Version	27.0
Version Date:	2023-03-07
Works on Operating Systems:	Linux, MacOs, Windows
License:	Apache 2.0
Distribution:	.jar file, local server installation with client opening in a browser window (Chrome, Safari, Firefox)
Download, installation, documentation, user guides:	https://inception-project.github.io https://inception-project.github.io/releases/27.0/docs/admin-guide.html https://fortext.net/tools/tools/inception (in German)
User interface:	client in the web browser
Docker instance:	No or not known
How does the tool process your text:	It reads in your tool, allows adding markup according to templates (defined by you or pre-defined) and stores it.
Tool exports results:	Yes (although issues may arise when it should preserve some previously added markup)
Statistical models:	This tool is (not) a statistical model / a set of statistical models, but it does include statistical models.
Plug in your own model:	You can easily train your statistical model to plug in this tool.

XML-TEI compatibility:	Yes
Required data input format:	This tool accepts many formats. See https://inception-project.github.io/releases/27.0/docs/user-guide.html#sect_formats

Related papers

Klie, J.-C., Bugert, M., Boullosa, B., de Castilho, R. E., & Gurevych, I. (2018). The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations, 5–9. http://tubiblio.ulb.tu-darmstadt.de/106270/

CORPUS MANAGEMENT TOOLS

DraCor

Quick description	Corpus manager -Textometry
Task it solves	Give summary statistics of a corpus
Method it uses for that task	
Features (text enrichment)	number of plays, number of characters, number of tokens in speeches and stage directions
Metric	
Formalism	
Tagset	
Quick description	Information extraction
Task it solves	Retrieve structural units and characters of a drama
Method it uses for that task	Drama analysis
Features (text enrichment)	characters, segments (acts, scenes, etc.)
Metric	
Formalism	
Tagset	
Quick description	Information extraction
Task it solves	Extract a character network from a drama; calculate network metrics
Method it uses for that task	Network analysis
Features (text enrichment)	Network graph based on co-occurring characters in scenes
Metric	
Formalism	
Tagset	
Quick description	Information extraction
Task it solves	Retrieve parts of the text
Method it uses for that task	Drama analysis
Features (text enrichment)	spoken text by character/gender, stage directions
Metric	
Formalism	
Tagset	

What can this tool do for you?

DraCor simplifies accessing specific parts of a TEI annotated dramatic texts (such as the spoken text per character; spoken text of characters by gender, stage directions, etc.) DraCor automatically extracts textual features (e. g. number of characters, speeches, scenes, etc.), generates networks based on co-occurrences of characters in a scene, and calculates network

metrics. The derived data can be exported in various formats, e.g., JSON, CSV, RDF but also specialized formats, e.g., GEXF format to be used in Gephi, a tool for network analysis. The DraCor API offers numerous options and additional formats for further processing of the unified metadata. A local instance of the DraCor system can be set up using Docker, which can be used to work with custom corpora or in copyright material.

Which languages can it work with (as of February 2023)?

The tool itself is **language-agnostic**. The table below lists languages currently represented in the corpora.

Language	Variety (geographical, or temporal if not modern)
Czech	
English	
French	
German	
Greek	Ancient Greek
Hungarian	
Italian	
Latin	
Russian	
Spanish	
Swedish	
Ukrainian	
Tatar	
Bashkir	
Alsatian	

Technical details

Version	0.87.1
Version Date:	2022-12-30
Works on Operating Systems:	Windows, Unix, MacOs, Linux
License:	Fully free
Distribution:	Cloud service
Download, installation, documentation, user guides:	https://github.com/dracor-org/dracor-api (see README.md)
User interface:	GUI, API-REST
Docker instance:	Yes
How does the tool process your text:	• Returns some metrics as numbers or tables. • Returns some visualization (plots, interactive elements, etc.).
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.

XML-TEI compatibility:	• The tool requires text in XML-TEI, adds its markup in XML-TEI, and ensures valid XML-TEI on the output.
Required data input format:	Should conform to DraCor Schema (see https://github.com/dracor-org/dracor-schema/tree/main/odd)

Recommended tutorials

Title	Provider/Author	URL
DraCor Notebooks	DraCor	https://github.com/dracor-org/dracor-notebooks
DraCor: Drama Corpora Project [in. German]	Jan Hostmann	https://fortext.net/ressourcen/textsammlungen/dracor-drama-corpora-project

Related papers

Börner, I., & Trilcke, P. (2023). *CLS INFRA D7.1 On Programmable Corpora*. https://doi.org/10.5281/zenodo.7664964

Fischer, F., Börner, I., Göbel, M., Hechtl, A., Kittel, C., Milling, C., & Trilcke, P. (2019, Juli 10). Programmable Corpora: Introducing DraCor, an Infrastructure for the Research on European Drama. *DH2019: »Complexities«. 9–12 July 2019. Book of Abstracts*. DH2019 »Complexities«, Utrecht. https://doi.org/10.5281/ZENODO.4284002

Trilcke, P., Ustinova, E., Börner, I., Fischer, F., & Milling, C. (2022, September 14). *Detecting Small Worlds in a Corpus of Thousands of Theatre Plays. A DraCor Study in Comparative Literary Network Analysis [Conference Version]*. Workshop on Computational Drama Analysis: Achievements and Opportunities, Cologne. https://github.com/dracor-org/small-world-paper/raw/conference-version/Detecting_Small_World_Networks__in_a_Huge_Multilingual_Corpus_of_Theater_Plays.pdf

IMS Corpus Workbench

alias CWB

Quick description	Corpus manager
Task it solves	Search in a text corpus
Method it uses for that task	Linear text search with Corpus Query Processor
Features (text enrichment)	Text snippets matching a query; supports also displaying the syntactic trees
Metric	--
Formalism	**CQL**
Tagset	--

What can this tool do for you?

The IMS Open Corpus Workbench is a tool for managing and querying text corpora. It is a prequel to other big corpus managers such as KonText, where the main difference is that CWB is just the server part (backend) that can be supplemented with a web-based GUI like CQPweb.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Any (it is language-agnostic)	

Technical details

Version	3.5
Version Date:	Unknown
Works on Operating Systems:	Linux, MacOS, Windows
License:	Fully free
Distribution:	Local server-client installation
Download, installation, documentation, user guides:	CWB is developed at https://sourceforge.net/projects/cwb/ Documentation: https://cwb.sourceforge.io/documentation.php
User interface:	GUI, API
Docker instance:	No
How does the tool process your text:	Allows you to search through your texts and returns matching results.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	The input text has to be transformed into a vertical, CoNLL-U-like format.

Related papers

Evert, S., & Hardie, A. (2011). Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium. *Proceedings of the Corpus Linguistics 2011 Conference*. Corpus Linguistics, Birmingham, UK. http://www.birmingham.ac.uk/documents/college-artslaw/corpus/conference-archives/2011/Paper-153.pdf

KonText

Quick description	Corpus manager
Task it solves	Search in a text corpus
Method it uses for that task	Linear text search with Corpus Query Processor
Features (text enrichment)	Text snippets matching a query; supports also displaying the syntactic trees
Metric	
Formalism	CQL (Corpus Query Language)
Tagset	
Quick description	**Corpus manager**
Task it solves	Basic statistics: frequency breakdown according to metadata, which words typically co-occur with a given word etc.
Method it uses for that task	Frequency distribution, Collocations analysis
Features (text enrichment)	
Metric	Statistics expressing the mutual attraction of two words or short text segments (MI, Log-Likelihood Ratio)
Formalism	
Tagset	

What can this tool do for you?

KonText is an advanced and customizable corpus query engine built above the Manatee library of NoSketch Engine. KonText is intended for professional use and deployment in organizations with many users, rather than for home use. Its main features are: full support for parallel corpora, partial support for syntactically annotated and spoken corpora. Manatee is a token-based backend that is fast and scalable, but its support for corpora with complex annotation (organized in multiple interfering layers) is limited.

Which languages can it work with (as of February 2023)?

The tool is language-agnostic (i.e., any language).

Technical details

Version	0.17
Version Date:	February 2023
Works on Operating Systems:	Linux
License:	Fully free (GPL 2.0)
Distribution:	Local server-client installation

Download, installation, documentation, user guides:	• KonText is developed openly at https://github.com/czcorpus/kontext/; this includes an installation manual. • KonText is deployed at several CLARIN centres in Europe, the canonical installation can be tested at https://www.korpus.cz/kontext.
User interface:	GUI, API
Docker instance:	Yes
How does the tool process your text:	• Allows you to search through your texts and returns matching results. • Returns some visualization (plots, interactive elements, etc.).
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	The input text has to be transformed into a vertical, CoNLL-U-like format.

Related papers

Machálek, T. (2020). KonText: Advanced and Flexible Corpus Query Interface. *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 7003–7008. https://www.aclweb.org/anthology/2020.lrec-1.865

SimpleCorp

Quick description	Corpus Manager, NLP
Task it solves	creating and searching an annotated corpus, preserving XML-TEI markup in documents
Method it uses for that task	tokenization, lemmatization, morphological tagging and syntactic parsing, indexing for CQL search.
Features (text enrichment)	token boundary tags, lemma, part of speech, syntactic labels
Metric	
Formalism	CQL, Universal Dependencies
Tagset	Universal Dependencies

What can this tool do for you?

SimpleCorp is primarily intended as a teaching tool for NLP courses. It allows students (or researchers) to create a corpus from a collection of documents, as long as they have an SSO login. And it tries to make that process as straightforward as possible. Upon upload, the tool converts the documents to TEI/XML, and then applies a pre-defined NLP pipeline. After uploading a collection of files, it will compile a searchable corpus in CWB from the collection of XML files.

Which languages can it work with (as of February 2023)?

The tool can work with any language supported by the UDPipe REST service.

Technical details

Version	0.6
Version Date:	2023-01-15
Works on Operating Systems:	Linux, MacOs
License:	to be determined
Distribution:	Server-based tool
Download, installation, documentation, user guides:	https://lindat.mff.cuni.cz/services/simplecorp/
User interface:	Web-based
Docker instance:	There currently is none, but once out of beta, we intend to create a docker instance
How does the tool process your text:	• Drop-and-drop files • Automatic conversion to TEI/XML (TEITOK style) • Automatic application of NLP pipeline • Automatic creation of a searchable corpus
Tool exports results:	Yes

Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.
Plug in your own model:	SimpleCorp is a streamlined, but restricted instance of TEITOK, which does not itself allow any training, but only applies pre-defined NLP tools (currently UDPipe2). But the project itself can be exported to TEITOK, where tools can be trained.
XML-TEI compatibility:	The whole tool is TEI/XML based, with minimal deviation from the standard to make it possible to edit and process/display, but all such deviation can be converted to standard TEI in the export.
Required data input format:	The drag-and-drop module converts from a number of different format, so that ideally no pre-structuring is necessary. But all internal files are TEI/MXL.

Which other tools from this list does this tool integrate?

- UDPipe
- IMS Corpus WorkBench
- TEITOK

Related papers

Arrabal Rodríguez, P. 2022. TEITOK, a visual solution for XML/TEI encoding: editing, annotating and hosting linguistic corpora. RIDE – A review journal for digital editions and resources

Janssen, M. 2021. A Corpus with Wavesurfer and TEI: Speech and Video in TEITOK. International Conference on Text, Speech, and Dialogue. LNAI 12848: 261–268

Janssen, M. 2021. Integrating TEITOK and Kontext/PMLTQ at LINDAT. Selected papers from the CLARIN Annual Conference 2020. Linköping Electronic Conference Proceedings 180: 180 104–110.104

Janssen, M. 2018. TEITOK as a tool for Dependency Grammar. In: Procesamiento del Lenguaje Natural, vol. 61.

Janssen, M. 2018. Adding Words to Manuscripts: From PagesXML to TEITOK. In: Méndez E., Crestani F., Ribeiro C., David G., Lopes J. (eds) Digital Libraries for Open Knowledge. TPDL 2018. Lecture Notes in Computer Science, vol 11057. Springer, Cham.

Janssen, M. and Vaamonde, G. 2017. From traditional historical corpora to TEI-based corpora. Benefits and challenges in joining digital edition and corpus annotation. In: Calen barbas, falen cartas. A escrita en galego na Idade Moderna. Santiago de Compostela.

Janssen, M. 2016. TEITOK: Text-Faithful Annotated Corpora. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia.

Janssen, M. 2015. Multi-level manuscript transcription: TEITOK. Congresso de Humanidades Digitais em Portugal, Lisboa. Portugal

TEITOK

Quick description	**NLP**
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Tokens; Sentence borders
Metric	
Formalism	
Tagset	
Quick description	**Corpus manager**
Task it solves	Search in a text corpus
Method it uses for that task	Linear text search with Corpus Query Language and its flavors
Features (text enrichment)	Text snippets matching a query
Metric	
Formalism	**CQL**
Tagset	
Quick description	**Corpus manager**
Task it solves	Search in a parsed text corpus (treebank)
Method it uses for that task	
Features (text enrichment)	Syntactic trees matching a query.
Metric	
Formalism	PMLTQ, Grew
Tagset	
	Corpus manager
Task it solves	Tell which words typically co-occur with a given word
Method it uses for that task	Collocations analysis
Features (text enrichment)	
Metric	Statistics expressing the mutual attraction of two words or short text segments (MI, Log-Likelihood Ratio)
Formalism	
Tagset	
	Corpus manager - Textometry
Task it solves	Give summary statistics of a corpus or selected texts
Method it uses for that task	Various formulas
Features (text enrichment)	

Metric	Token frequencies, Lexical richness/diversity; Key words, Thematic concentration, Descriptivity/activity, Verb distance, Entropy, Hapaxes
Formalism	
Tagset	
Quick description	**Annotation tool**
Task it solves	Manually annotate your texts, e. g. with named entities
Method it uses for that task	Manual text annotation/annotation editing
Features (text enrichment)	Labeled text spans or tokens or their mutual relations
Metric	
Formalism	
Tagset	
Quick description	**Information extraction**
Task it solves	Relational extraction from metadata (DraCor etc.)
Method it uses for that task	Network analysis
Features (text enrichment)	Network graph with characters as nodes and interactions as edges;
Metric	Degree centrality, Eigenvector centrality, Weight
Formalism	
Tagset	

What can this tool do for you?

TEITOK can be used to create corpora and keep a wide range of different types of information (facsimile alignment, audio alignment, dependency relations, pos tagging, geolocation information, IGT, morphological data, parallel alignment, witness alignment). It then provides various interfaces to exploit all these types of information, and in most cases provides an editing environment as well to provide automatic annotations and manually correct or create annotations. It also provides searches through the corpus, either using CQL for simple annotated files, or Grew or PML-TQ for dependency parsed corpora. It also provides specific search functionality for parallel corpora (be it based on translation or on witnesses) as well as for spoken and facsimile aligned data.

Which languages can it work with (as of February 2023)?

The tool is fully language independent, and has support for right-to-left languages, as well as non-alphabetic scripts. TEITOK has been often used for NLP on left-to-right languages such as Ladino or Sardinian.

Technical details

Version	3.2
Version Date:	2023-01-15
Works on Operating Systems:	Linux, MacOs
License:	Fully free (GNU GPLv3)
Distribution:	Local client-server installation
Download, installation, documentation, user guides:	• https://gitlab.com/maartenes/TEITOK • Project URL: http://www.teitok.org/ • Help Pages: http://www.teitok.org/index.php?action=help
User interface:	• Web GUI • REST-API
Docker instance:	Yes – a docker install script is provided in the Git
How does the tool process your text:	• Convert from several input formats • Run annotation tools on the server for automatic annotation • Manually correct or annotate in the GUI • Use the API to run processing or annotation tools remotely.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.
Plug in your own model:	There is a native POS tagger (NeoTag) that can be easily trained in the GUI on the manually corrected data. But NeoTag is gradually being replaced by UDPIPE, which currently can only be trained in the GUI using the older 1.0 version
XML-TEI compatibility:	The whole tool is TEI/XML based, with minimal deviation from the standard to make it possible to edit and process/display, but all such deviation can be converted to standard TEI in the export.
Required data input format:	The input should ideally be in TEI/XML, but there are various conversion tools that convert from a collection of OCR, Spoken, and annotated formats. But all internal files are TEI/MXL

Which other tools from this list does this tool integrate?

- UDPipe
- IMS Corpus WorkBench
- DraCor
- NameTag
- QuitaUp (under development)
- INCePTION
- GrewMatch

Related papers

Arrabal Rodríguez, P. 2022. TEITOK, a visual solution for XML/TEI encoding: editing, annotating and hosting linguistic corpora. RIDE – A review journal for digital editions and resources

Janssen, M. 2021. A Corpus with Wavesurfer and TEI: Speech and Video in TEITOK. International Conference on Text, Speech, and Dialogue. LNAI 12848: 261–268

Janssen, M. 2021. Integrating TEITOK and Kontext/PMLTQ at LINDAT. Selected papers from the CLARIN Annual Conference 2020. Linköping Electronic Conference Proceedings 180: 180 104–110.104

Janssen, M. 2018. TEITOK as a tool for Dependency Grammar. In: Procesamiento del Lenguaje Natural, vol. 61.

Janssen, M. 2018. Adding Words to Manuscripts: From PagesXML to TEITOK. In: Méndez E., Crestani F., Ribeiro C., David G., Lopes J. (eds) Digital Libraries for Open Knowledge. TPDL 2018. Lecture Notes in Computer Science, vol 11057. Springer, Cham.

Janssen, M. and Vaamonde, G. 2017. From traditional historical corpora to TEI-based corpora. Benefits and challenges in joining digital edition and corpus annotation. In: Calen barbas, falen cartas. A escrita en galego na Idade Moderna. Santiago de Compostela.

Janssen, M. 2016. TEITOK: Text-Faithful Annotated Corpora. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia.

Janssen, M. 2015. Multi-level manuscript transcription: TEITOK. Congresso de Humanidades Digitais em Portugal, Lisboa. Portugal

TXM desktop

Quick description	NLP
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Tokens; Sentence borders
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Give the basic dictionary form of a token
Method it uses for that task	Lemmatization
Features (text enrichment)	Lemmas, aka dictionary forms of tokens
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Determine part of speech and relevant morphological categories
Method it uses for that task	Morphological tagging
Features (text enrichment)	Parts of speech; Morphological categories
Metric	
Formalism	
Tagset	TreeTagger
Quick description	Corpus management
Task it solves	Build a subcorpus
Method it uses for that task	Build a corpus configuration by selecting a set of texts or a set of text parts Corpus configuration
Features (text enrichment)	Selection of texts by a combination of their properties; selection of text structures by a combination of their properties
Metric	
Formalism	CQL
Tagset	
Quick description	Corpus management
Task it solves	Build a partition
Method it uses for that task	Build a contrastive corpus configuration by selecting a set of texts or a set of text parts Corpus configuration
Features (text enrichment)	selection of texts by a combination of their properties; selection of text structures by a combination of their properties ; the sum of selections is always the whole corpus (resp.) the whole text

Metric	
Formalism	
Tagset	CQL
Quick description	**Information extraction**
Task it solves	Search in a text corpus
Method it uses for that task	
Features (text enrichment)	Selection of texts by a combination of their properties ; selection of text structures by a combination of their properties ; the sum of selections is always the whole corpus (resp.) the whole text
Metric	
Formalism	CQL
Tagset	
Quick description	**Information extraction**
Task it solves	Search in a text corpus
Method it uses for that task	Linear text search with Corpus Query Processor Quantification
Features (text enrichment)	Lexical pattern observables extractions; Text snippets matching a query ; Word lists matching a query
Metric	
Formalism	CQL
Tagset	
Quick description	**Information extraction**
Task it solves	Search in a parsed text corpus (treebank)
Method it uses for that task	Syntactic tree search Quantification
Features (text enrichment)	Syntactic trees matching a query ; Text snippets matching a query ; Word lists matching a query ; Lexical pattern observables extractions
Metric	
Formalism	TIGER Search, CQL
Tagset	
Quick description	**Information extraction**
Task it solves	Search in an annotated text corpus (co-reference)
Method it uses for that task	Co-reference chain search Quantification
Features (text enrichment)	Co-reference chains Units, Relations or Schema matching a query
Metric	
Formalism	URSQL
Tagset	
Quick description	**Corpus analysis**
Task it solves	Tabulate the contexts of occurrences of word patterns
Method it uses for that task	Concordance analysis Qualitative analysis

Features (text enrichment)	Tabulate word pattern occurrences with reference, left context, occurrence and right context columns ; Bring similar contexts together by multiple sort on any column ; Link to the text edition (to read) with occurrence highligthed ; Link to audio/video stream (to play) synchronized with occurrence
Metric	
Formalism	URSQL
Tagset	
Quick description	**Corpus analysis**
Task it solves	Show word patterns occurrences density within the corpus
Method it uses for that task	Progression analysis Analytical views production and interaction Syntagmatic analysis Qualitative analysis
Features (text enrichment)	Visualisation (interactive graphic) showing a density curve for each word pattern searched ; link to concordances
Metric	
Formalism	CQL, TIGER Search, URSQL
Tagset	
Quick description	Corpus Analysis
Task it solves	Tabulate word properties of occurrences of word patterns
Method it uses for that task	Hierarchical Index analysis Paradigmatic analysis Quantitative analysis
Features (text enrichment)	Tabulate word pattern properties values (eg combine word form, pos or lemma) of occurrences with their frequency
Metric	
Formalism	CQL, TIGER Search, URSQL
Tagset	
Quick description	Corpus Analysis
Task it solves	Tell which words typically co-occur with a given word
Method it uses for that task	Collocations analysis Syntagmatico-Paradigmatic analysis Quantitative analysis
Features (text enrichment)	
Metric	Statistics expressing the mutual attraction of two words or short text segments (specificity statistical measure)
Formalism	**CQL**
Tagset	
Quick description	**Corpus Analysis**
Task it solves	Tell which words typically occur within a sub-corpus or inside parts of a partition
Method it uses for that task	Keywords analysis of a partition or subcorpus Paradigmatic analysis Quantitative analysis

Features (text enrichment)	
Metric	Statistics expressing the attraction of a word or a short text segment with a sub-corpus of the whole corpus or with parts of a partition of the corpus (specificity statistical measure) Visualisation: Bar plot showing statistic value for each word pattern searched
Formalism	**CQL**
Tagset	
Quick description	**Corpus Analysis**
Task it solves	Show similarities between texts or structures based on word pattern frequencies
Method it uses for that task	Correspondence factor analysis of a partition Analytical views production and interaction Paradigmatic analysis Quantitative analysis
Features (text enrichment)	factorial coordinates, contributions, quality of representation Visualisation: factorial planes
Metric	chi-square
Formalism	**CQL**
Tagset	
Quick description	**Corpus Analysis**
Task it solves	Show hierarchy of similarities between texts or structures based on word pattern frequencies
Method it uses for that task	Hierarchical Cluster analysis of a partition Analytical views production and interaction Paradigmatic analysis Quantitative analysis
Features (text enrichment)	
Metric	chi-square; clusters clusters; hierarchy, factorial planes
Formalism	
Tagset	
Quick description	**Corpus Analysis**
Task it solves	Give summary statistics of a corpus or selected texts
Method it uses for that task	Various statistics
Features (text enrichment)	Token properties (form, pos, lemma) list and frequencies ; Structures properties list and frequencies
Metric	
Formalism	
Tagset	
Quick description	**Annotation tool**
Task it solves	Manually annotate your tokens, e. g. with pos
Method it uses for that task	Manual token annotation/annotation editing Annotation through concordance Qualitative analysis

Features (text enrichment)	Labeled token spans or tokens
Metric	
Formalism	CQL
Tagset	
Quick description	**Annotation tool**
Task it solves	Semi-automatically annotate your tokens, e. g. with pos
Method it uses for that task	Semi-automatical token annotation/annotation editing Annotation through scripts Qualitative analysis
Features (text enrichment)	Labeled token spans or tokens
Metric	
Formalism	Unit-Relation-Schema (URS)
Tagset	
Metric	
Formalism	
Tagset	

What can this tool do for you?

TXM can be used to create and analyze four main types of textual corpora: written texts (possibly aligned with fac-simile), speech transcriptions (possibly synchronized with media files at word level), multilingual/parallel corpora and corpora encoded in a table (e.g. answers to a survey, tweets). TXM can automatically annotate texts with part of speech and lemma with the help of TreeTagger (or other NLP tools) or use linguistic annotations already encoded into the sources.

One specificity of TXM is that it can process progressively from plain text to richly structured XML-TEI encoded text. It offers a continuous range of import modules covering the most frequently used standard formats:

- TXT: for any plain text coming from word processors, PDFs, websites, etc.;
- XML: for slightly structured texts (only sentences or paragraphs for example) or linguistically enriched (with XML tags that encode certain words with lexical properties);
- TEI: for texts encoded according to the recommendations of the TEI consortium and which are intended to be capitalized for long-term projects, shared with other initiatives or compatible with archiving systems.

A project can apply TXM to its encoded data progressively, from the simplest way (and most limited to use) to the most complex way (and richest). TXM therefore makes it possible to adapt the costs of corpus encoding according to the real needs of the study, especially when these needs are discovered as the corpus is being analyzed. Under these conditions, TXM assists both the encoding activity and the exploitation of corpora.

When using TEI, TXM helps to build and host rich HTML 5 + CSS 3 + Javascript based text editions, with possible fac-simile synoptic views (page image links are simply encoded in <pb/> tags).

TXM supports various flavors of TEI P4/P5 encoding practices: Perseus, TextGrid, Base de Français Médiéval (BFM), BVH Epistemon, etc. TEI sources are preprocessed by several XSL stylesheets that are delivered with TXM. Other stylesheets are available in the TXM XSL stylesheets library online: http://sourceforge.net/projects/txm/files/library/xsl.

TXM provides four types of tools to analyze a corpus:

- 1) Corpus configuration tools to build specific subcorpora or corpus partitions, based on sets of texts or sets of text structures (inside a text)
- 2) Semi-manual annotation tools at word or word sequence level: through concordance or text reading
- 3) Qualitative analytic tools like word pattern lists, concordances or progression views
- 4) Quantitative analytic tools like word pattern cooccurrents analysis, keyword analysis, and contrastive tools like correspondence factorial analysis or cluster analysis

Every analytic tool uses word patterns expressed with search queries on annotations with the help of different search engines (Corpus Query Processor, TIGER Search or URSQL). For example, word patterns expressed with the CQL query language:

- "aiming": to simply search for the word 'aiming'
- [pos="VERB" & word=".*ing"]: to search for verb forms ending in ".ing" (where Part of Speech annotation is present)
- [lemma="group"] []{0,3} [pos="VERB" & word=".*ing"]: to search for the collocation <group lemma> followed by a <verb with progressive aspect> with at most 3 words in between

TXM makes an analysis working session more fluid by interconnecting the tools as much as possible through hypertextual links located in their result views. For example: a double-click from a density curve of a word pattern in a text (general view of Progression) to the Concordance of this pattern (focused contextual reading), then from a line of this concordance to an Edition page where the occurrence is highlighted (precise reading of the text), then from this page to the Playback of the video corresponding to the transcription at the time the occurrence is pronounced (verification in the primary source).

All statistical models used by the quantitative tools are implemented and documented in R.

Results can be exported as spreadsheets (for data) or images (for visualisations). Texts and annotations can be exported in XML-TEI format. Corpora can be exported as a .txm file to be

loaded in another TXM or uploaded to a TXM portal server version for online access and analysis.

TXM tools can be driven by Groovy scripts (a Java based script language) or by R scripts.

Which languages can it work with (as of February 2023)?

The tool is fully language independent: any Unicode script and character collation.

Linguistic annotations depend on NLP tools used or annotations imported from source.

For example, TreeTagger can lemmatize English, German, Italian, Spanish, Russian, Latin, Greek, Ancient Greek, written modern French, spoken modern French, written old French, writen middle French, written early modern French, etc. (see TreeTagger description for all available languages).

Technical details

Version	0.8.2
Version Date:	2022-06-10
Works on Operating Systems:	Windows, MacOs, Linux
License:	Fully free (GNU GPLv3)
Distribution:	Local application
Download, installation, documentation, user guides:	• Download: https://txm.gitpages.huma-num.fr/textometrie/files/software/TXM/0.8.2/en/ • Project URL: https://txm.gitpages.huma-num.fr/textometrie/en/ • User Manual (French): https://txm.gitpages.huma-num.fr/txm-manual • User manual (old 0.7): https://txm.gitpages.huma-num.fr/textometrie/files/documentation/TXM%20Manual%200.7.pdf • All the docs: https://txm.gitpages.huma-num.fr/textometrie/en/Documentation • User Wiki (French) : https://groupes.renater.fr/wiki/txm-users/index (includes a FAQ) • Leaflet: http://sourceforge.net/projects/txm/files/documentation/TXM%20Leaftlet%20EN.pdf/download
User interface:	• GUI • script (Groovy) • script (R)

Docker instance:	No
How does the tool process your text:	• Import a set of documents as a corpus • Convert from several input formats to XML-TEI TXM pivot format • Run annotation tools for automatic annotation (TreeTagger, UDPipe, Stanford NLP...) • Manually correct or annotate in the GUI • Extract various observables based on lexical patterns • Build various sub-corpora and partitions • Build various visualisations and data tables from statistical analysis of observables inside sub-corpora/partitions
Tool exports results:	• As spreadsheets [results data] • As R data frames [results data] • As images (vector or bitmap) [data visualisations] • As XML-TEI [annotations]
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.
Plug in your own model:	Yes, TreeTagger can be trained from the GUI to build new language models from a corpus.
XML-TEI compatibility:	• All formats are converted to XML-TEI TXM pivot format on input, including any XML-TEI as input • The XML-TEI TXM format is a specific TEI extension that can be normalized to TEI on export.
Required data input format:	The tool imports a set of documents as a corpus. Metadata can be associated with each document through a spreadsheet file. The structure of documents depend on their type: • Written texts [structure model TEI]: any XML-TEI structure encoding, links to fac-simile images, hyperlinks, etc. • Speech transcriptions (e. g. interview recording) [structure model Transcriber]: link to media file, episodes, speech turns and word level timing with media files • Multilingual/parallel [structure model TMX]: alignments on any structure (div, p...) • As a table (e. g. answers to a survey, tweets) [structure model spreadsheet]: columns for metadata and columns for textual content

Recommended tutorials

Title	Provider/Author	URL
Explore, play, analyse your corpus with TXM: A short introduction of TXM	José Calvo and Silvia Gutiérrez, April 17 2014	http://dhd-blog.org/?p=3384

TXM-Kurzreferenz	Christof Schöch, July 4 2014	https://zenodo.org/record/10769
Videocast of TXM 0.6 initiation Workshop (in French)	Bénédicte Pincemin, September 27 2012	https://txm.gitpages.huma-num.fr/textometrie/html/enregistrement_atelier_initiation_TXM_fr.html

Which other tools from this list does this tool integrate?

- UDPipe (prototype)
- Corpus Query Processor (IMS Corpus Workbench)

Related papers

Béranger, M., Heiden, S., & Lavrentiev, A. (2015). Reengineering Akkadian Tablets with TEI and TXM for Linguistic Analysis. *TEI Conference and Members' Meeting*, 36. https://halshs.archives-ouvertes.fr/halshs-01318713

Grobol, L., Landragin, F., & Heiden, S. (2018, October). XML-TEI-URS: using a TEI format for annotated linguistic resources. *CLARIN Annual Conference 2018*. https://hal.archives-ouvertes.fr/hal-01827563

Heiden, S. (2010). The TXM Platform: Building Open-Source Textual Analysis Software Compatible with the TEI Encoding Scheme. In R. Otoguro, K. Yoshimoto, K. Ishikawa, H. Umemoto, & Y. Harada (Eds.), *24th Pacific Asia Conference on Language, Information and Computation* (pp. 389–398). Institute for Digital Enhancement of Cognitive Development, Waseda University. https://halshs.archives-ouvertes.fr/halshs-00549764

Heiden, S. (2018). Annotation-based Digital Text Corpora Analysis within the TXM Platform. *Fourteenth International Conference on the Statistical Analysis of Textual Data*, *1*, 367–374. https://hal.archives-ouvertes.fr/hal-02015898

Heiden, S. (2019). Coping With The Complexity Of The TXM Platform Annotation Services With A Unified TEI Encoding Framework [Data set]. In *Abstracts of the Digital Humanities Conference*. Digital Humanities Conference, Utrecht. DataverseNL. https://doi.org/https://doi.org/10.34894/YYKDMM

Lavrentiev, A., & Heiden, S. (2012). The TXM Portal Software giving access to Old French Manuscripts Online. *7th International Conference on Language Resources and Evaluation (LREC)*, 29–35. https://halshs.archives-ouvertes.fr/halshs-00759361

Pincemin, B., Heiden, S., & Decorde, M. (2020). Textometry on Audiovisual Corpora. In P. M. & P. RATINAUD (Ed.), *15th International Conference on Statistical Analysis of Textual Data JADT 2020*. University of Toulouse. https://halshs.archives-ouvertes.fr/halshs-02779055

Pincemin, B., Mayaffre, D., Heiden, S., & Weyl, P. (2016). Génétique mémorielle. Shoah, mémoire et ADT. In D. Mayaffre, C. Poudat, L. Vanni, V. Magri, & P. Follette (Eds.),

JADT 2016 - Statistical Analysis of Textual Data: Vol. volume 2 (pp. 495–506). Presses de FacImprimeur. https://hal.archives-ouvertes.fr/hal-01361988

Quignard, M., Heiden, S., Landragin, F., & Decorde, M. (2018). Textometric Exploitation of Coreference-annotated Corpora with TXM: Methodological Choices and First Outcomes. In M. M. Domenica Fioredistella IEZZI Livia CELARDO (Ed.), *Fourteenth International Conference on the Statistical Analysis of Textual Data* (pp. 610–615). UniversItalia. https://hal.archives-ouvertes.fr/hal-01814858

TXM portal

Quick description	Portal management
Task it solves	Create user account
Method it uses for that task	Allow authenticated or anonymous access to each corpus
Features (text enrichment)	Create user account ; verify mail address ; set user to an access profile
Metric	
Formalism	
Tagset	
Quick description	Portal management
Task it solves	Set access permissions to user profiles
Method it uses for that task	Allow or disallow access to each corpus or text by commands
Features (text enrichment)	Set user commands permissions at corpus or text level
Metric	
Formalism	
Tagset	
Quick description	Portal management
Task it solves	Start user session
Method it uses for that task	Build the user GUI in the browser
Features (text enrichment)	A similar GUI as TXM desktop is built to access and analyze corpora ; an URL based API allows to start a TXM portal session in a specific command with parameters on a specific corpus
Metric	
Formalism	
Tagset	
Quick description	Corpus management
Task it solves	Build a subcorpus
Method it uses for that task	Build a corpus configuration by selecting a set of texts or a set of text parts Corpus configuration
Features (text enrichment)	Selection of texts by a combination of their properties; selection of text structures by a combination of their properties
Metric	
Formalism	CQL
Tagset	
Quick description	Corpus management
Task it solves	Build a partition
Method it uses for that task	Build a contrastive corpus configuration by selecting a set of texts or a set of text parts Corpus configuration

Features (text enrichment)	selection of texts by a combination of their properties ; selection of text structures by a combination of their properties ; the sum of selections is always the whole corpus (resp.) the whole text
Metric	
Formalism	CQL
Tagset	
Quick description	**Information extraction**
Task it solves	Search in a text corpus
Method it uses for that task	
Features (text enrichment)	Selection of texts by a combination of their properties ; selection of text structures by a combination of their properties ; the sum of selections is always the whole corpus (resp.) the whole text
Metric	
Formalism	CQL
Tagset	
Quick description	**Information extraction**
Task it solves	Search in a text corpus
Method it uses for that task	Linear text search with Corpus Query Processor Quantification
Features (text enrichment)	Lexical pattern observables extractions; Text snippets matching a query ; Word lists matching a query
Metric	
Formalism	CQL
Tagset	
Quick description	**Information extraction**
Task it solves	Search in a parsed text corpus (treebank)
Method it uses for that task	Syntactic tree search Quantification
Features (text enrichment)	Syntactic trees matching a query ; Text snippets matching a query ; Word lists matching a query ; Lexical pattern observables extractions
Metric	
Formalism	TIGER Search, CQL
Tagset	
Quick description	**Corpus analysis**
Task it solves	Tabulate the contexts of occurrences of word patterns
Method it uses for that task	Concordance analysis Qualitative analysis
Features (text enrichment)	Tabulate word pattern occurrences with reference, left context, occurrence and right context columns ; Bring similar contexts together by multiple sort on any column ; Link to the text edition (to read) with occurrence highligthed ; Link to audio/video stream (to play) synchronized with occurrence
Metric	
Formalism	CQL, TIGER Search
Tagset	

Quick description	Corpus Analysis
Task it solves	Tabulate word properties of occurrences of word patterns
Method it uses for that task	Hierarchical Index analysis Paradigmatic analysis Quantitative analysis
Features (text enrichment)	Tabulate word pattern properties values (eg combine word form, pos or lemma) of occurrences with their frequency
Metric	
Formalism	CQL, TIGER Search
Tagset	
Quick description	Corpus Analysis
Task it solves	Tell which words typically co-occur with a given word
Method it uses for that task	Collocations analysis Syntagmatico-Paradigmatic analysis Quantitative analysis
Features (text enrichment)	
Metric	Statistics expressing the mutual attraction of two words or short text segments (specificity statistical measure)
Formalism	**CQL**
Tagset	
Quick description	**Corpus Analysis**
Task it solves	Tell which words typically occur within a sub-corpus or inside parts of a partition
Method it uses for that task	Keywords analysis of a partition or subcorpus Paradigmatic analysis Quantitative analysis
Features (text enrichment)	
Metric	Statistics expressing the attraction of a word or a short text segment with a sub-corpus of the whole corpus or with parts of a partition of the corpus (specificity statistical measure) Visualisation: Bar plot showing statistic value for each word pattern searched
Formalism	**CQL**
Tagset	
Quick description	**Corpus Analysis**
Task it solves	Show similarities between texts or structures based on word pattern frequencies
Method it uses for that task	Correspondence factor analysis of a partition Analytical views production and interaction Paradigmatic analysis Quantitative analysis
Features (text enrichment)	factorial coordinates, contributions, quality of representation Visualisation: factorial planes
Metric	chi-square
Formalism	**CQL**
Tagset	

Quick description	**Corpus Analysis**
Task it solves	Give summary statistics of a corpus or selected texts
Method it uses for that task	Various statistics
Features (text enrichment)	Token properties (form, pos, lemma) list and frequencies; Structures properties list and frequencies
Metric	
Formalism	
Tagset	

What can this tool do for you?

A TXM portal hosts textual corpora for online access and analysis.

Users access their corpora online with a simple web browser. No TXM software installation is needed. TXM web portals are often used for courses with students.

Depending on access permission settings, access can be anonymous (by anybody) or need a connexion and specific rights. This allows to share corpora within working groups.

TXM portals allows to host rich text editions and combine them with analytic tools. See below for some examples of direct access to text editions inside a corpus hosted in a TXM portal.

A TXM portal is managed by a special user called a ‘portal administrator’ who manages corpora, user accounts and access permissions.

Corpora are first created with the TXM desktop software, then exported in a .txm file and uploaded to a TXM portal.

See the documentation of the tool “TXM desktop” for further details on the kind of corpora that can be hosted and the tools provided for analysis.

A TXM portal doesn’t implement all the analysis tools of the TXM desktop software, but whose two tools share the same platform core and are developped together.

For an online demo see: https://txm-demo.huma-num.fr/txm/.

Examples of direct access to a TXM portal by URL API:

- Direct access to an Edition page of an Old French (1225) Holy Grail manuscript transcription highligthing words “dist” and “Lancelot” with a synoptic view of 'facsimile, diplomatic and critical edition': http://portal.textometrie.org/demo?command=edition&path=/GRAAL&textid=qgraal_c

m&editions=ms-colonne,diplomatique,courante&wordids=w_106_030047,w_106_030049

- Direct access to a Concordance of the word 'Lancelot' (then double-click on a line to read the full text): http://portal.textometrie.org/demo?command=concordance&path=/GRAAL&query=%22Lancelot%22
- Direct access to an Edition page of an ancient Babylonian (-IIth mil.) tablet transcription with synoptic view combining the transliterated version, the cuneiform version and the facsimile image of the tablet: http://portal.textometrie.org/demo/?command=edition&path=/OBLCUNEIF&textid=TXM_cuneif_Ha_S_AbB_2_4&editions=translit,cuneiform,facs&pageid=3

Examples of public TXM portals:

- Lyon, BFM-TXM portal (Old French litterature), UMR IHRIM laboratory: http://txm.bfm-corpus.org/
- Tours, BVH-TXM portal (Montaigne, Rabelais, Ronsard, etc.), UMR CESR laboratory: http://txm.bvh.univ-tours.fr/txm
- Montpellier, Praxiling TXM portal, UMR PRAXILING laboratory: http://textometrie.univ-montp3.fr/txm
- Paris, TXM portal of the French TreeBank (FTB) corpus, UMR LLF laboratory: http://manganese.lab.parisdescartes.fr:8813/txm/
- Besançon, TXM portal, EA ELLIADD laboratory: http://fanum-txm.univ-fcomte.fr/txm

Which languages can it work with (as of February 2023)?

Like the TXM desktop, the tool is fully language independent: any Unicode script and character collation.

Linguistic annotations depend on NLP tools used or annotations imported from source.

For example, TreeTagger can lemmatize English, German, Italian, Spanish, Russian, Latin, Greek, Ancient Greek, written modern French, spoken modern French, written old French, writen middle French, written early modern French, etc. (see TreeTagger description for all available languages).

Technical details

Version	0.6.3
Version Date:	2022-12-02
Works on Operating Systems:	Linux - Ubuntu 20.04
License:	Fully free (GNU GPLv3)

Distribution:	• Web application
	  ○ Administrators install the TXM portal software on a Ubuntu server
	  ○ Users access the TXM portal through a simple web browser (no installation)

Download, installation, documentation, user guides:	• Download: https://txm.gitpages.huma-num.fr/textometrie/files/software/TXM%20portal/0.6.3/ • See demo portal: https://txm-demo.huma-num.fr/txm/ • Project URL: https://txm.gitpages.huma-num.fr/textometrie/en/ • Portal Administrator Manual (French): https://txm.gitpages.huma-num.fr/textometrie/files/documentation/Portail%20TXM/Manuel%20du%20portail%20TXM%200.6.3%20-%20admin.pdf • All the docs: https://txm.gitpages.huma-num.fr/textometrie/en/Documentation
User interface:	• GUI • API
Docker instance:	VMWare Virtual Machine (to clone) available at https://documentation.huma-num.fr/txm
How does the tool process your text:	• No source text processing on a TXM portal • Corpora are first created locally from sources with TXM desktop software, then exported to a .txm file, and then uploaded to a TXM portal
Tool exports results:	• As .csv [results data] • As images (bitmap) [data visualisations]
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	• All formats are converted to XML-TEI TXM pivot format on input, including any XML-TEI as input • The XML-TEI TXM format is a specific TEI extension that can be normalized to TEI on export. • If a corpus uploaded to a TXM portal contains the text pivot sources (option), then the corpus download from the TXM portail delivers the XML-TEI TXM text files

Required data input format:	The tool imports a set of documents as a corpus. Metadata can be associated with each document through a spreadsheet file. The structure of documents depend on their type: • Written texts [structure model TEI]: any XML-TEI structure encoding, links to fac-simile images, hyperlinks, etc. • Speech transcriptions (e. g. interview recording) [structure model Transcriber]: link to media file, episodes, speech turns and word level timing with media files • Multilingual/parallel [structure model TMX]: alignments on any structure (div, p...) • As a table (e. g. answers to a survey, tweets) [structure model spreadsheet]: columns for metadata and columns for textual content

Recommended tutorials

Title	Provider/Author	URL
Explore, play, analyse your corpus with TXM: A short introduction of TXM	José Calvo and Silvia Gutiérrez, April 17 2014	http://dhd-blog.org/?p=3384
TXM-Kurzreferenz	Christof Schöch, July 4 2014	https://zenodo.org/record/10769
Videocast of TXM 0.6 initiation Workshop (in French)	Bénédicte Pincemin, September 27 2012	https://txm.gitpages.huma-num.fr/textometrie/html/enregistrement_atelier_initiation_TXM_fr.html

Which other tools from this list does this tool integrate?

- TreeTagger
- Corpus Query Processor
- R

Related papers

Béranger, M., Heiden, S., & Lavrentiev, A. (2015). Reengineering Akkadian Tablets with TEI and TXM for Linguistic Analysis. *TEI Conference and Members' Meeting*, 36. https://halshs.archives-ouvertes.fr/halshs-01318713

Grobol, L., Landragin, F., & Heiden, S. (2018, October). XML-TEI-URS: using a TEI format for annotated linguistic resources. *CLARIN Annual Conference 2018*. https://hal.archives-ouvertes.fr/hal-01827563

Heiden, S. (2010). The TXM Platform: Building Open-Source Textual Analysis Software Compatible with the TEI Encoding Scheme. In R. Otoguro, K. Yoshimoto, K. Ishikawa, H. Umemoto, & Y. Harada (Eds.), *24th Pacific Asia Conference on Language, Information and Computation* (pp. 389–398). Institute for Digital Enhancement of Cognitive Development, Waseda University. https://halshs.archives-ouvertes.fr/halshs-00549764

Heiden, S. (2018). Annotation-based Digital Text Corpora Analysis within the TXM Platform. *Fourteenth International Conference on the Statistical Analysis of Textual Data*, *1*, 367–374. https://hal.archives-ouvertes.fr/hal-02015898

Heiden, S. (2019). Coping With The Complexity Of The TXM Platform Annotation Services With A Unified TEI Encoding Framework [Data set]. In *Abstracts of the Digital Humanities Conference*. Digital Humanities Conference, Utrecht. DataverseNL. https://doi.org/https://doi.org/10.34894/YYKDMM

Lavrentiev, A., & Heiden, S. (2012). The TXM Portal Software giving access to Old French Manuscripts Online. *7th International Conference on Language Resources and Evaluation (LREC)*, 29–35. https://halshs.archives-ouvertes.fr/halshs-00759361

Pincemin, B., Heiden, S., & Decorde, M. (2020). Textometry on Audiovisual Corpora. In P. M. & P. RATINAUD (Ed.), *15th International Conference on Statistical Analysis of Textual Data JADT 2020*. University of Toulouse. https://halshs.archives-ouvertes.fr/halshs-02779055

Pincemin, B., Mayaffre, D., Heiden, S., & Weyl, P. (2016). Génétique mémorielle. Shoah, mémoire et ADT. In D. Mayaffre, C. Poudat, L. Vanni, V. Magri, & P. Follette (Eds.), *JADT 2016 - Statistical Analysis of Textual Data: Vol. volume 2* (pp. 495–506). Presses de FacImprimeur. https://hal.archives-ouvertes.fr/hal-01361988

Quignard, M., Heiden, S., Landragin, F., & Decorde, M. (2018). Textometric Exploitation of Coreference-annotated Corpora with TXM: Methodological Choices and First Outcomes. In M. M. Domenica Fioredistella IEZZI Livia CELARDO (Ed.), *Fourteenth International Conference on the Statistical Analysis of Textual Data* (pp. 610–615). UniversItalia. https://hal.archives-ouvertes.fr/hal-01814858

NATURAL LANGUAGE PROCESSING (NLP) TOOLS

Calc

Quick description	statistics
Task it solves	Calculate basic statistical tasks for common research questions
Method it uses for that task	Various formulas
Features (text enrichment)	--
Metric	Various metrics
Formalism	--
Tagset	--

What can this tool do for you?

Calc provides quick support to users when calculating basic statistical tasks most commonly encountered in corpus research. The GUI is divided into a number of modules reflecting specific research problems. Unlike other similar tools, Calc is task-based, which means that suitable statistical tests have already been pre-selected for a given task, so that users can draw statistically well-founded conclusions based on the input data.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Any (it is language-agnostic)	

Technical details

Version	--
Version Date:	--
Works on Operating Systems:	Linux
License:	fully free
Distribution:	Shiny web application
Download, installation, documentation, user guides:	The tool is running at https://korpus.cz/calc/, where also the documentation is available.
User interface:	GUI
Docker instance:	No
How does the tool process your text:	
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and but it does not include statistical models, but it implements basic statistical tests.

XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	The tool does not process a text. The input is typically numbers (frequencies) and the output is statistical significance, confidence intervals etc.

Flair

Quick description	NLP
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Tokens; Sentence borders
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Give the basic dictionary form of a token
Method it uses for that task	Lemmatization
Features (text enrichment)	Lemmas, aka dictionary forms of tokens
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Determine part of speech and relevant morphological categories
Method it uses for that task	Morphological tagging
Features (text enrichment)	Parts of speech; Morphological categories;
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies
Quick description	NLP
Task it solves	Determine syntactic relations between tokens (e. g. verb subjects)
Method it uses for that task	Syntactic parsing
Features (text enrichment)	Syntactic relations between tokens within one sentence
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies
Quick description	NLP
Task it solves	Extract entities such as persons, dates, and locations
Method it uses for that task	Named Entity Recognition
Features (text enrichment)	Labels on recognized entities
Metric	
Formalism	

Tagset	PER (person), LOC (location), ORG (organisation), MISC (miscellaneous)
Quick description	**NLP**
Task it solves	Connect named entities in a text with knowledge bases
Method it uses for that task	Entity Linking (aka Wikification, Grounding)
Features (text enrichment)	Mentions of recognized entities linked to external authority databases, e. g. WikiData
Metric	
Formalism	
Quick description	**NLP**
Task it solves	Identify mentions of an entity even when they are expressed by a personal pronoun
Method it uses for that task	Coreference (anaphora) resolution
Features (text enrichment)	References between personal or possessive pronouns referring to the same antecedent (typically a noun)
Metric	
Formalism	
Tagset	
Quick description	**NLP**
Task it solves	Sentiment analysis (aka Opinion mining, Emotion AI)
Method it uses for that task	Text classification (classes: positive, neutral, negative)
Features (text enrichment)	Labels on recognized sentiment of a chunk of text.
Metric	
Formalism	
Tagset	positive, neutral, negative

What can this tool do for you?

Flair is an open-source Python library which allows users to 1) use the trained models provided by Flair and 2) to train their own NLP-models using the provided Python library framework. The tool builds upon the state-of-the-art deep learning Pytorch framework.

- Examples of how a model can be trained can be found on their GitHub pages: https://github.com/flairNLP/flair/blob/master/resources/docs/TUTORIAL_7_TRAINING_A_MODEL.md .

Tasks you can perform include but are not limited to Named Entity Recognition (NER), sentiment analysis, Parts-of-speech tagging (POS) and text classification.
Since the tool is a Python library, you need a programming environment to use it. You can also use it in Colab, Google's online IDE (example: https://www.analyticsvidhya.com/blog/2019/02/flair-nlp-library-python/).

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Danish	
Dutch	
English	
French	
German	
Spanish	

Technical details

Version	0.11
Version Date:	2022-04-10
Works on Operating Systems:	Windows, Unix, MacOs, Linux
License:	Fully free
Distribution:	Library for Python
Download, installation, documentation, user guides:	https://github.com/flairNLP/flair
User interface:	None
Docker instance:	No or not known
How does the tool process your text:	Automatically adds some labels (annotation) to your texts, whether it is a sentiment label on the level of a text chunk or an entity label on the sequence level.
Tool exports results:	Yes
Statistical models:	The tool provides a framework to train your own Named Entity Recognition and sentiment analysis models, and to use the already trained Huggingface transformer models trained and made available by Flair.
Plug in your own model:	You can easily train your own statistical model using the framework provided by this tool and your own labeled data.
XML-TEI compatibility:	The tool does not support XML-TEI at all, only plain text.
Required data input format:	The tool accepts plain text as input to generate labels. To train models, it requires the data to be in a specific format as specified on their GitHub.

Related papers

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., Vollgraf, R., 2019. FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP, in: Proceedings of the 2019 Conference of the North. Presented at the Proceedings of the 2019 Conference of the North, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 54–59. https://doi.org/10.18653/v1/N19-4010

Akbik, A., Blythe, D., Vollgraf, R., 2018. Contextual String Embeddings for Sequence Labeling. Presented at the COLING 27th International Conference on Computational Linguistics, pp. 1638–1649.

NameTag 1

Quick description	NLP
Task it solves	extraction of flat named entities
Method it uses for that task	Named-Entity Recognition
Features (text enrichment)	named-entity labels
Metric	
Formalism	
Tagset	depending on language model

What can this tool do for you?

This tool marks named entities in your texts. You can easily supply to it your own NER model in any language.

Which languages can it work with (as of February 2023)?

This list presents currently available language models. Your own model can be for any language.

Language	Variety (geographical, or temporal if not modern)
Czech	
English	

Technical details

Version	1.2.0
Version Date:	February 2023 (maintenance version of a release from 2021-04-21)
Works on Operating Systems:	Linux
License:	Fully free (Mozilla Public License 2.0)
Distribution:	• webservice (http://lindat.mff.cuni.cz/services/nametag); • code https://github.com/ufal/nametag/releases/tag/v1.2.0
Download, installation, documentation, user guides:	http://hdl.handle.net/11858/00-097C-0000-0023-43CE-E https://ufal.mff.cuni.cz/nametag/2 https://lindat.mff.cuni.cz/services/nametag/ https://github.com/ufal/nametag/releases/tag/v2.0.0
User interface:	web GUI, REST API
Docker instance:	no
How does the tool process your text:	The tool detects and marks named entities.
Tool exports results:	
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.

Plug in your own model:	You can easily train your statistical model to plug in this tool.
XML-TEI compatibility:	The tool produces XML tags but does not support XML-TEI input.
Required data input format:	plain text, vertical, CoNLL-U

Which other tools from this list does this tool integrate?

NameTag1 models

Recommended tutorials

- https://ufal.mff.cuni.cz/nametag/1/tutorials (in Czech)
- https://github.com/ufal/nametag/blob/master/bindings/python/examples/run_ner_simple.py

Related papers

Straková, J., Straka, M., & Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 13–18. http://www.aclweb.org/anthology/P/P14/P14-5003.pdf

NameTag1 Models

Quick description	NLP
Task it solves	identify and mark named entities in texts
Method it uses for that task	Named-entity recognition – machine learning
Features (text enrichment)	named-entity labels
Metric	
Formalism	dependent on language model
Tagset	dependent on language model

What can this tool do for you?

These are models to plug in the NameTag2 named-entity recognizer.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Czech	
English	

Technical details

Version	
Version Date:	2014
Works on Operating Systems:	
License:	Fully free (CC-BY-NC-SA 4.0)
Distribution:	zipped files from the LINDAT/CLARIAH-CZ repository
Download, installation, documentation, user guides:	• English Model (CoNLL-2003) for NameTag: http://hdl.handle.net/11234/1-3118 Czech Models (CNEC) for NameTag: http://hdl.handle.net/11858/00-097C-0000-0023-7D42-8 •
User interface:	None
Docker instance:	No
How does the tool process your text:	It gives the named-entity recognizer the know-how of named-entity annotation in a given language, according to a given tagset.
Tool exports results:	No
Statistical models:	This tool is a statistical model / a set of statistical models.
XML-TEI compatibility:	This tool does not support XML-TEI at all.
Required data input format:	None

Related papers

Straková, J., Straka, M., & Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 13–18. http://www.aclweb.org/anthology/P/P14/P14-5003.pdf

NameTag 2

Quick description	NLP
Task it solves	extraction of named entities, even nested
Method it uses for that task	Named-Entity Recognition
Features (text enrichment)	named-entity labels, also nested
Metric	
Formalism	
Tagset	depending on language model

What can this tool do for you?

This tool marks named entities in your texts. When named entities are nested, it recognizes the nestings. For instance, two entities can be recognized in *Humboldt University*: institution (the whole string) and surname (*Humboldt*).

Which languages can it work with (as of February 2023)?

This list presents currently available language models.

Language	Variety (geographical, or temporal if not modern)
Czech	
Dutch	
English	
German	
Spanish	
Ukrainian	

Technical details

Version	2.0.0
Version Date:	2021-04-21
Works on Operating Systems:	Linux
License:	Fully free (Mozilla Public License 2.0)
Distribution:	• webservice (http://lindat.mff.cuni.cz/services/nametag); • code to run your own NameTag server • commandline tool at https://github.com/ufal/nametag/releases/tag/v2.0.0
Download, installation, documentation, user guides:	http://hdl.handle.net/11858/00-097C-0000-0023-43CE-E https://ufal.mff.cuni.cz/nametag/2 https://lindat.mff.cuni.cz/services/nametag/ https://github.com/ufal/nametag/releases/tag/v2.0.0
User interface:	web GUI, REST API
Docker instance:	no

How does the tool process your text:	The tool detects and marks named entities.
Tool exports results:	
Statistical models:	This tool is (not) a statistical model / a set of statistical models, and/ but it does (not) include statistical models.
Plug in your own model:	You can(not) easily train your statistical model to plug in this tool.
XML-TEI compatibility:	The tool produces XML tags but does not support XML-TEI input.
Required data input format:	plain text, vertical, CoNLL-U

Which other tools from this list does this tool integrate?

NameTag2 models

Related papers

Straková, J., Straka, M., & Hajič, J. (2019). Neural Architectures for Nested NER through Linearization. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 5326–5331.

NameTag2 Models

Quick description	NLP
Task it solves	identify and mark named entities in texts
Method it uses for that task	Named-entity recognition – machine learning
Features (text enrichment)	named-entity labels
Metric	
Formalism	dependent on language model
Tagset	dependent on language model

What can this tool do for you?

These are models to plug in the NameTag2 named-entity recognizer.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Czech	
Dutch	
English	
German	
Spanish	
Ukrainian	

Technical details

Version	
Version Date:	
Works on Operating Systems:	
License:	Fully free (CC-BY-NC-SA 4.0)
Distribution:	zipped files from the LINDAT/CLARIAH-CZ repository
Download, installation, documentation, user guides:	https://ufal.mff.cuni.cz/nametag/2/models https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-3773
User interface:	None
Docker instance:	No
How does the tool process your text:	It gives the named-entity recognizer the know-how of named-entity annotation in a given language, according to a given tagset.
Tool exports results:	
Statistical models:	This tool is a statistical model / a set of statistical models.
XML-TEI compatibility:	This tool does not support XML-TEI at all.

Required data input format:	None

Related papers

Straková, J., Straka, M., & Hajič, J. (2019). Neural Architectures for Nested NER through Linearization. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 5326–5331.

QuitaUp

Quick description	Textometry
Task it solves	Give summary statistics of a corpus or selected texts
Method it uses for that task	Various formulas
Features (text enrichment)	
Metric	Token frequencies, Lexical richness/diversity; Key words, Thematic concentration, Descriptivity/activity, Verb distance, Entropy, Hapaxes
Formalism	
Tagset	

What can this tool do for you?

After you upload a text, the tool performs basic text-processing functions using UD Pipe (tokenization, lemmatization, POS tagging, syntactic parsing). These functions are available only for ca 20 languages. The tool then calculates ca 16 stylometric indices of the input text.

Which languages can it work with (as of February 2023)?

The tool supports all languages that can be parsed with UDPipe.

Technical details

Version	Unknown
Version Date:	Unknown
Works on Operating Systems:	Linux
License:	Fully free
Distribution:	Shiny web application
Download, installation, documentation, user guides:	The tool is running at https://korpus.cz/quitaup/, where also its documentation is available.
User interface:	GUI
Docker instance:	No
How does the tool process your text:	Returns some metrics as numbers or tables.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
Plug in your own model:	You can(not) easily train your statistical model to plug in this tool.
XML-TEI compatibility:	The tool does not support XML-TEI at all.

Required data input format:	The tool can handle .txt, .docx, .doc, .odt, .rtf or .pdf formats and treats them all as plain text.

Which other tools from this list does this tool integrate?

UDPipe

spaCy

Quick description	NLP
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Tokens; Sentence borders
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Give the basic dictionary form of a token
Method it uses for that task	Lemmatization
Features (text enrichment)	Lemmas, aka dictionary forms of tokens
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Determine part of speech and relevant morphological categories
Method it uses for that task	Morphological tagging
Features (text enrichment)	
Metric	
Formalism	
Tagset	Universal Dependencies
Quick description	NLP
Task it solves	Determine syntactic relations between tokens (e. g. verb subjects)
Method it uses for that task	Syntactic parsing
Features (text enrichment)	Syntactic relations between tokens within one sentence as labels on each token
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies
Quick description	NLP
Task it solves	Extract entities such as persons, dates, and locations
Method it uses for that task	Named Entity Recognition
Features (text enrichment)	Labels on recognized entities
Metric	
Formalism	

Tagset	
Quick description	**NLP**
Task it solves	Connect named entities in a text with knowledge bases
Method it uses for that task	Entity Linking (aka Wikification, Grounding)
Features (text enrichment)	Mentions of recognized entities linked to external authority databases, e. g. WikiData
Metric	
Formalism	
Tagset	
Quick description	**NLP**
Task it solves	Identify mentions of an entity even when they are expressed by a personal pronoun
Method it uses for that task	Coreference (anaphora) resolution
Features (text enrichment)	Features: References between personal or possessive pronouns referring to the same antecedent (typically a noun)
Metric	
Formalism	
Tagset	
Quick description	**NLP**
Task it solves	Sentiment analysis (aka Opinion mining, Emotion AI)
Method it uses for that task	Text classification (classes: positive, neutral, negative)
Features (text enrichment)	Labels on recognized sentiment of a chunk of text.
Metric	
Formalism	
Tagset	positive, neutral, negative

What can this tool do for you?

spaCy is a free open-source Python library for advanced Natural Language Processing purposes.

- Everything the library can do is neatly outlined here: https://spacy.io/usage/spacy-101/ .

Tasks you can perform include but are not limited to Named Entity Recognition (NER), sentiment analysis, Parts-of-speech tagging (POS), lemmatization, entity linking, rule-based matching, and text classification. It allows you to train your own domain-specific models – but you can also use the ones provided by the library.
Since the tool is a Python library, you need a programming environment to use it, and knowledge of NLP. For digital humanities purposes, there is a great YouTube series (https://www.youtube.com/@python-programming) which shows how you can use spaCy in different DH settings (from information extraction to visualization).

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Catalan, Valencian	
Chinese	
Croatian	
Danish	
Dutch (not Flemish)	
English	
Finnish	
French	
German	
Greek	
Italian	
Japanese	
Korean	
Lithuanian	
Macedonian	
Norwegian Bokmål	
Polish	
Portuguese	
Romanian	
Russian	
Spanish	
Swedish	
Ukrainian	

Technical details

Version	V3.5
Version Date:	2022-04-10
Works on Operating Systems:	Windows, Unix, MacOs, Linux
License:	Fully free
Distribution:	Library for Python
Download, installation, documentation, user guides:	https://spacy.io/
User interface:	None
Docker instance:	No or not known
How does the tool process your text:	Automatically adds some labels (annotation) to your texts.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.
Plug in your own model:	Yes. (see explanation here: https://spacy.io/usage/spacy-101/).

XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	It accepts plain text as input to generate labels. To train models, it requires the data to be in a specific format as specified on their website (https://spacy.io/).

Recommended tutorials

Title	Provider/ Author	URL
Introduction to spaCy	Python Tutorials for Digital Humanities	https://www.youtube.com/@python-programming spaCy series: https://www.youtube.com/watch?v=8HZ4BjWMod4&list=PL2VXyKi-KpYvuOdPwXR-FZfmZ0hjoNSUo

Related papers

Montani, I., Honnibal, M., Honnibal, M., Van Landeghem, S., Boyd, A., Peters, H., McCann, P.O., Geovedi, J., O'Regan, J., Samsonov, M., Orosz, G., De Kok, D., Altinok, D., Kristiansen, S.L., Madeesh Kannan, Bournhonesque, R., Lj Miranda, Baumgartner, P., Edward, Explosion Bot, Hudson, R., Mitsch, R., Roman, Fiedler, L., Ryn Daniels, Wannaphong Phatthiyaphaibun, Howard, G., Tamura, Y., Bozek, S., 2023. explosion/spaCy: v3.5.0: New CLI commands, language updates, bug fixes and much more. https://doi.org/10.5281/ZENODO.1212303

Udapi

Quick description	NLP
Task it solves	programmatically querying, editing, and computing statistics over CoNLL-U files – many routines pre-programmed but, most importantly, you can write your own.
Method it uses for that task	traversing syntactic trees of Universal Dependencies
Features (text enrichment)	user provides its own features
Metric	user can compute their own statistics
Formalism	Universal Dependencies
Tagset	Universal Dependencies

What can this tool do for you?

Udapi is a framework providing an API for processing Universal Dependencies data. You can use it to similar tasks as the tree query languages PMLTQ or Grew, but it gives you much more freedom. Also, you can run Udapi on your data directly, while both PMLTQ and Grew need to first create a relational database of your entire data. This also means that you can process larger data with Udapi than with PMLTQ and Grew.
However, this comes at a cost – you must be able to write a Python script and read developer documentation. So far, the tools has been used in the treebanking community, but there is no fool-proof tutorial yet for newcomers to the field.
The tool is structured in packages and modules (https://udapi.readthedocs.io/en/latest/modules.html). To search trees, the *udapi.core* package is the relevant one. Nodes and their attributes are represented as object classes with defined properties and methods. To explain the logic of these properties and methods, tutorials tailored to the CLS community are likely to be needed.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Any (it is language-agnostic)	

Technical details

Version	0.3.0
Version Date:	2022-04-06
Works on Operating Systems:	Windows, MacOs, Unix, Linux
License:	Fully free (GNU GPL3.0)
Distribution:	Python library
Download, installation, documentation, user guides:	https://udapi.readthedocs.io https://github.com/udapi/udapi-python https://github.com/udapi/

User interface:	none
Docker instance:	No or not known
How does the tool process your text:	The tool reads connl-u files and enables search, visualization as well as extraction of text/annotation snippets and tree manipulation.
Tool exports results:	
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	This tool does not support XML-TEI at all.
Required data input format:	CoNLL-U

Recommended tutorials

Title	Provider/Author	URL
Udapi Tutorial	Martin Popel	https://udapi.github.io/tutorial/
Slides about Udapi	Martin Popel	https://ufal.mff.cuni.cz/~popel/papers/2017_03_13_working_with_ud.pdf
Udapi visualization tutorial	Martin Popel	https://nbviewer.org/github/udapi/udapi-python/blob/master/tutorial/01-visualizing.ipynb

Related papers

Popel, M., Žabokrtský, Z., & Vojtek, M. (2017). Udapi: Universal API for Universal Dependencies. *NoDaLiDa 2017 Workshop on Universal Dependencies*, 96–101.

UDPipe1

Quick description	NLP
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Features: Tokens; Sentence borders; Paragraphs
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Give the basic dictionary form of a token
Method it uses for that task	Lemmatization
Features (text enrichment)	Lemmas, aka dictionary forms of tokens
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Determine part of speech and relevant morphological categories
Method it uses for that task	Determine part of speech and relevant morphological categories Morphological tagging
Features (text enrichment)	Parts of speech; Morphological categories
Metric	
Formalism	
Tagset	Universal Dependencies
Quick description	NLP
Task it solves	Determine syntactic relations between tokens (e. g. verb subjects)
Method it uses for that task	Syntactic parsing
Features (text enrichment)	Syntactic relations between tokens within one sentence
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies

What can this tool do for you?

This is an older version of UDPipe2, the tagger, lemmatizer, and syntactic parser. It uses language models trained on the Universal Dependencies treebank collection, which you can browse and learn more about at universaldependencies.org. The language models are retrained twice a year, after the regular updates of the treebank collection by the Universal Dependencies

Treebanks team. Depending on the changes in the treebank collection, the update may provide more training data for already existing language models or even new language models.
UDPipe1 achieves a somewhat lower performance than UDPipe2, but it is comfortably trainable by the users, given annotated data in the CoNLL-U format. It is still being maintained.

Which languages can it work with (as of February 2023)?

Note that UDPipe 1 models differ from UDPipe2 models. The last pre-trained model version is 2.5 and contains fewer languages than more recent models. Also, some training corpora for the represented languages may be smaller.
The parsers itself is language-agnostic. The list below shows the pre-trained models.

Language	Variety (geographical, or temporal if not modern)
Afrikaans	
Arabic	
Armenian	
Basque	
Belarusian	
Bulgarian	
Catalan, Valencian	
Chinese	
Chinese	literary
Coptic	
Croatian	
Czech	
Danish	
Dutch (not Flemish)	
Estonian	
English	
Estonian	
Finnish	
French	
French	old (842-ca. 1400)
Gaelic	Scottish
Galician	
German	
Gothic	
Greek	ancient (to 1453)
Greek	modern (since 1453)
Hebrew	ancient
Hebrew	modern
Hindi	
Hungarian	
Irish	
Italian	
Japanese	
Korean	
Latin	
Latvian	

Lithuanian	
Maltese	
Marathi	
Norwegian	Nynorsk
Norwegian	Bokmål
Old Church Slavonic	
Persian	
Polish	
Portuguese	
Romanian	
Russian	
Russian	old
Serbian	
Slovak	
Slovenian	
Spanish	
Swedish	
Tamil	
Telugu	
Turkish	
Uighur	
Ukrainian	
Urdu	
Vietnamese	
Wolof	

Technical details

Version	1.3.0
Version Date:	2023-02-16
Works on Operating Systems:	Linux, Windows, OsX
License:	Fully free (Mozilla Public License 2.0)
Distribution:	UDPipe is available as a binary for Linux/Windows/OS X, and as a library for C++, Python, Perl, Java, C#. Third-party R CRAN package also exists.
Download, installation, documentation, user guides:	http://github.com/ufal/udpipe http://lindat.mff.cuni.cz/services/udpipe/ http://ufal.mff.cuni.cz/udpipe http://hdl.handle.net/11234/1-1702 (as of March 2023, only the 2016 version)
User interface:	
Docker instance:	http://github.com/samisalkosuo/udpipe-rest-server-docker (third-party)

How does the tool process your text:	It recognizes tokens and adds labels to each token, regarding its morphological characteristics as well as its syntactic relation to a governing token in the Universal Dependencies formalism.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.
Plug in your own model:	You can easily train your statistical model to plug in this tool.
XML-TEI compatibility:	This tool does not support XML-TEI at all.
Required data input format:	Plain text, CoNLL-U, vertical... see documentation for more detail on accepted plain-text formats.

Which other tools from this list does this tool integrate?

- Universal Dependencies Language Models

Related papers

Straka, M. (2018). UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 197–207. https://doi.org/10.18653/v1/K18-2020

Agić, Ž., Aranzabe, M., Atutxa, A., Bosco, C., Choi, J., Marneffe, M.-C. de, Dozat, T., Farkas, R., Foster, J., Ginter, F., Goenaga, I., Gojenola, K., Goldberg, Y., Hajič, J., Johannsen, A., Kanerva, J., Kuokkala, J., Laippala, V., Lenci, A., … Zeman, D. (2015). *Universal Dependencies 1.1*. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Straková, J., Straka, M., & Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 13–18.

UDPipe2

Quick description	NLP
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Features: Tokens; Sentence borders; Paragraphs
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Give the basic dictionary form of a token
Method it uses for that task	Lemmatization
Features (text enrichment)	Lemmas, aka dictionary forms of tokens
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Determine part of speech and relevant morphological categories
Method it uses for that task	Determine part of speech and relevant morphological categories Morphological tagging
Features (text enrichment)	Parts of speech; Morphological categories
Metric	
Formalism	
Tagset	Universal Dependencies
Quick description	NLP
Task it solves	Determine syntactic relations between tokens (e. g. verb subjects)
Method it uses for that task	Syntactic parsing
Features (text enrichment)	Syntactic relations between tokens within one sentence
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies

What can this tool do for you?

This is a tagger, lemmatizer, and syntactic parser. It uses language models trained on the Universal Dependencies treebank collection, which you can browse and learn more about at universaldependencies.org. The language models are retrained twice a year, after the regular updates of the treebank collection by the Universal Dependencies Treebanks team. Depending

on the changes in the treebank collection, the update may provide more training data for already existing language models or even new language models.
UDPipe is a state-of-the art system. This means that it is able to learn from less data than older systems. However, the truth is, still, that the quality of the analysis depends on the size of training data related to the complexity of the language. Hence, e. g. languages with rich inflection or very specific syntax need bigger training data. Besides, the UDPipe developers do normally not actively look for external resources for specific languages, such as morphological lexicons. There are known cases of parsers optimized for one given language with bigger data and additional resources, which perform better than UDPipe, e. g. the Magyarlanc parser for Hungarian.
When considering several parsers for your language, have a look at the evaluation of the UDPipe performance with the models from a given relase here:
https://ufal.mff.cuni.cz/udpipe/2/models
When you evaluate this parser for your data, mind to record the model and version you were using. Note that one language can have several separate models from different domains, e. g. newspapers and data from social networks!

Which languages can it work with (as of February 2023)?

The parser itself is language-agnostic, but it works with language-specific models. The current models exist for languages listed in the table below.

Language	Variety (geographical, or temporal if not modern)
Some, but they do not seem to be explicitly listed	
Afrikaans	
Arabic	
Armenian	
Armenian	Western
Basque	
Belarusian	
Bulgarian	
Catalan, Valencian	
Chinese	
Chinese	literary
Coptic	
Croatian	
Czech	
Danish	
Dutch (not Flemish)	
Estonian	
English	
English	Nigerian Pidgin
Estonian	
Faroese	
Finnish	
French	

French	old (842-ca. 1400)
Gaelic	Scottish
Galician	
Gambian/Wolof	
German	
Gothic	
Greek	ancient (to 1453)
Greek	modern (since 1453)
Hebrew	ancient
Hebrew	modern
Hindi	
Hungarian	
Icelandic	
Indonesian	
Irish	
Italian	
Japanese	
Korean	
Latin	
Latvian	
Lithuanian	
Maltese	
Marathi	
Sami	Northern
Norwegian	Nynorsk
Norwegian	Bokmål
Old Church Slavonic	
Persian	
Polish	
Portuguese	
Romanian	
Russian	
Russian	old
Serbian	
Slovak	
Slovenian	
Spanish	
Swedish	
Tamil	
Telugu	
Turkish	
Uighur	
Ukrainian	
Urdu	
Vietnamese	
Welsh	
Wolof	

Technical details

Version	2 Note: UDPipe exists in two versions. Version 2 is a Python prototype (while UDPipe 1 is a user-friendly standalone C++ application with bindings for Python, Java, C#, and Perl). In general, it is meant for research, not as a user-friendly replacement for UDPipe1. It does not perform tokenization by itself: it uses UDPipe1 to do that, or you can select a language-independent generic script. In comparison to UDPipe1, its language models require more computation power (while the UDPipe1 models are small and fast). On the other hand, it mostly performs better than UDPipe1, since the models are neural-network (BERT) based. If you use the R library **udpipe** by J. Wijffels, mind that it operates with UDPipe 1. If you want to take advantage of the UDPipe2 output, you have to call the UDPipe2 API outside of this library (and then you can of course use the resulting CoNLL-U file with other **udpipe** functions).
Version Date:	2022-08-05
Works on Operating Systems:	• Linux (not tested for other operation systems) • Online through a REST API
License:	Fully free (CC BY-NC-SA 4.0)
Distribution:	• Local application (source code at https://github.com/ufal/udpipe/releases/tag/v2.0.0) • Cloud service • Python script to interact with the API: https://github.com/ufal/udpipe/blob/udpipe-2/udpipe2_client.py
Download, installation, documentation, user guides:	• http://hdl.handle.net/11234/1-4816 • Project URL: http://ufal.mff.cuni.cz/udpipe • Demo URL: http://lindat.mff.cuni.cz/services/udpipe/ • Universal Dependencies website: https://universaldependencies.org/ is the information hub for Universal Dependencies: the list of currently available UD treebanks, the logic UD annotation scheme, language-specific UD annotation guidelines, how to contribute your own treebank, commonly used tools to query or annotate UD treebanks, and much more.
User interface:	Demo URL: http://lindat.mff.cuni.cz/services/udpipe/
Docker instance:	docker pull ghcr.io/ufal/udpipe:2.10-all-tf-1.15.4-gpu-staging
How does the tool process your text:	Automatically adds some labels (annotation) to your texts.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.

Plug in your own model:	To train a model for UDPipe2 yourself, you would have to set up a UDPipe 2 instance locally and run the source scripts as described here: https://github.com/ufal/udpipe/tree/udpipe-2. So, theoretically yes, but it is not meant to be performed by users without solid programming background. If you want to train your model yourself before the release for UDPipe 2, you can easily do so with UDPipe 1, although probably with somewhat lower performance than UDPipe2 would achieve. You are very welcome to contribute to the Universal Dependencies Treebanks. Once your treebank is valid and large enough (10,000 tokens), it will be automatically included in the next version of the models for UDPipe 2.
XML-TEI compatibility:	UDPipe 2 does not support XML-TEI at all but there are various tools to combine the CoNLL-U output with TEI.
Required data input format:	The input for UDPipe 2 parser must be UTF-encoded text without any other markup. Structuring options: • CoNLL-U • horizontal (each sentence on a separate line) • vertical (each token on a separate line, with an empty line denoting end of the sentence. For details, see https://lindat.mff.cuni.cz/services/udpipe/api-reference.php and look for Method **process**, parameter **input**.

Which other tools from this list does this tool integrate?

- Universal Dependencies Language Models
- UDPipe 1 (it uses its tokenizer by default)

Related papers

Straka, M. (2018). UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 197–207. https://doi.org/10.18653/v1/K18-2020

Agić, Ž., Aranzabe, M., Atutxa, A., Bosco, C., Choi, J., Marneffe, M.-C. de, Dozat, T., Farkas, R., Foster, J., Ginter, F., Goenaga, I., Gojenola, K., Goldberg, Y., Hajič, J., Johannsen, A., Kanerva, J., Kuokkala, J., Laippala, V., Lenci, A., … Zeman, D. (2015). *Universal Dependencies 1.1*. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Straková, J., Straka, M., & Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 13–18.

Universal Dependencies Models for UDPipe

UDPipe language models

Quick description	NLP
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Features: Tokens; Sentence borders; Paragraphs
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Give the basic dictionary form of a token
Method it uses for that task	Lemmatization
Features (text enrichment)	Lemmas, aka dictionary forms of tokens
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Determine part of speech and relevant morphological categories
Method it uses for that task	Determine part of speech and relevant morphological categories Morphological tagging
Features (text enrichment)	Parts of speech; Morphological categories
Metric	
Formalism	
Tagset	Universal Dependencies
Quick description	NLP
Task it solves	Determine syntactic relations between tokens (e. g. verb subjects)
Method it uses for that task	Syntactic parsing
Features (text enrichment)	Syntactic relations between tokens within one sentence
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies

What can this tool do for you?

This is a package of language models for the UDPipe parser and tagger. It is updated after each update of the Universal Dependencies treebank collection at universaldependencies.org. Depending on the changes in the treebank collection, the update may provide more training data for already existing language models or even new language models.
UDPipe is a state-of-the art system. This means that it is able to learn from less data than older systems. However, the truth is, still, that the quality of the analysis depends on the size of training data related to the complexity of the language. Hence, e. g. languages with rich inflection or very specific syntax need bigger training data. Besides, the UDPipe developers do normally not actively look for external resources for specific languages, such as morphological lexicons. There are known cases of parsers optimized for one given language with bigger data and additional resources, which perform better than UDPipe, e. g. the Magyarlanc parser for Hungarian.
When considering several parsers for your language, have a look at the evaluation of the UDPipe performance with the models from a given relase here:
https://ufal.mff.cuni.cz/udpipe/2/models

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Afrikaans	
Arabic	
Armenian	
Armenian	Western
Basque	
Belarusian	
Bulgarian	
Catalan, Valencian	
Chinese	
Chinese	literary
Coptic	
Croatian	
Czech	
Danish	
Dutch (not Flemish)	
Estonian	
English	
English	Nigerian Pidgin
Estonian	
Faroese	
Finnish	
French	
French	old (842-ca. 1400)
Gaelic	Scottish
Galician	
Gambian/Wolof	
German	

Gothic	
Greek	ancient (to 1453)
Greek	modern (since 1453)
Hebrew	ancient
Hebrew	modern
Hindi	
Hungarian	
Icelandic	
Indonesian	
Irish	
Italian	
Japanese	
Korean	
Latin	
Latvian	
Lithuanian	
Maltese	
Marathi	
Sami	Northern
Norwegian	Nynorsk
Norwegian	Bokmål
Old Church Slavonic	
Persian	
Polish	
Portuguese	
Romanian	
Russian	
Russian	old
Serbian	
Slovak	
Slovenian	
Spanish	
Swedish	
Tamil	
Telugu	
Turkish	
Uighur	
Ukrainian	
Urdu	
Vietnamese	
Welsh	
Wolof	

Technical details

Version	2.10. These models form an eco-system with Universal Dependencies treebanks (Agić et al., 2015). The Universal Dependencies Treebanks are regularly updated twice a year, and so are the models based on them
Version Date:	2022-07-11

Works on Operating Systems:	Windows, Unix, MacOs, Linux
License:	Fully free (CC BY-NC-SA 4.0)
Distribution:	a zipped file
Download, installation, documentation, user guides:	• http://hdl.handle.net/11234/1-4804 • Project URL: http://ufal.mff.cuni.cz/udpipe • Demo URL: http://lindat.mff.cuni.cz/services/udpipe/ • To use these models, you have to install the tool UDPipe binary version at least 1.2, which you can download from http://ufal.mff.cuni.cz/udpipe • User's manual to UDPipe 1: https://ufal.mff.cuni.cz/udpipe/1/users-manual#run_udpipe_input • Universal Dependencies website: https://universaldependencies.org/ is the information hub for Universal Dependencies: the list of currently available UD treebanks, the logic UD annotation scheme, language-specific UD annotation guidelines, how to contribute your own treebank, commonly used tools to query or annotate UD treebanks, and much more.
User interface:	None
Docker instance:	No or not known
How does the tool process your text:	• When plugged in the UDPipe parser, it makes it automatically add some labels (annotation) to your texts.
Tool exports results:	
Statistical models:	This tool is a statistical model / a set of statistical models.
Plug in your own model:	You cannot easily train your statistical model to plug in this tool. It is possible, but rather complex for a user without solid programming background. However, you are very welcome to contribute to the Universal Dependencies Treebanks. Once your treebank is valid and large enough (min. 10,000 tokens), it will be automatically included in the next version of the models for the current UDPipe parser (UDPipe 2.0). If you want to train your model yourself before the release for UDPipe2, you can easily do so with UDPipe 1.0, with probably somewhat lower performance than UDPipe2 would achieve. To train a model for UDPipe2 yourself, you would have to set up a UDPipe instance locally and run the source scripts as described here: https://github.com/ufal/udpipe/tree/udpipe-2.
XML-TEI compatibility:	This tool does not support XML-TEI at all.

Required data input format:	-

Which other tools from this list does this tool integrate?

This tool is integrated into the UDPipe2 parser.

Related papers

Agić, Ž., Aranzabe, M., Atutxa, A., Bosco, C., Choi, J., Marneffe, M.-C. de, Dozat, T., Farkas, R., Foster, J., Ginter, F., Goenaga, I., Gojenola, K., Goldberg, Y., Hajič, J., Johannsen, A., Kanerva, J., Kuokkala, J., Laippala, V., Lenci, A., … Zeman, D. (2015). *Universal Dependencies 1.1*. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Straková, J., Straka, M., & Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 13–18.

udpipe

UDPipe R library

Quick description	**NLP**
Task it solves	Recognize borders of tokens and sentences
Method it uses for that task	Tokenization & Sentence splitting
Features (text enrichment)	Features: Tokens; Sentence borders; Paragraphs
Metric	
Formalism	
Tagset	
Quick description	**NLP**
Task it solves	Give the basic dictionary form of a token
Method it uses for that task	Lemmatization
Features (text enrichment)	Lemmas, aka dictionary forms of tokens
Metric	
Formalism	
Tagset	
Quick description	**NLP**
Task it solves	Determine part of speech and relevant morphological categories
Method it uses for that task	Determine part of speech and relevant morphological categories Morphological tagging
Features (text enrichment)	Parts of speech; Morphological categories
Metric	
Formalism	
Tagset	Universal Dependencies
Quick description	**NLP**
Task it solves	Determine syntactic relations between tokens (e. g. verb subjects)
Method it uses for that task	Syntactic parsing
Features (text enrichment)	Syntactic relations between tokens within one sentence
Metric	
Formalism	Universal Dependencies
Tagset	Universal Dependencies
Quick description	**Corpus analysis**
Task it solves	Count frequencies of words, n-grams, dependency pairs of parent and child, co-occurrence analysis, collocation analysis
Method it uses for that task	various statistic metrics, extraction of text snippets according to these metrics

Features (text enrichment)	
Metric	
Formalism	
Tagset	
Quick description	**NLP**
Task it solves	Train a UDPipe 1 model to use with this library, using your own annotated data
Method it uses for that task	Statistical machine learning
Features (text enrichment)	
Metric	Probabilities of different tags with different words for a language (domain)
Formalism	Universal Dependencies
Tagset	Universal Dependencies

What can this tool do for you?

This tool is an R library that allows for using the UDPipe 1 parser directly within R. Besides, it provides a range of handy functions for text mining, such as basic frequency statistics, collocation analysis, key word extraction, and topic modeling with Latent Dirichlet Allocation as well as Latent Semantic Analysis.

Which languages can it work with (as of February 2023)?

Note that this tool uses the UDPipe 1 parser and hence can only use models up to version 2.5, which is the last pre-trained model version for UDPipe1. It contains fewer languages than more recent models trained for UDPipe2, which you cannot use through this library. Also, some training corpora for the represented languages may be smaller.
The parsers itself is language-agnostic. The list below shows the pre-trained models.

Language	Variety (geographical, or temporal if not modern)
Afrikaans	
Arabic	
Armenian	
Basque	
Belarusian	
Bulgarian	
Catalan, Valencian	
Chinese	
Chinese	literary
Coptic	
Croatian	
Czech	
Danish	
Dutch (not Flemish)	
Estonian	
English	

Estonian	
Finnish	
French	
French	old (842-ca. 1400)
Gaelic	Scottish
Galician	
German	
Gothic	
Greek	ancient (to 1453)
Greek	modern (since 1453)
Hebrew	ancient
Hebrew	modern
Hindi	
Hungarian	
Irish	
Italian	
Japanese	
Korean	
Latin	
Latvian	
Lithuanian	
Maltese	
Marathi	
Norwegian	Nynorsk
Norwegian	Bokmål
Old Church Slavonic	
Persian	
Polish	
Portuguese	
Romanian	
Russian	
Russian	old
Serbian	
Slovak	
Slovenian	
Spanish	
Swedish	
Tamil	
Telugu	
Turkish	
Uighur	
Ukrainian	
Urdu	
Vietnamese	
Wolof	

Technical details

Version	0.8.11
Version Date:	2023-01-06
Works on Operating Systems:	Windows, Unix, Linux, MacOs
License:	Fully free (MPL-2.0)
Distribution:	
Download, installation, documentation, user guides:	https://cran.r-project.org/web/packages/udpipe/index.html https://cran.r-project.org/web/packages/udpipe/udpipe.pdf https://bnosac.github.io/udpipe/en/index.html https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-usecase-topicmodelling.html https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-usecase-postagging-lemmatisation.html
User interface:	None
Docker instance:	No
How does the tool process your text:	It runs the UDPipe 1 parser on your text. In addition, you can analyze keywords, collocations, and topics of any text in the CoNLL-U format. That is, you can parse something with UDPipe 2 outside this library directly through the UDPipe2 REST API (e.g. using the RCurl library in R) and then continue with this udpipe library to analyze the CoNLL-U file.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does include statistical models.
Plug in your own model:	You can easily train your statistical model to plug in this tool.
XML-TEI compatibility:	This tool does not support XML-TEI at all.
Required data input format:	Plain text

Which other tools from this list does this tool integrate?

UDPipe1, UDPipe Language Models

Related papers

Straka, M. (2018). UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 197–207. https://doi.org/10.18653/v1/K18-2020

Agić, Ž., Aranzabe, M., Atutxa, A., Bosco, C., Choi, J., Marneffe, M.-C. de, Dozat, T., Farkas, R., Foster, J., Ginter, F., Goenaga, I., Gojenola, K., Goldberg, Y., Hajič, J., Johannsen, A., Kanerva, J., Kuokkala, J., Laippala, V., Lenci, A., … Zeman, D. (2015). *Universal*

Dependencies 1.1. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Straková, J., Straka, M., & Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 13–18.

POETRY PROCESSING TOOLS

Alberti

Alberti Multilingual Model

What can this tool do for you?

This is a language model trained with poetic texts that returns word embeddings. It has been trained for the word masking task, but it fills the masked token by trying ot be poetic rather than the most probable one. For example, for the sentence “I went to the library to buy a <MASK>” it would return “treasure” instead of “book”.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
English	
French	
Italian	
Czech	
Portuguese	
Spanish	
Arabic	
Chinese	
Finnish	
German	
Hungarian	
Russian	

Technical details

Version	1.0
Version Date:	2022
Works on Operating Systems:	Windows, Unix, MacOs
License:	Fully free (CC-BY-4.0)
Distribution:	Language model (transformers)
Download, installation, documentation, user guides:	https://huggingface.co/flax-community/alberti-bert-base-multilingual-cased
User interface:	none
Docker instance:	No
How does the tool process your text:	It returns word-embeddings
Tool exports results:	No
Statistical models:	This tool is a statistical model / a set of statistical models (transformer)

Plug in your own model:	Yes
XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	--

Recommended tutorials

Title	Provider/Author	URL
Readme	Postdata	https://huggingface.co/flax-community/alberti-bert-base-multilingual-cased
Demo	Postdata	https://huggingface.co/spaces/flax-community/alberti

Related papers

Pérez Pozo, Á., de la Rosa, J., Ros, S., González-Blanco, E., Hernández, L., & de Sisto, M. (2022). A bridge too far for artificial intelligence?: Automatic classification of stanzas in Spanish poetry. *Journal of the Association for Information Science and Technology*, *73*(2), 258–267. https://doi.org/10.1002/asi.24532

Alberti-Stanzas

Quick description	NLP
Task it solves	Automatic annotation of stanzas
Method it uses for that task	Word embeddings
Features (text enrichment)	Stanza type
Metric	
Formalism	
Tagset	

What can this tool do for you?

This is a language model trained with poetic texts that returns word embeddings. It has been trained for the stanzas detection task port Spanish poetry. It can return the stanza type of a Spanish poem.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Spanish	

Technical details

Version	Unknown
Version Date:	2022
Works on Operating Systems:	Windows, Unix, MacOs
License:	Fully free (CC-BY-4.0)
Distribution:	Language model (transformers)
Download, installation, documentation, user guides:	https://huggingface.co/alvp/alberti-stanzas
User interface:	Yes
Docker instance:	Yes
How does the tool process your text:	Automatic annotation of stanza type for Spanish poems
Tool exports results:	No
Statistical models:	This tool is a statistical model / a set of statistical models (a fine-tuned transformers model)
Plug in your own model:	You can easily train your statistical model to plug in this tool.
XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	Plain text

Recommended tutorials

Title	Provider/Author	URL
UI/Docker	Postdata	https://github.com/linhd-postdata/alberti-stanzas-streamlit
Metrics and instructions	Postdata	https://github.com/linhd-postdata/alberti-stanzas-api
Model	Postdata	https://huggingface.co/alvp/alberti-stanzas

Related papers

Pérez Pozo, Á., de la Rosa, J., Ros, S., González-Blanco, E., Hernández, L., & de Sisto, M. (2022). A bridge too far for artificial intelligence?: Automatic classification of stanzas in Spanish poetry. *Journal of the Association for Information Science and Technology*, *73*(2), 258–267. https://doi.org/10.1002/asi.24532

Averell

Quick description	NLP
Task it solves	Automatic poetic annotation for Spanish poetry (rhyme, stanza type, metric)
Method it uses for that task	Syllabify, tokenization & Sentence splitting, stanza detection.
Features (text enrichment)	Tokens, syllables, Sentence borders, stanzas, rhymes
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Download one or several corpora and export to a common format
Method it uses for that task	Text parsing
Features (text enrichment)	
Metric	
Formalism	
Tagset	

What can this tool do for you?

Averell is a python library and command line interface that facilitates working with existing repositories of annotated poetry. Averell is able to download an annotated corpus and reconcile different TEI entities to provide a unified JSON output at the desired granularity. That is, for their investigations some researchers might need the entire poem, poems split line by line, or even word by word if that is available. Averell allows to specify the granularity of the final generated dataset, which is a combined JSON with all the entities in the selected corpora. Each corpus in the catalog must specify the parser to produce the expected data format.

Which languages can it work with (as of February 2023)?

- For automatic annotation: Spanish
- For automatic integration: any (it is language-agnostic)

Technical details

Version	1.2.2
Version Date:	2021-11-15
Works on Operating Systems:	Windows, Unix, MacOs
License:	Fully free (Apache Software License 2.0)
Distribution:	• Local application

	• Library for Python
Download, installation, documentation, user guides:	https://github.com/linhd-postdata/averell
User interface:	None, command line https://github.com/linhd-postdata/averell
Docker instance:	No, but work in progress
How does the tool process your text:	• Automatically adds some labels (annotation) to your texts. • Downloads an annotated corpus and reconcile different TEI entities to provide a unified JSON output at the desired granularity.
Tool exports results:	Yes. JSON, Ontopoetry RDF, TEI work in progress
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	• The tool requires or accepts text in XML-TEI, adds its markup in XML-TEI, and ensures valid XML-TEI on the output. • The tool requires or accepts text in XML-TEI but the original markup gets lost and/or the output will not be integrated into the original XML-TEI format.
Required data input format:	The input format does not matter but the following information must be given as a bare minimum: • Corpus name • Annotation type (manual, automatic, or none) • Author name • Poem title • Poem text split into stanzas

Recommended tutorials

Title	Provider/Author	URL
Readme, basic usage and installation	Postdata	https://averell.readthedocs.io/en/latest/

Related papers

Díaz, A., de la Rosa, J., Pérez, Á., Lorenzo, L. H., González-Blanco, E., & Ros, S. (2020). *Averell a management tool to transform XML/TEI poetic corporain JSON POSTDATA ontology compliant.* Zenodo. https://doi.org/10.5281/zenodo.5704837

Horace

Quick description	NLP
Task it solves	Convert rantanplan Json to Ontopoetry RDF
Method it uses for that task	Parsing
Features (text enrichment)	
Metric	
Formalism	
Tagset	

What can this tool do for you?

Format converter from PoetryLab JSON to POSTDATA semantic formats. It receives a scansion dictionary from Rantanplan and outputs an RDFLib Graph object. With it, serialization options become available.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Spanish	

Technical details

Version	Unknown
Version Date:	2021
Works on Operating Systems:	Windows, MacOs, Unix
License:	Fully free (Apache 2.0)
Distribution:	• Local application • Library for Python
Download, installation, documentation, user guides:	https://github.com/linhd-postdata/Horace
User interface:	None
Docker instance:	No
How does the tool process your text:	Automatically adds some labels (annotation) to your texts.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does not include statistical models.
XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	JSON using the rantanplan format

Recommended tutorials

Title	Provider/Author	URL
Documentation	Postdata	https://horace.readthedocs.io/en/latest/
README	Postdata	https://github.com/linhd-postdata/Horace

Related papers

de la Rosa, J., Pérez, Á., Hernández, L., Díaz, A., Ros, S., & González-Blanco, E. (2021). *PoetryLab as Infrastructure for the Analysis of Spanish Poetry*. 75–82. https://doi.org/10.3384/ecp1809

Poetrylab

alias Poetrylab web

Quick description	Automatic annotation
Task it solves	
Method it uses for that task	
Features (text enrichment)	poetic features
Metric	
Formalism	
Tagset	
Quick description	Corpus manager
Task it solves	
Method it uses for that task	
Features (text enrichment)	Visualization of poetic features
Metric	
Formalism	
Quick description	Corpus manager
Task it solves	
Method it uses for that task	
Features (text enrichment)	Full text search
Metric	
Formalism	
Tagset	

What can this tool do for you?

This is a web application show and visualize poetic information for the POSTDATA multilingual corpus. You can search by author, or poem name, or write your own poem, and you can visualize the metrical and structural information.

Which languages can it work with (as of February 2023)?

This tool is language-agnostic (supports any language).

Technical details

Version	Unknown
Version Date:	2022-04-27
Works on Operating Systems:	Windows, Unix, MacOs
License:	Fully free (Apache 2.0)
Distribution:	Local application
Download, installation, documentation, user guides:	https://github.com/linhd-postdata/poetrylab
User interface:	GUI
Docker instance:	Yes
How does the tool process your text:	• Automatically adds some labels (annotation) to your texts. • Allows you to search through your texts and returns matching results.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and but it does not include statistical models.
XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	Plain text

Related papers

de la Rosa, J., Pérez, Á., Hern´andez, L., D´ıaz, A., Ros, S., & Gonz´alez-Blanco, E. (2021). *PoetryLab as Infrastructure for the Analysis of Spanish Poetry*. 75–82. https://doi.org/10.3384/ecp1809

Poetrylab-API

Quick description	NLP
Task it solves	Automatic annotation of Spanish poetry
Method it uses for that task	Tokenization, syllabification
Features (text enrichment)	rhyme, metric, part of speech, stanza type
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Automatic enjambment detection
Method it uses for that task	Rule-based enjambment detection
Features (text enrichment)	enjambment
Metric	
Formalism	
Tagset	

What can this tool do for you?

The PoetryLab API is a REST API that provides an analysis endpoint to extract information about the poems. It can annotate Spanish poetry with rhyme, metric, enjambment, and part of speech.

Only the method POST is allowed at the /analysis endpoint. A playground interface documenting the API can be found at the /ui endpoint.

Which languages can it work with (as of February 2023)?

Language	Variety (geographical, or temporal if not modern)
Spanish	

Technical details

Version	Unknown
Version Date:	2021-09-17
Works on Operating Systems:	Windows, Unix, MacOs
License:	Fully free (Apache 2.0)
Distribution:	• REST API • Library for Python

Download, installation, documentation, user guides:	https://github.com/linhd-postdata/poetrylab-api
User interface:	API-REST
Docker instance:	Yes
How does the tool process your text:	The tool automatically adds some labels (annotation) to your texts.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, and it does include statistical models.
Plug in your own model:	You can easily train your statistical model to plug in this tool.
XML-TEI compatibility:	The tool does not support XML-TEI at all.
Required data input format:	Plain text

Recommended tutorials

Title	Provider/Author	URL
Documentation	Postdata	https://rantanplan.readthedocs.io/en/latest/readme.html
Readme	Postdata	https://github.com/linhd-postdata/poetrylab-api

Related papers

de la Rosa, J., Pérez, Á., Hern´andez, L., D´ıaz, A., Ros, S., & Gonz´alez-Blanco, E. (2021). *PoetryLab as Infrastructure for the Analysis of Spanish Poetry*. 75–82. https://doi.org/10.3384/ecp1809

RhymeTagger

Quick description	NLP
Task it solves	Provide phonetic transcription
Method it uses for that task	
Features (text enrichment)	**IPA strings**
Metric	
Formalism	
Tagset	
Quick description	NLP
Task it solves	Rhyme detection
Method it uses for that task	
Features (text enrichment)	list
Metric	
Formalism	
Tagset	

What can this tool do for you?

A simple collocation-driven **recognition of rhymes**. Contains pre-trained models for **Czech, Dutch, English, French, German, Russian,** and **Spanish** poetry.

Which languages can it work with (as of February 2023)?

The tool is language-agnostic (i. e. supports any language you train your language model for). Currently it has pre-trained models for Czech, Dutch, English, French, German, Russian, and Spanish.

Technical details

Version	0.2.2
Version Date:	2021-03-02
Works on Operating Systems:	Windows, Unix, MacOs, Linux
License:	Fully free
Distribution:	Library for Python
Download, installation, documentation, user guides:	• pip install rhymetagger • https://github.com/versotym/rhymetagger
User interface:	None
Docker instance:	No or not known

How does the tool process your text:	• Returns some metrics as numbers or tables.
Tool exports results:	Yes
Statistical models:	This tool is not a statistical model / a set of statistical models, but it does not include statistical models.
Plug in your own model:	You can easily train your statistical model to plug in this tool.
XML-TEI compatibility:	None
Required data input format:	Flat Python list of verse lines or list of lists (stanzas X lines)

Recommended tutorials

Title	Provider/Author	URL
GitHub readme	Petr Plecháč	https://github.com/versotym/rhymetagger

Related papers

Plecháč, P. (2018). A Collocation-Driven Method of Discovering Rhymes (in Czech, English, and French Poetry). In Taming the Corpus: From Inflection and Lexis to Interpretation (pp. 79–95). Springer. https://doi.org/10.1007/978-3-319-98017-1_5

References

Agić, Ž., Aranzabe, M., Atutxa, A., Bosco, C., Choi, J., Marneffe, M.-C. de, Dozat, T., Farkas, R., Foster, J., Ginter, F., Goenaga, I., Gojenola, K., Goldberg, Y., Hajič, J., Johannsen, A., Kanerva, J., Kuokkala, J., Laippala, V., Lenci, A., Lindén, K., Ljubešić, N., Lynn, T., Manning, C., Alonso, H.M., McDonald, R., Missilä, A., Montemagni, S., Nivre, J., Nurmi, H., Osenova, P., Petrov, S., Piitulainen, J., Plank, B., Prokopidis, P., Pyysalo, S., Seeker, W., Seraji, M., Silveira, N., Simi, M., Simov, K., Smith, A., Tsarfaty, R., Vincze, V., Zeman, D., 2015. Universal Dependencies 1.1. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague, Praha, Czechia.

Bastian, M., Heymann, S., Jacomy, M., 2009. Gephi: An Open Source Software for Exploring and Manipulating Networks. Presented at the International AAAI Conference on Weblogs and Social Media.

Biber, D., 2004. Conversation text types: A multi-dimensional analysis., in: Gérald Purnelle, Cédrick Fairon, Anne Dister (Eds.), Le Poids Des Mots: Proceedings of the 7th International Conference on the Statistical Analysis of Textual Data,. Presented at the 7th International Conference on the Statistical Analysis of Textual Data, Presses universitaires de Louvain, Louvain.

Biber, D., Conrad, S., 2009. Register, Genre, and Style, Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511814358

Bloomfield, L., Hockett, C.F., 1984. Language. University of Chicago Press.

Chomsky, N., Lightfoot, D.W., 2002. Syntactic Structures, De Gruyter Reference Global. Mouton de Gruyter.

Christ, O., 1994. A modular and flexible architecture for an integrated corpus query system, in: Proceedings of COMPLEX'94: 3rd Conference on Computational Lexicography and Text Research. Budapest, Hungary, pp. 23–32.

Dimitrova, L., Erjavec, T., Ide, N., Kaalep, H. J., Petkevic, V., & Tufis, D. (1998). Multext-East: Parallel and Comparable Corpora and Lexicons for Six Central and Eastern European Languages. *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*. https://aclanthology.org/C98-1049

Eder, M., Rybicki, J., Kestemont, M., 2016. Stylometry with R: a package for computational text analysis. R Journal 8, 107–121.

Evert, S., CWB Development Team, 2022. The IMS Open Corpus Workbench (CWB) CQP Interface and Query Language Manual.

Firth, J., 1957. A Synopsis of Linguistic Theory 1930-1955, in: Studies in Linguistic Analysis. Philological Society, Oxford.

Gerdes, K., Hajičová, E., Wanner, L., Press, I., 2013. Computational Dependency Theory, EBL-Schweitzer. IOS Press.

Gréciano, G., Schumacher, H., 2015. Lucien Tesnière - Syntaxe structurale et opérations mentales: Akten des deutsch-französischen Kolloquiums anläßlich der 100. Wiederkehr seines Geburtstages, Strasbourg 1993, Linguistische Arbeiten. De Gruyter.

Halliday, M.A.K., Matthiessen, C.M.I.M., 2013. Halliday's Introduction to Functional Grammar, Contemporary Security Studies. Taylor & Francis.

Hernáez, I., Navas, E., Odriozola, I., Sarasola, K., Diaz de Ilarraza, A., Leturia, I., Diaz de Lezana, A., Beñat Oihartzabal, Jasone Salaberria, 2012. The Basque Language in the Digital Age / Euskara Aro Digitalean, Languages in the European Information Society. The META-NET Language White Paper series. Springer, Berlin.

Hjelmslev, L., Whitfield, F.J., 1961. Prolegomena to a Theory of Language, Prolegomena to a Theory of Language. University of Wisconsin Press.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R., 2006. OntoNotes: the 90% solution, in: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, NAACL-Short '06. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 57–60.

Kermes, H., Evert, S., 2001. Exploiting large corpora: A circular process of partial syntactic analysis, corpus query and extraction of lexikographic information, in: Rayson, P., Wilson, A., McEnery, T., Hardie, A., Khoja, S. (Eds.), Proceedings of the Corpus Linguistics 2001 Conference. Lancaster, pp. 332–340.

Kučera, H., Francis, W.N., 1967. Computational Analysis of Present-day American English. Brown University Press.

Lavid, J., 2010. Towards a science of corpus annotation: a new methodological challenge for Corpus Linguistics. International Journal of Translation 22, 13–36.

Leech, G., Garside, R., Bryant, M., 1994. CLAWS4: The tagging of the British National Corpus, in: Proceedings of the 15th Intl. Conference on Computational Linguistics. Kyoto, Japan, pp. 622–628.

Marcus, M., Kim, G., Marcinkiewicz, M.A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B., 1994. The Penn Treebank: Annotating Predicate-Argument Structure, in: HLT '94: Proceedings of the Workshop on Human Language Technology. Association for Computational Linguistics, Morristown, NJ, USA, pp. 114–119. http://dx.doi.org/10.3115/1075812.1075835

Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., Manning, C.D., 2014. Universal Stanford Dependencies: A cross-linguistic typology. Proceedings of the 9Th International Conference on Language Resources and Evaluation (LREC) 4585–4592.

McCallum, A.K., 2002. MALLET: A Machine Learning for Language Toolkit.

Mel'cuk, I., 2021. Ten Studies in Dependency Syntax, Trends in Linguistics. Studies and Monographs [TiLSM]. De Gruyter.

Montani, I., Honnibal, M., Honnibal, M., Van Landeghem, S., Boyd, A., Peters, H., McCann, P.O., Geovedi, J., O'Regan, J., Samsonov, M., Orosz, G., De Kok, D., Altinok, D., Kristiansen, S.L., Madeesh Kannan, Bournhonesque, R., Lj Miranda, Baumgartner, P., Edward, Explosion Bot, Hudson, R., Mitsch, R., Roman, Fiedler, L., Ryn Daniels, Wannaphong Phatthiyaphaibun, Howard, G., Tamura, Y., Bozek, S., 2023. explosion/spaCy: v3.5.0: New CLI commands, language updates, bug fixes and much more. https://doi.org/10.5281/ZENODO.1212303

Rossum, L.M. van, Šeļa, A., 2022. CLS INFRA D4.1 Skills Gap Analysis. https://doi.org/10.5281/zenodo.6421513

Santorini, B., 1990. Part-of-Speech Tagging Guidelines for the Penn Treebank Project. University of Pennsylvania 3rd Revision 2nd Printing 33.

Straka, M., Straková, J., 2022. UDPipe 2.

Svartvik, J., 2011. Directions in Corpus Linguistics: Proceedings of Nobel Symposium 82 Stockholm, 4-8 August 1991, Trends in Linguistics. Studies and Monographs [TiLSM]. De Gruyter.

Teufel, S., Schmidt, H., Ulrich Heid, Anne Schiller, 1996. EAGLES Study of the relation between Tagsets and Taggers.

Vignaux, G., 2000. The PAROLE Program, in: Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00). European Language Resources Association (ELRA), Athens, Greece.

Zeman, D., 2008. Reusable Tagset Conversion Using Tagset Drivers, in: Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08). European Language Resources Association (ELRA), Marrakech, Morocco.
Zeman, D., et al. 2022. Universal Dependencies 2.11.