Human Language Technologies – The Baltic Perspective
K. Muischnek and K. Müürisep (Eds.)

doi:10.3233/978-1-61499-912-6-183

NLP-PIPE: Latvian NLP Tool Pipeline

Artūrs ZNOTIŅŠ [a,1], Elita CĪRULE [b]
[a] *Institute of Mathematics and Computer Science, University of Latvia*
[b] *Latvian Information Agency LETA*

Abstract. The paper introduces a modular pipeline that allows to combine multiple Natural Language Processing tools into a unified framework. It aims to make NLP technology more accessible for researchers, non-experts and software developers. The paper describes the architecture of NLP-PIPE and presents publicly available NLP components for Latvian.

Keywords. Natural language processing, language resources and tools, Latvian

1. Introduction

Many Natural Language Processing (NLP) applications usually require multiple linguistic processing steps, like tokenization, part of speech tagging (POS), named entity recognition (NER), etc. Generally, NLP tools are focused on specific tasks using different data formats and programming languages. This usually leads to complex installations, configurations, data format transformations and ad-hoc gluing of multiple components. Multiple NLP toolkits try to solve this problem using monolithic architectures that are hard to extend and usually are limited to a single programming language.

NLP-PIPE[2] is a modular NLP component pipeline that makes integration and replacement of separate components easy. This enables specialists to focus on development of specific NLP components and newly created and published components can be used by other researchers or integrated in other applications.

Objectives of NLP-PIPE are:

- Ease of use: Command-line and web-based interface.
- Portable: All components are containerized with *Docker*[3] meaning they will run on any platform with *Docker* installed.
- Modular: Necessary NLP components can be easily picked and changed.
- Scalable: It can utilize multiple CPU cores and can be distributed among multiple machines (additional workers can be added or removed on the go as required).
- Open sourced: NLP-PIPE and trained models are freely available under GNU General Public License v3.0[4].

[1] Corresponding Author: Artūrs Znotiņš, Artificial Intelligence Laboratory, Institute of Mathematics and Computer Science, University of Latvia, Raina blvd. 29, Riga, LV-1459, Latvia; E-mail: arturs.znotins@lumii.lv

[2] https://github.com/LUMII-AILab/nlp-pipe, web-based demo available at: http://nlp.ailab.lv

[3] https://www.docker.com/

[4] https://www.gnu.org/licenses/gpl-3.0.en.html

2. Architecture

NLP-PIPE is a distributed tool pipeline for fast automatic linguistic annotation with minimal configuration and without any installation or compilation.
The architecture is designed to use minimal amount of gluing code to combine NLP tools implemented in different programming languages and different library version requirements.

NLP-PIPE is based on four layers (see Figure 1):

- **Pipeline API**: Software that is aware of all available services and allows user to submit text documents for processing and retrieve results. API provides a base for multiple user clients: command-line tool and web-based API that can further be utilized in a user-friendly web interface with result visualizations.
- **Service**: Software that abstracts processing of a single NLP task. It consists of input and output task queues. Service distributes workload among workers that are attached to this service. Additional workers can be added to the service on the fly without interruption.
- **Worker**: Wrapper that bundles one or more NLP core technologies. Workers process documents from the service input job queue and put results in the corresponding service output queue.
- **Core**: Software that performs a single NLP task like part-of-speech tagging. Cores are created by NLP specialists using programming languages and libraries of their own choice.

Communication between components is supported by *ZeroMQ* [5] distributed messaging library based on TCP sockets. NLP processing inputs, intermediary and final results are represented with a JSON document that is passed through pipeline components and augmented with new task results. JSON document scheme is inspired by KAF [1]. Processing starts with a minimal document containing text and general metadata like job identifier and submission time. Generally, each NLP task further adds a single subdocument describing results of this task and updates metadata describing processing of this particular tool: version, start time and errors. Next pipeline steps can refer to annotations from previous steps, e.g., part-of-speech tagging refers to token identifier from tokenization step. Used JSON document format is more human readable and easier to construct and parse compared to KAF making integration easier and allows to add custom tools not specified in KAF.

NLP-PIPE supports synchronous and asynchronous (batch) processing where large amount of jobs are submitted for further processing and results are saved on a filesystem or in a database. Web-based API allows easy integration of NLP technologies in other client programming languages.

All NLP-PIPE components are containerized using *Docker* that allows to setup the pipeline on every machine (or cluster using *Docker Swarm*) with *Docker* installed by defining a single configuration file containing needed services and their versions. *Docker Hub* [6] provides means of packaged module integration and distribution.

To add a new step to the pipeline it is necessary to create a program that connects to *ZeroMQ* socket to get documents for processing and write results back. Alternately, core software can be left intact by creating a custom worker wrapper (using any programming language) that handles *ZeroMQ* connection and data format transformations using

[5] http://zeromq.org/
[6] https://hub.docker.com/

standard input and output for communication with the core software. Then it is necessary to build a *Docker* image containing all the necessary software dependencies and publish it to *Docker Hub*. Additional benefit of NLP component containerization with *Docker* is that these images can also be used for other purposes (not just in NLP-PIPE) like training of new NLP models on prepared data using just *Docker*.

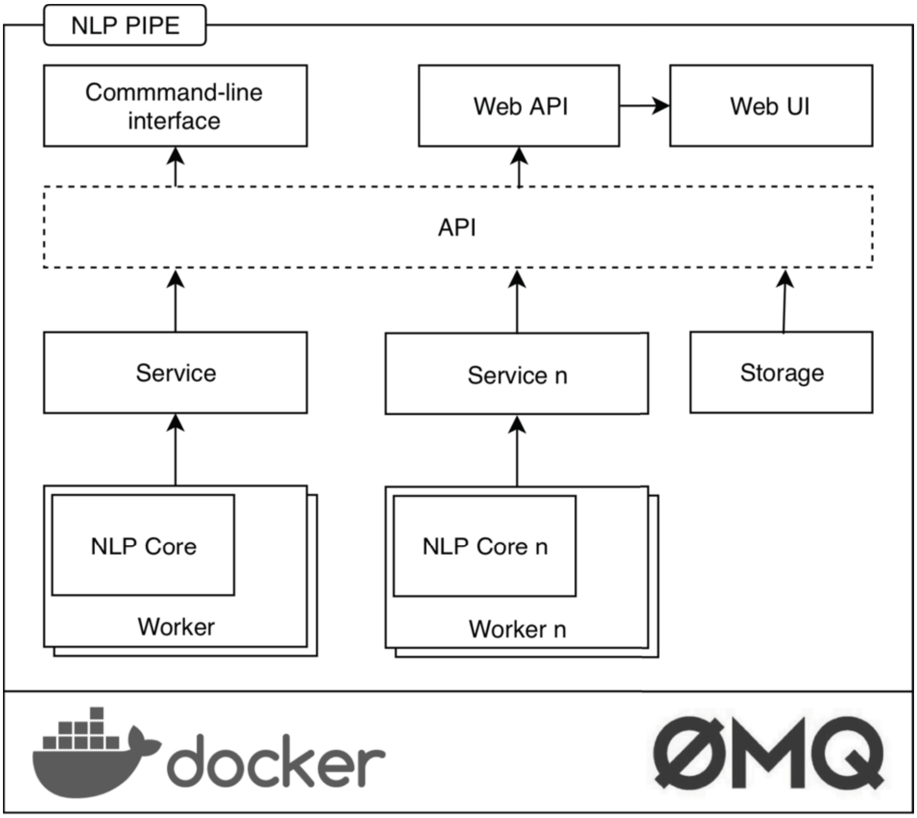


Figure 1. NLP-PIPE architecture.

3. Latvian NLP Components

NLP-PIPE currently provides following automatic annotations for Latvian: tokenization, morphological analysis, dependency parsing, named entity recognition and coreference resolution. Components are based on already available tools, some with improved models and trained on new larger datasets.

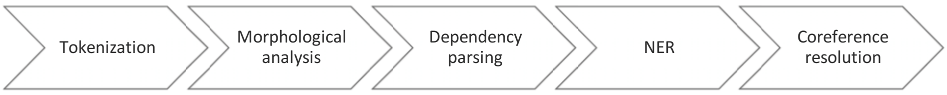


Figure 2. Latvian NLP tool pipeline.

3.1. Tokenizer

This component provides tokenization and sentence segmentation using a fast and simple deterministic finite automaton that is a part of Latvian morphological tagger [2] implemented in *Java*.

3.2. Morphological Analysis

This component provides a statistical morphological tagger which achieves 97.9% accuracy for part of speech recognition and 93.6% for the full morphological feature tag set that includes case, gender, number, person, etc. It is implemented in *Java* based on *CoreNLP*.

3.3. Dependency Parsing

This component provides a continuous transition-based dependency parser based on LSTM using pre-trained word embeddings, learned character and morphological tag embeddings as features. Parser achieves 76.84% LAS (Labelled Attachment Score), 81.24% UAS (Unlabeled Attachment Score) on Latvian Universal Dependencies test set. It is implemented in *Python* and *C++* using *DyNet* library [4], [5].

Table 1. Dependency parsing model results.

Model	**Validation Set**		**Test Set**	
	LAS, %	**UAS, %**	**LAS, %**	**UAS, %**
MORPH_EMB+WORD_EMB	**77.6**	**82.0**	**76.8**	**81.2**
MORPH_EMB	76.6	81.0	75.3	80.2
WORD_EMB	67.5	74.2	65.2	72.5

3.4. Named Entity Recognition

This component provides automatic tagging of following set of named entity categories: *person, organization geopolitical entity, location, product, time* and *event*. NER tagger is based on bidirectional LSTM neural network with additional CRF layer and utilizes pre-trained word embeddings, learned character and word shape embeddings as features. Model is trained on [3] achieving 74.01% F1-score. Tagger is implemented in *Python* using *Keras* framework based on [5], [6].

Table 2. NER model results evaluated using 5-fold cross-validation.

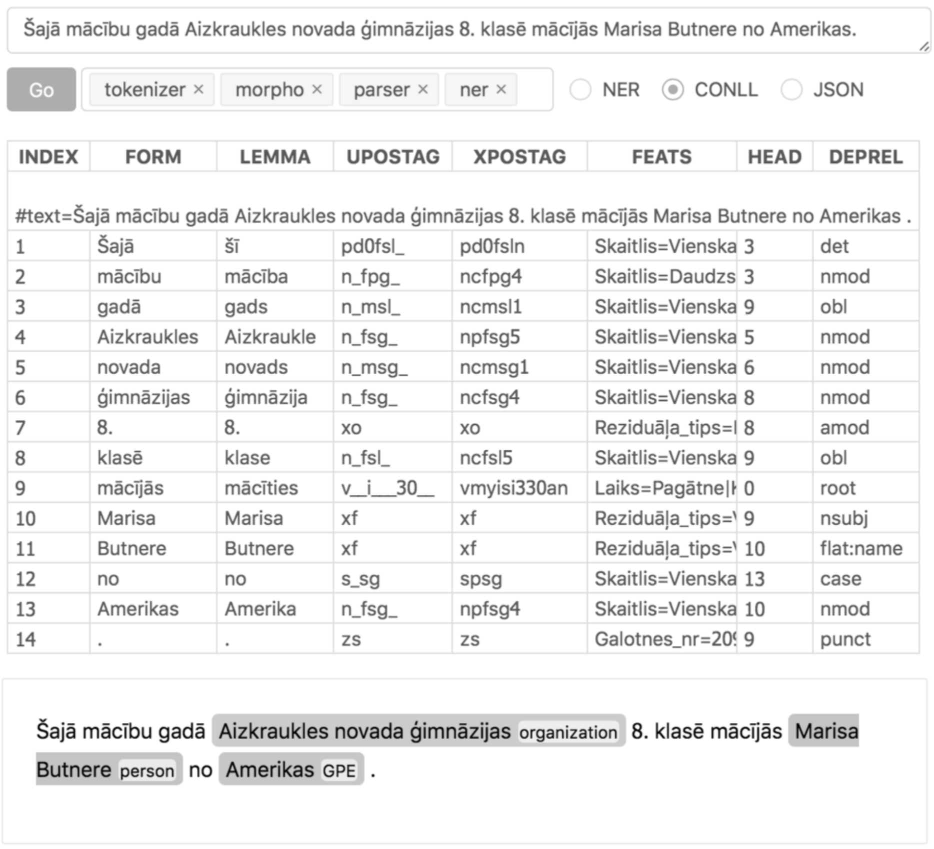
Model	**F1, %**
LSTM+CHAR_EMB+CRF+DROPOUT	**74.0 ± 0.6**
LSTM+CHAR_EMB +CRF	73.2 ± 0.5
LSTM+CHAR_EMB	70.4 ± 0.8
LSTM+CRF	72.6 ± 1.4

Table 3. NER results by category.

Category	F1, %	Precision, %	Recall, %	Count
GPE	79.0	79.2	78.7	449
event	20.0	37.5	13.6	47
location	45.1	40.5	50.9	134
organization	78.5	73.7	84.0	615
person	85.2	82.2	88.4	808
product	40.0	37.5	42.9	50
time	71.7	71.0	72.5	241

3.5. Coreference Resolution

Component contains a rule based coreference resolution system that achieves 58% F1-score [7]. It is implemented in Java.


Šajā mācību gadā Aizkraukles novada ģimnāzijas 8. klasē mācījās Marisa Butnere no Amerikas.

Go | tokenizer × | morpho × | parser × | ner × | NER [✓ CONLL] JSON

INDEX	FORM	LEMMA	UPOSTAG	XPOSTAG	FEATS	HEAD	DEPREL
#text=Šajā mācību gadā Aizkraukles novada ģimnāzijas 8. klasē mācījās Marisa Butnere no Amerikas .							
1	Šajā	šī	pd0fsl_	pd0fsln	Skaitlis=Vienska	3	det
2	mācību	mācība	n_fpg_	ncfpg4	Skaitlis=Daudzs	3	nmod
3	gadā	gads	n_msl_	ncmsl1	Skaitlis=Vienska	9	obl
4	Aizkraukles	Aizkraukle	n_fsg_	npfsg5	Skaitlis=Vienska	5	nmod
5	novada	novads	n_msg_	ncmsg1	Skaitlis=Vienska	6	nmod
6	ģimnāzijas	ģimnāzija	n_fsg_	ncfsg4	Skaitlis=Vienska	8	nmod
7	8.	8.	xo	xo	Reziduāļa_tips=I	8	amod
8	klasē	klase	n_fsl_	ncfsl5	Skaitlis=Vienska	9	obl
9	mācījās	mācīties	v__i__30__	vmyisi330an	Laiks=Pagātne|k	0	root
10	Marisa	Marisa	xf	xf	Reziduāļa_tips=\	9	nsubj
11	Butnere	Butnere	xf	xf	Reziduāļa_tips=\	10	flat:name
12	no	no	s_sg	spsg	Skaitlis=Vienska	13	case
13	Amerikas	Amerika	n_fsg_	npfsg4	Skaitlis=Vienska	10	nmod
14	.	.	zs	zs	Galotnes_nr=20	9	punct

Šajā mācību gadā Aizkraukles novada ģimnāzijas organization 8. klasē mācījās Marisa Butnere person no Amerikas GPE .

Figure 3. Screenshot of web-based user interface where user can input text and get results in CONLL format or text with marked named entities.

4. Related Work

Multiple NLP frameworks and toolkits have been created and each of them uses a different approach to achieve interoperability among all their components. Among the most notable ones are: *GATE* [8] (could be used to manage distributed NLP components, but it is a large and complex system) *UIMA* [9], *Freeling* [10] and *CoreNLP*[7] and *spaCy*[8] (all are monolithic architectures that are generally limited to a single programming language).
Taenga (code not available) [11] and *OpeNER*[9] [12] are the closest ones to what NLP-PIPE is trying to achieve. *OpeNER* makes it quite hard to wrap of new NLP components, it uses quite complex *KAF* (XML based) data format and scalability is achieved through paid *Amazon Web Services*[10].

5. Conclusion and Future Work

NLP-PIPE provides a simple and scalable NLP pipeline publicly available under GNU General Public License v3.0 license. Modularity and containerization with *Docker* makes it easy to setup and offers a lot of flexibility to pick and change different pipeline components and their versions as necessary and to add new linguistic annotators. Currently NLP-PIPE is focused on linguistic annotation for Latvian but it can be used in a multilingual setting for other languages with already available NLP components.

NLP-PIPE includes state-of-the-art NLP components for Latvian including new Universal Dependency parser and Named Entity Recognizer that are trained on the Latvian Multilayer Corpus for NLU (in progress) dataset [3].

It has been proven useful for ad-hoc text analysis and larger batch tasks, e.g., person mention extraction from archive of photo descriptions and automatic text annotation for further manual selection and post-edition for new corpus creation.

Acknowledgements

This work has received financial support from the European Regional Development Fund under the grant agreement No. 1.1.1.1/16/A/219 ("Full Stack of Language Resources for Natural Language Understanding and Generation in Latvian"). The initial research leading to these results has received funding from the research project "Competence Centre of Information and Communication Technologies" of the EU Structural Funds, Contract No. 1.2.1.1/16/A/007 signed between IT Competence Centre and Central Finance and Contracting Agency, Research No. 2.5 "Automated visual material recognition and annotation system for LETA archive".

[7] http://nlp.stanford.edu/software/corenlp.shtml
[8] https://spacy.io/
[9] http://www.opener-project.eu/
[10] https://aws.amazon.com/

References

[1] Wauter Bosma, Piek Vossen, Aitor Soroa, German Rigau, Maurizio Tesconi, Andrea Marchetti, Monica Monachini, and Carlo Aliprandi, Kaf: A Feneric Semantic Annotation Format, Proceedings of the GL2009 Workshop on Semantic Annotation, 2009.

[2] Pēteris Paikens, Laura Rituma and Lauma Pretkalniņa, Morphological analysis with limited resources: Latvian example, Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013), 2013.

[3] Normunds Gruzitis, Lauma Pretkalnina, Baiba Saulite, Laura Rituma, Gunta Nespore-Berzkalne, Arturs Znotins and Peteris Paikens, Creation of a Balanced State-of-the-Art Multilayer Corpus for NLU, *Proceedings of the 11th International Conference on Language Resources and Evaluation*, 2018.

[4] Miguel Ballesteros, Chris Dyer and Noah A. Smith, Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs, *Proceedings of EMNLP 2015*, 2015.

[5] Arturs Znotins, Word embeddings for Latvian natural language processing tools, *Human Language Technologies -- The Baltic Perspective,* 2016.

[6] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami and Chris Dyer, Neural Architectures for Named Entity Recognition, *Proceedings of NAACL 2016*, 2016.

[7] Arturs Znotins, Coreference resolution in Latvian, *Human Language Technologies -- The Baltic Perspective,* 2014.

[8] Hamish Cunningham, Diana Maynard, Kalina Bontcheva and Valentin Tablan, Gate: An Architecture for Development of Robust HLT Applications, *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics,* 2002.

[9] David Ferrucci and Adam Lally, UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment, *Nat. Lang. Eng*., 2004.

[10] Lluís Padró and Evgeny Stanilovsky, Freeling 3.0: Towards Wider Multilinguality, *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, 2012.

[11] Housam Ziad, John Philip McCrae and Paul Buitelaar, Teanga: A Linked Data based platform for Natural Language Processing, Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[12] Rodrigo Agerri, Josu Bermudez, German Rigau, IXA pipeline: Efficient and Ready to Use Multilingual NLP tools, Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014), 2014.