LaMachine: A meta-distribution for NLP software

Maarten van Gompel and Iris Hendrickx
Centre for Language and Speech Technology (CLST)
Radboud University, Nijmegen, the Netherlands
proycon@anaproy.nl, i.hendrickx@let.ru.nl
https://proycon.github.io/LaMachine

Abstract

We introduce LaMachine, a unified Natural Language Processing (NLP) open-source software distribution to facilitate the installation and deployment of a large amount of software projects that have been developed in the scope of the CLARIN-NL project and its current successor CLARIAH. Special attention is paid to encouragement of good software development practices and reuse of established infrastructure in the scientific and open-source software development community. We illustrate the usage of LaMachine in an exploratory text mining project at the Dutch Health Inspectorate where LaMachine was applied to create a research environment for automatic text analysis for health care quality monitoring.

1 Introduction

Software is a key deliverable and a vital component for research in projects such as those under the CLARIN umbrella. It is software that provides researchers the instruments to yield for their research; It is CLARIN's core mission to make digital language resources, including software, available to the wider research community.

We see that NLP software often takes on complex forms such as processing pipelines invoking various individual components, which in turn rely on various dependencies. Add dedicated web-interfaces on top of that and you obtain a suite of interconnected software that is often non-trivial to install, configure, and deploy. This is where LaMachine comes in.

LaMachine incorporates software providing different types of interfaces[1] that typically address different audiences. Whilst we attempt to accommodate both technical[2] and less-technical audiences[3], there is a natural bias towards the former as lower-level interfaces are often a prerequisite to build higher-level interfaces on. Depending on the *flavour* of LaMachine chosen, it makes a good virtual research environment for a data scientist, whether on a personal computer or on a computing cluster, a good development environment for a developer or a good deployment method for production servers in for example CLARIN centres. We demonstrate how LaMachine can create a fully functioning and standalone research environment for text mining and NLP for Dutch texts in a use case project at the Healthcare Inspectorate.

2 Architecture

Being an open-source NLP software distribution, LaMachine is constrained to Unix-like platforms; this primarily means Linux, but also BSD and, with some restrictions, macOS. Cygwin[4] is not tested or supported. However, virtualisation technology enables deployment on a wider range of platforms, including Windows. The focus of the LaMachine distribution stands in contrast with mobile platforms (Android/iOS/etc), native Windows/mac desktop software, or certain interface types in general such as classical desktop GUI applications or mobile 'apps', all of which fall beyond our scope.

[1] Command line interfaces, programming interfaces, web-user interfaces, webservices.
[2] Data scientists, DevOps, system administrators, developers.
[3] The wider researcher community, particularly the Humanities; also educational settings.
[4] A unix environment on Windows

All software that is incorporated in LaMachine must 1) bear some relevance to NLP, 2) be under a recognised open-source license, 3) be deposited in a public version controlled repository[5] and 4) have a release protocol (with semantic versioning) using the proper technology-specific channels.

LaMachine is a *meta distribution* as it can be installed in various contexts. At its core, LaMachine consists of a set of machine-parsable instructions on how to obtain, build (e.g. compile from source), install and configure software. These are implemented using Ansible[6]. This is notably different from the more classical notion of Linux distributions, which generally provide their own repositories with (often binary) software packages. LaMachine builds on this already established infrastructure by taking these repositories as a given and only needs to know which repositories to use. Similarly, there are different programming-language-specific ecosystems providing their own repositories, such as the Python Package Index[7] for Python, CRAN[8] for R, CPAN[9] for Perl, Maven Central[10] for Java. LaMachine again relies on those to pull and install software from and never forks, archives, or modifies the software in any way. In doing so, we compel participating software projects to adhere to well-established distribution standards and ensure the software is more sustainable towards the future (van Gompel et al., 2016). Moreover, we ensure that LaMachine never becomes a prerequisite for the software but merely a courtesy or convenience.

LaMachine provides ample flexibility that allows it to be deployable in different contexts. First of all there is flexibility with regard to the target platform, where we support several major GNU/Linux distributions (Debian, Ubuntu, CentOS, RedHat Enterprise Linux, Fedora, Arch Linux), as well as macOS (although with more limitations). Second, there is flexibility with regard to the form, where we support *containerisation* through Docker[11], *virtualisation* through Vagrant[12] and VirtualBox[13], direct remote provisioning through Ansible (for production servers), or an installation that is either global to the machine or local in a custom directory for a specific user (using `virtualenv`). Pre-built docker containers and virtual machine images with a limited selection of participating software are regularly uploaded to the Docker Hub and Vagrant Cloud, respectively. The different flavours all offer a different degree of seperation from the host OS, where Virtual Machines are completely virtualised, Docker Containers still share the kernel with the host OS, and the machine-specific installation flavour actually compiles against the machine's distribution itself and thus offers the least amount of overhead.

Installation of LaMachine begins with a single bootstrap command[14]. It may interactively query the users for their software preferences, e.g. the flavour of LaMachine and the set of software to install, which is never static but can be customized by the user. The user may also opt for installing the latest releases, the more experimental development versions of the software, or specific custom versions (to facilitate scientific reproducibility). The bootstrap procedure detects and installs the necessary prerequisites automatically and eventually invokes Ansible to perform the bulk of the work.

LaMachine also aims to harmonise the metadata of all installed software, by converting metadata from upstream repositories, i.e. the repositories where tool providers deposit their software, to a common standard called CodeMeta [15] (Jones et al., 2016; Boettiger, 2017) where possible, or encouraging software developers to provide their codemeta metadata inside their source code repositories and using that directly. This in turn enables other tools to do proper service discovery and provenance logging.

Leveraging this metadata, LaMachine comes with a webserver that offers a portal website with access and overview of all installed tools, including web services and web applications. It also comes with a

[5]e.g. Github, Gitlab, Bitbucket, provided the repository is public
[6]`https://www.ansible.com`
[7]`https://pypi.org`
[8]`https://cran.r-project.org/`
[9]`https://www.cpan.org`
[10]`https://search.maven.org`
[11]`https://www.docker.com`
[12]`https://vagrant.org`
[13]`https://www.virtualbox.org`
[14]See `https://proycon.github.io/LaMachine`
[15]`https://codemeta.github.io/`, described in JSON-LD

Jupyter Lab[16] environment which provides a web-based Integrated Development Environment (IDE) for scripting in Python and R, web-based terminal access, and so-called *notebooks* which mix text, code and data output and have gained great popularity in data science community nowadays.

3 Software

LaMachine exists since May 2015 and has been used extensively ever since by numerous users, in 2018 version 2 was released which was a significant rewrite. LaMachine was initially conceived as the primary means of distribution of the software stack developed at CLST, Radboud University Nijmegen. It therefore includes a lot of our software. A full list of included software goes beyond the scope of this overview; we will merely mention some CLARIN-NL/CLARIAH-funded tools: ucto (a tokeniser), Frog (an NLP suite for Dutch), FoLiA (Format for Linguistic Annotation, with assorted tools), FLAT (a web-based linguistic Annotation tool), PICCL (an OCR and post-OCR correction pipeline) and CLAM. However, this project is not limited to one research group and is open to participation by other software providers, especially those also in CLARIAH and the upcoming CLARIAH PLUS project. We already include some relevant software by the University of Groningen (such as Alpino, a dependency parser) and VU Amsterdam. Moreover, LaMachine incorporates a large number of renowned tools by external international parties, offering most notably a mature Python environment with renowned scientific modules such as scipy, numpy, scikit-learn, matplotlib, nltk, spacy, pytorch, keras, gensim, tensorflow, and many others, but also R, Java and tools such as Stanford CoreNLP and Kaldi.

4 Case study

We participated in a small Dutch national project titled *“ Text mining for Inspection: an exploratory study on automatic analysis of health care complaints”* [17] led by IQhealthcare[18], the scientific centre for healthcare quality of RadboudUMC hospital. This project took place at the Dutch Health Inspectorate and aimed to apply text mining techniques to health care complaints that have been registered at the national contact point for health care (Landelijk Meldpunt Zorg[19]) We investigated the usefulness of text mining to categorise and cluster complaints, to automatically determine the severity of incoming complaints, to extract patterns and to identify risk cases. This project turned out to be a good test case of the applicability and usefulness of LaMachine as a standalone research environment. As the complaint data is highly sensitive, it could not leave the secure servers of the health inspectorate and was stored in an environment without internet access. We needed to bring the software to the data via a shared folder.

We used a virtual machine (VM) image of LaMachine and we ran this 64-bits Linux-based VM inside another VM with Windows Server 2012, provided to us by the health inspectorate for this project, in which we did have administrative rights but no internet access. In terms of hardware we ran on a machine with 8 cores and 32GB internal memory available. LaMachine provided a fully functional research environment and we ran all our experiments within LaMachine. We interacted with LaMachine both through the command line, which offers a standard shell and enables access to all lower-level tools and programming languages; and through the (offline) webbrowser to use the Jupyter Notebook environment.

LaMachine comes with some simple data sharing facilities that allowed us to access the sensitive complaint data via a single shared dataspace between host and the VM. (Note that extensive data search and management functions are deliberately beyond the scope of LaMachine, and left to more high-level tooling.)

We used many of the available tools in LaMachine within this project: Frog for linguistic annotation of the textual content of the complaint and the scikit-learn Python package (Pedregosa et al., 2011) for classification, T-scan (Pander Maat et al., 2014) for feature extraction in the form of text characteristics and colibri-core for n-gram analysis.

[16] `https://jupyter.org/`

[17] project website: `https://www.zonmw.nl/nl/onderzoek-resultaten/kwaliteit-van-zorg/programmas/project-detail/effectief-toezicht/tekstmining-in-het-toezicht-een-explorative-studie-`

[18] `http://www.iqhealthcare.nl/nl/`

[19] `https://www.landelijkmeldpuntzorg.nl`

5 Conclusion & Future work

The recent release of LaMachine v2, which constituted a full rewrite, has opened up LaMachine to outside contribution. Contributor documentation has been written, and at this stage, we greatly welcome external participants to join in. Use cases as the example in section 4 contribute to thorough testing and running of LaMachine in less ideal circumstances such as nested VM constructions and offline usage.

Aside from the incorporation of new relevant software, the main objectives for the future are to provide greater *interoperability* between the included tools through better *high-level interfaces* for the researcher. We see this as a bottom-up process and have now established a firm foundation to build upon. Note that such proposed interfaces, including the current portal application in LaMachine, are always considered separate independent software projects, which may be deployed by/in/for LaMachine, but also in other contexts. LaMachine remains 'just' a software distribution at heart.

Development of LaMachine presently takes place in collaboration with the CLARIAH WP3 Virtual Research Environment (VRE) project[20], which has higher ambitions in accommodating the researcher and connectivity of data and services, and transcends also those of the CLARIN Language Resource Switchboard (Zinn, 2016). An important part of our future focus will therefore be on interoperability with the higher-level tools emerging from the VRE efforts, but also with other parts of the CLARIN infrastructure; single-sign on authentication being a notable example here.

Acknowledgement

This research was funded by NWO CLARIN-NL, CLARIAH and the ZonMw project *Tekstmining in het toezicht: een exploratieve studie naar de automatische verwerking van klachten ingediend bij het Landelijk Meldpunt Zorg*, project number 516004614. We thank all project partners: the Dutch Health Inspectorate, IQhealthcare, and Tim Voets for their valuable contributions and help in the ZonMw project.

References

[Boettiger2017] C. Boettiger. 2017. Generating CodeMeta Metadata for R packages. *The Journal of Open Source Software*, 2:454.

[Jones et al.2016] MB. Jones, C. Boettiger, A. Cabunoc Mayes, A. Smith, P. Slaughter, K. Niemeyer, Y. Gil, M. Fenner, K. Nowak, M. Hahnel, et al. 2016. CodeMeta: an exchange schema for software metadata. *KNB Data Repository*.

[Pander Maat et al.2014] H. Pander Maat, R. Kraf, A. van den Bosch, N. Dekker, M. van Gompel, S. Kleijn, T. Sanders, and K. van der Sloot. 2014. T-scan: a new tool for analyzing dutch text. *Computational Linguistics in the Netherlands Journal*, 4:53–74, 12/2014.

[Pedregosa et al.2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[van Gompel et al.2016] M. van Gompel, J. Noordzij, R. de Valk, and A. Scharnhorst. 2016. Guidelines for Software Quality. CLARIAH Task 54.100.

[Zinn2016] C. Zinn. 2016. The CLARIN Language Resource Switchboard. *Proceedings of the CLARIN Annual Conference. CLARIN ERIC.*

[20] `https://github.com/meertensinstituut/clariah-wp3-vre`