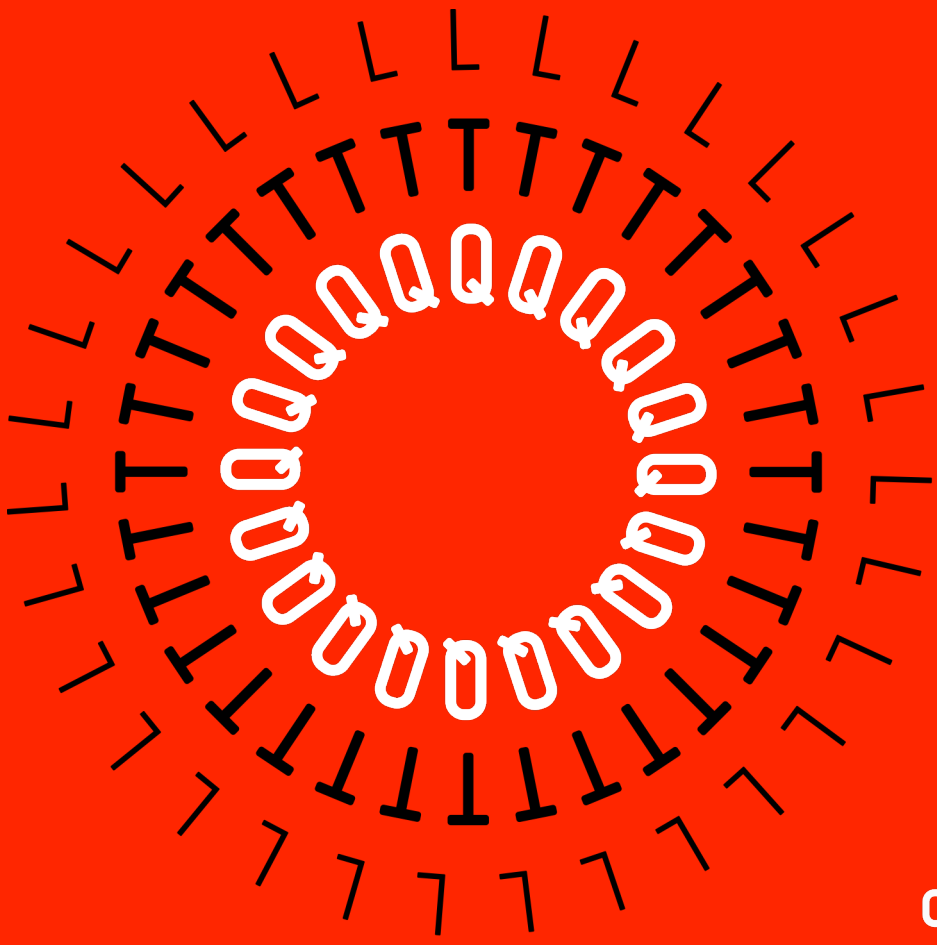


qtleap
quality translation by deep language engineering approaches

REPORT ON THE WORKBENCH FOR DEVELOPERS

DELIVERABLE D3.2

VERSION 1.3 | 2015 JUNE 15

QTLeap

Machine translation is a computational procedure that seeks to provide the translation of utterances from one language into another language.

Research and development around this grand challenge is bringing this technology to a level of maturity that already supports useful practical solutions. It permits to get at least the gist of the utterances being translated, and even to get pretty good results for some language pairs in some focused discourse domains, helping to reduce costs and to improve productivity in international businesses.

There is nevertheless still a way to go for this technology to attain a level of maturity that permits the delivery of quality translation across the board.

The goal of the QTLeap project is to research on and deliver an articulated methodology for machine translation that explores deep language engineering approaches in view of breaking the way to translations of higher quality.

The deeper the processing of utterances the less language-specific differences remain between the representation of the meaning of a given utterance and the meaning representation of its translation. Further chances of success can thus be explored by machine translation systems that are based on deeper semantic engineering approaches.

Deep language processing has its stepping-stone in linguistically principled methods and generalizations. It has been evolving towards supporting realistic applications, namely by embedding more data based solutions, and by exploring new types of datasets recently developed, such as parallel DeepBanks.

This progress is further supported by recent advances in terms of lexical processing. These advances have been made possible by enhanced techniques for referential and conceptual ambiguity resolution, and supported also by new types of datasets recently developed as linked open data.

The project QTLeap explores novel ways for attaining machine translation of higher quality that are opened by a new generation of increasingly sophisticated semantic datasets and by recent advances in deep language processing.

www.qtleap.eu

Funded by

QTLeap is funded by the 7th Framework Programme of the European Commission.

Supported by

And supported by the participating institutions:

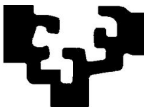
Faculty of Sciences, University of Lisbon

German Research Centre for Artificial Intelligence

Charles University in Prague

Bulgarian Academy of Sciences

Humboldt University of Berlin



University of Basque Country

University of Groningen

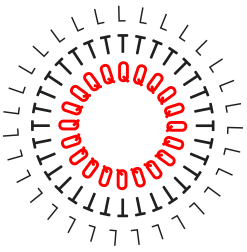
Higher Functions, Lda

Revision History

version	date	author	organisation	description
0.1	2014 APR 4	Eleftherios Avramidis	DFKI	First draft
0.2	2014 APR 5	Aljoscha Burchardt	DFKI	Extended executive summary and minor corrections
0.3	2014 APR 9	António Branco	FCUL	Comments and feedback
1.0	2014 APR 11	Eleftherios Avramidis, Aljoscha Burchardt	DFKI	Text edits; fixed resolution of pipeline graph
1.1	2014 APR 30	Eleftherios Avramidis	DFKI	Integrate internal review, finalize
	2014 September	Aljoscha Burchardt, Elefherios Avramidis	DFKI	Added annex updating on the use of pipelines using input from partners; deliverable approved by all partners.
1.2	2015 June 15	Aljoscha Burchardt, Elefherios Avramidis	DFKI	Documentation of workbench Version 2 and updates after review report
1.3	2015 June 15	Eleftherios Avramidis	DFKI	Incorporated internal review comments

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.



REPORT ON THE WORKBENCH FOR DEVELOPERS

DOCUMENT QTLEAP-2015-D3.2
EC FP7 PROJECT #610516

DELIVERABLE D3.2

completion

FINAL

status

SUBMITTED

dissemination level

PUBLIC

responsible

ALJOSCHA BURCHARDT (WP3 COORDINATOR)

reviewer

MARTIN POPEL

contributing partners

DFKI, FCUL, CUNI, IICT-BAS, UPV/EHU, UG, HF

authors

ALJOSCHA BURCHARDT, ELEFTHERIOS AVRAMIDIS

Contents

1 Executive Summary

The goal of the QTLeap project is to develop Machine Translation (MT) technology that employs innovative methods and knowledge sources in order to operate more and more "in depth" through a number of MT pilots. The project consortium consists of partners which already have a good experience in the fields of MT and deep processing and in using software developing tools like version control, workbench, data storage, and pipeline systems.

As a result of language specificities and prior expertise, significant amount of engineering work will be committed locally using proven tools and workflows. Yet, it is clear that all advances on the project goal depend on joining the decentralized efforts in an effective way, that allows all project members to have better overview on the research progress by other members and interchange ideas, methods and solutions that may match common needs and lead altogether to the common goal.

Task 3.1 has a key role in this direction, as it sets a backbone for common engineering work across all project partners. The current Deliverable documents the second part of Milestone B "Baselines and pipelines". It is closely related to D2.2 "Report on evaluation metrics and baselines for the project" (due M6) in that according to the DoW some selected baselines from Task 2.1 will be first integrated as example pipelines. To avoid overlap, we will not elaborate on the baselines themselves in this deliverable.

As described in the DoW, the goal of Task 3.1 is to support the usage of existing tools and to handle marginal adjustments if needed. It will run throughout the whole project and continuously adjust to the needs and requirements posed by the different MT pilots.

The workbench has been developed in two consecutive steps. In Section 0, we present the initial development infrastructure (Workbench Version 1) that has been set, as a result with close communication among the responsible project partners, already since the Kick-off meeting, where the basic engineering aspects were agreed upon. This part of the workbench covers aspects of general development and experimentation used for developing the MT Pilots.

Based on experiences of the consortium, especially in progress tracking and the evaluation of MT Pilot 1, the consortium decided not only to follow the reviewer's suggestion to extend the workbench by including versioning of data, but to extend it by another set of functionalities that supports progress tracking and evaluation in a principled way. In Section 0, we will report on these recent developments that lead to a unified and extended workbench Version 2 that includes all functionalities from Version 1.

2 Workbench Version 1: General Development

2.1 Directions of the "workbench" design

According to our project organization, the research workbench sets the basis for a set of tools and resources that can aid joined work towards the aims of the research. In particular, the workbench has been designed in order to allow developers to:

- work **multi-site**, i.e. all basic code and resources should be available and modifiable (if needed) on all sites where the contributors reside
- enjoy **interoperability**, i.e. permit the combinations of techniques and approaches that may be based on different coding frameworks, architectures, systems, or platforms
- add and replace **datasets**, which is required for the needs of maintaining the data-intensive work of the empirical side of the systems
- integrate processing **components**, in a way that allows the various tools to exchange inputs and outputs
- record **performance** of pipelines and parts, by providing numeric indications on whether the developed solutions confirm the set research hypotheses. This is done by associating the outputs with the respective evaluation interface
- **debug** pipeline functionality and check **robustness** by monitoring the progress of the individual components, their execution and error messages**.**

The above agreed goals led to the development of particular parts for the workbench, which will be described below.

2.2 Multi-site interoperability

In order to confirm multi-site interoperability, the project will base its engineering efforts on a combination of development decisions that are known to have fit well in similar scenarios:

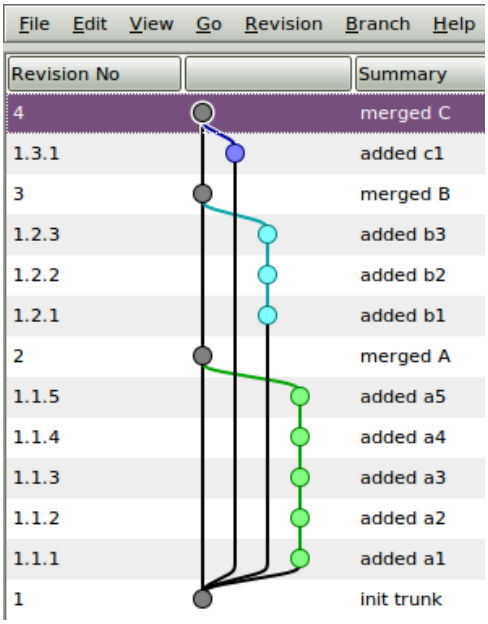


Figure 1: Example of non-linear development by using GIT

2.2.1 Distributed version control source management system

The GIT system has been widely used for storing and organizing the code of the systems, across a wide number of programmers in decentralized locations. We are planning to take advantage of the following aspects:

- **Support for non-linear development**: this allows project partners to produce several modifications of the same component, in parallel. It provides the freedom to proceed with the necessary modification, but also keeps track of the changes. In the end of the development cycle, this allows for merging the individual solutions in a common piece of code (see figure 1).
- Infinite history tracking allows the users to look back on previous versions of the code and go back to some parts of them, if necessary.
- Automatic merging algorithm offers the possibility automatically join parts of code that have been developed by different committers but do not overlap.
- Supported by many integrated development environments (such as Eclipse, NetBeans, etc) which allows the developers to co-operate with each other, but still use locally their own environment preferences and coding tools.

2.2.2 Merging strategy

A merging strategy is necessary, so that it is clear for the programmers when and how to join several parts of code that have been developed individually. Our strategy will be to develop new features in a separate branch each, without affecting the master branch during the development phase. When the new features have been tested by their developers and the ones affected, the group can decide merging the separate branch with the master branch.

It is advised to not develop more than one new feature per branch, and to also try and restrict the life-cycle of branches in reasonable time-periods, so that the progress of all branches is comparable.

One final aspect, relevant to the merging strategy, is to refrain from including compiled programs in the repository. This may lead into merging conflicts, whereas it stays out of the main philosophy of common repository, as compiled code cannot be indexed, compared and merged. For this purpose we suggest the use of a data hosting cloud server, which will be detailed later in this document (see Section 2.2.4).

2.2.3 Web-based graphical interface

A graphical interface is necessary to allow communication through an issue tracker, insertion of line comments and wiki pages. This includes visualization of the commit history, a repository browser, an issue tracker, a calendar, a Gantt diagram and a tool that permits code reviewing with merge requests. Its use is possible via any common web browser (see Figure 2).

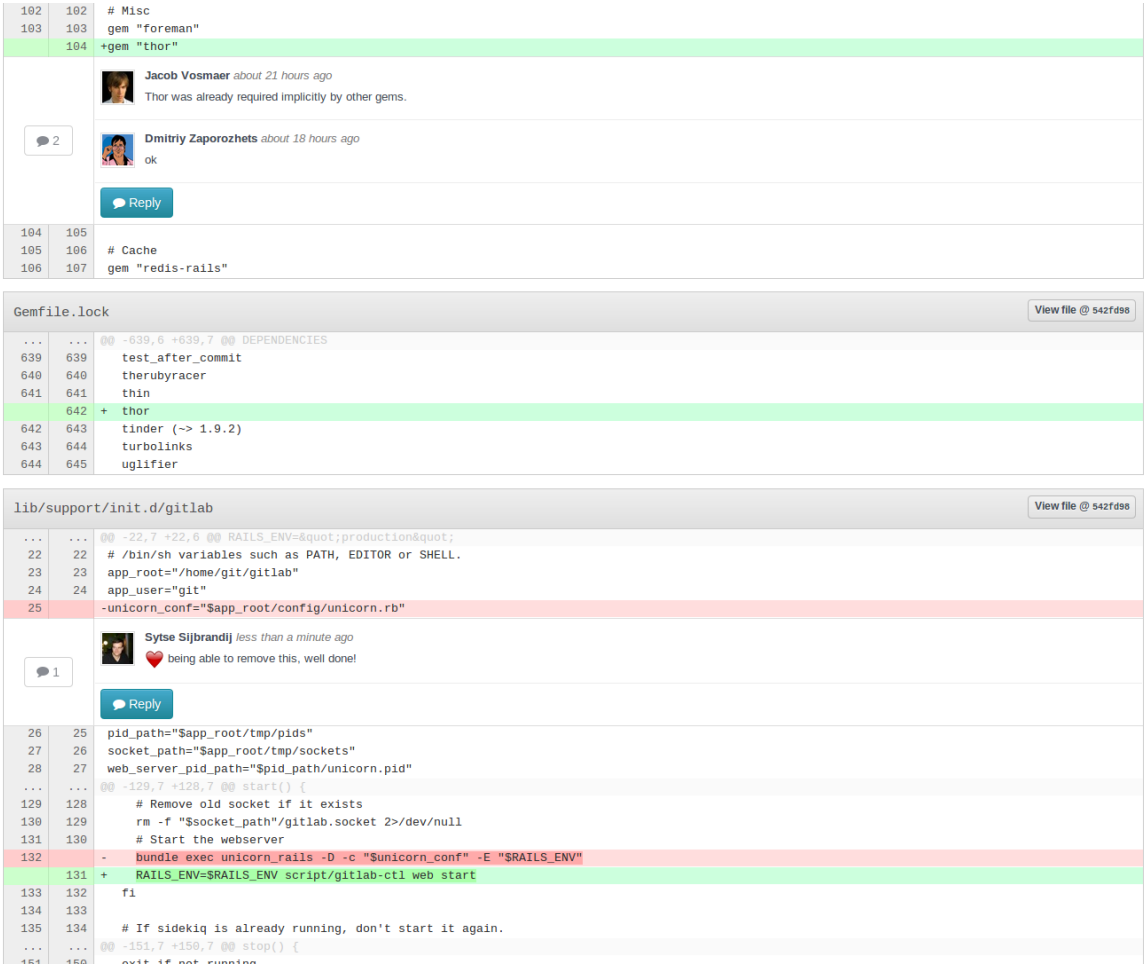


Figure 2: Repository graphical interface

The graphical interface that has been chosen is based on the open source software *Redmine*, written using the Ruby on Rails framework, cross-platform

and cross-database. Both repository and graphical interface are hosted by CUNI and accessible by all partners.[1]

2.2.4 Share data hosting repository

In addition to the version tracking software for the code, the workbench infrastructure includes a data repository. This is intended to host corpus data and material that is not expected to undergo severe development stages. It practically includes training datasets, in their different versions, that will be used in the project, especially those applied to develop each one of the different MT pilots.

The server is based on the open source cloud framework "OwnCloud". Its usage can be done through graphical interface or specific desktop client tools, similar to well-known cloud software (e.g. Dropbox). The server is hosted by the University of Lisbon, collocated with the project intranet.[2]

2.3 Pipelines

Pipelines are a key part of integrating the various components in a steady workflow. Through our communication between the partners there is a continuous flow of updates about the existing pipelines, the use cases, the desired components from all work packages, the programming languages used, the operating system compatibility, the employed interfaces and the needs for evaluation. We thereby describe the basic conclusions, allowing for modifications as the project development gets formed.

2.3.1 Review of available pipeline software

In order to acquire basic information about the state-of-the-art in pipeline software, we conducted a review on suggested tools and frameworks. The basic characteristics that the review was based upon, are:

- programming language
- user interface for design and execution
- parallelized execution
- execution on server-side
- documentation
- existing template support for particular tools
- learning curve

Additionally we collected information about the suitability for monolingual and bilingual pipelines, the extensibility and support by developers and community, the monitoring of progress and errors and the aggregation of results.

Basic capabilities of the reviewed software are outlined in Table 1.

[1] The code repos. interface can be accessed via: https://redmine.ms.mff.cuni.cz/projects/qtleap
[2] The data repository can be access via: http://qtleap.eu/repository

system	lang	design GUI	exec GUI	parallel	server exec	doc	exist. tools	effort
DuctTape	Jython	yes	yes	yes	yes	fair	SMT	some
EMAN	perl/ bash	no	no	yes	yes	good	SMT	few
Moses EMS	perl/ bash	no	yes	yes	yes	fair	SMT/ Moses	few
PCL	python	no	no	no	yes	little	SMT	big
SoapLab	Java	no	no	-	yes	good		some
Taverna	Java	yes	yes	yes	no	good		some
Treex	Perl	no	no	yes	yes	little	TectoMT	few
UIMA/ compare	Java/ XML	yes	-	-	yes	good	NLP, RBMT	some

Table 1: Review of pipelining software features

2.3.2 Bottom-up pipeline strategy

Through the kick-off meeting, project internal discussions and the technology survey, we acquired a broad overview on the software and existing tool chains that have already been used by the partners and are intended to be combined in the QTLeap pipelines.

As a result, one basic direction that was emphasized as a need by partners, is that the pipeline architecture has to be flexible and minimise the migration effort of the tools to be chained. For this purpose we proceed by adopting the pipeline systems that have already been widely used, given the possibility to easily integrate and extend previously implemented components that have been proven to be useful and succesful. These should serve on the bottom level for the pipelines of the baselines and the pilots' software, whereas they will altogether be pipelined on a higher level in a more generic pipeline graph, as required within the duration of the project. Through this modular approach, different pipeline architectures can be used simultaneously as per need, re-assuring that there is a clear documentation on how their configuration and interfaces can be matched.

Pipelines have to be designed and plotted through the code repository.

2.3.2.1 Moses EMS pipeline

The Moses Experiment Management System[3] has been one of the first pipeline systems that have been bound to building statistical machine translation system. It has been built in order to simplify the execution of the required training, tuning and evaluation scripts for phrase-based and hierarchical SMT systems based on MOSES. The definition of the pipeline is done by parameterising a

[3] http://www.statmt.org/moses/?n=FactoredTraining.EMS

configuration file, whereas execution can be done via a handy command that fires and monitors the underlying tasks.

The system offers additionally a graphical interface for listing and plotting the results of automatic evaluation metrics, whereas the execution can be monitored via constantly refreshed execution graph.

As this pipeline adapts better to the Moses software, this pipeline has been chosen by the majority of the partners in order to build and tune the baseline systems.

2.3.2.2 EMAN

EMAN has been considered as an alternative for the baselines, due to its modularity and its use by some partners, who required a complex chain of local linguistic tools. It offers a command-line kit for server execution, also based on configuration files and controlling the executed steps.

Users can add their own tools by implementing *seeds*, which allow full tracking of the execution variables, reproducing and cloning previous experiments etc. For instance, the implementation of the baseline of English to Czech takes advantages of the included "seeds" for processing morphological and truecased base forms on the Czech side.

2.3.2.3 Treex

The third considered solution for pipelines has also a long history connected with dependency-based translation. Treex is a highly modular software system making use of the ideas and technology created during the Prague Dependency Treebank project. It offers recipes for organizing modules of TectoMT on Machine Translation (English to Czech) as well as Depfix, a system for rule-based correction of English-Czech statistical outputs. It also includes modules for segmentation, tokenization, lemmatization and dependency parsing.

2.3.3 Implemented pipeline

An EMS pipeline for developing and testing one of our development baseline systems is plotted for illustration in Figure 6 (overview) and Figure 7 (detailed zoom) in the Appendix. It models the sequence of processing events, as they follow each other through the entire development. The first modules include the processing of the corpora, including the instantiation and tuning of included tools such as the *tokenizer,* the *truecaser* and the *compound splitter*. The core steps for building the phrase table are included in the "TRAINING" block, whereas the optimization of the parameters is handled by the tasks in the "TUNING" block. The last phase handles several scripts for testing the output on a particular development set and acquiring scores by automatic metrics.

2.4 Summary & Outlook

Development effort of QTLeap is organized through a multi-site interoperable workbench, which fulfills the basic project requirement for easy and effecting programming co-operation. The workbench is based on a GIT code version tracking system and a shared data cloud repository, both aided by graphical interfaces.

Building pipelines is done in a modular way, by allowing the integration of various pipeline structures of different levels, in a bottom-up approach, a path decided in order to minimize migration efforts.

The baseline pipeline that have been described in this Deliverable will be extended and chained to other pipelines, when more pilots and processing modules are available (e.g. deep processing) in the further steps of the project.

2.5 Update per M11

This section provides a brief update of the pipeline software used by the partners as per M11.

As has been pointed out above in Section 4.2, the plan was to: "[…] proceed by adopting the pipeline systems that have already been widely used [Moses EMS pipeline, EMAN and Treex], given the possibility to easily integrate and extend previously implemented components that have been proven to be useful and successful [...] [These systems] will altogether be pipelined on a higher level in a more generic pipeline".

This is in line with what was planned in the DoW where it says in the description of Task 3.1 "A selective subset of the baselines […] will first be integrated by example pipelines, followed by the ones involving deeper processing."

So far, after MT Pilots 0 (baselines) have been set up, all these baseline pilots have been implemented as pipelines, with EMS for Basque, Bulgarian, Dutch, German, Portuguese, Spanish, and EMAN for Czech. The pipelines are encapsulated on a higher level within the web service provided to the company to be embedded in the real usage scenario. The report on these pilots is due M12, under Deliverable D3.6, when the final baseline pipelines will be also documented in and uploaded into the project repository.

For Pilot 1, the entry-level to a deeper approach, the following languages will migrate to Treex: Basque, Czech, Dutch, Portuguese, and Spanish. Bulgarian and German will continue using EMS.

Modules within deep approaches in general are mostly language-specific. Through the use of pipelines, we ensure maximum internal operability and easy set-up of experiments and tests. The pipelines will be also encapsulated on a higher level within the web service to be embedded in the real usage scenario.

3 Workbench Version 2: Development, Evaluation, and Progress Tracking

During the Year 2 of the project a set of improvements were incorporated in the existing components of the workbench, following the experience that was gained from the development process of the baseline and the first pilot. As part of this new version a new subdomain "workbench.qtleap.eu" was created, in order to provide a unified interface for all the functions that are necessary to the developers. In particular, the workbench connects to:

- the "intranet", which serves as a general tool for communication and co-ordination of common activities
- the "data-cloud" (explained in Section 2.2.4), which serves for storing and exchanging common data among the partners
- the "Qtleap code" and "Treex code", which point to the git repositories where active development of the Treex code is taking place
- the "development translations", which points to an internal git repository which hosts translations produced by the current version of the actively developed systems
- the "evaluation panel", which introduces a new interface for comparing and evaluating systems against the QTLeap Corpus along the development effort.

For the latter parts, which are newly introduced at this point, we provide further explanations below

3.1.1 Code repositories

The Treex code has been migrated from a legacy SVN repository to a public Github repository[4]. Treex contains all the source code of TectoMT-based Pilots. This move has been done because git and GitHub allows us more flexible workflow than the original svn repository. Meanwhile, the code is publicly available as an open source project into GitHub's community and it has been cleaned from several big files that existed in the svn history of the legacy Treex repository. The GitHub issue tracker of this repository is meant to be used also for tracking problems with the Pilots (e.g., reporting problems in English analysis, language independent blocks etc.).

Additionally to the code repository, there is one repository[5] for training and scripts that are only relevant to the QTLeap project, without affecting the development of Treex and TectoMT. This contains makefiles and scenarios of the full training pipeline process, including particular experiments such as the "t-roundtrip".

[4] https://github.com/ufal/treex
[5] https://github.com/ufal/qtleap/

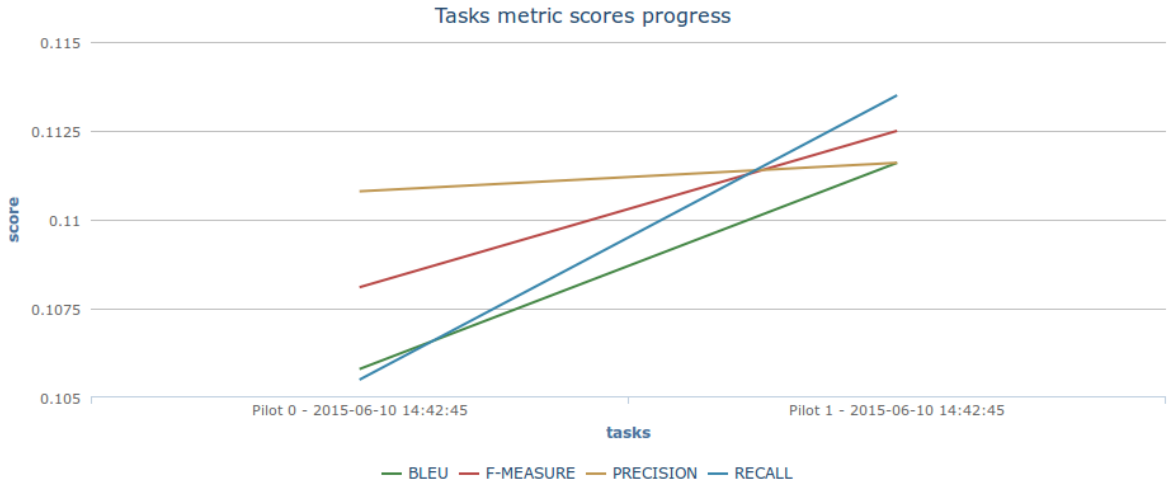
3.1.2 Development translations repository

The QTLeap Corpus is also stored in repository. It also includes the produced translations in plain text format, so that there can be easy tracking of changing between different versions. As developers proceed with creating a new version of their systems, they produce translations for the given corpus. These translations are placed in text files and committed as a new version of the corpus translations.

3.1.3 Evaluation panel

The development efforts are unified through the new evaluation panel.[6] The evaluation panel features automatic updating based on the development translations repository. In particular, every time a new version of a pilot translation is submitted by a developer, the backend of the evaluation panel detects that, and proceeds with computing automatic scores. The current version of the panel provides BLEU, precision, recall and F-measure. For every new system, the panel records the time this system was submitted, but also the commit id of the current state of the repository, so that developers can track the changes back to particular settings and revert them, if necessary.

Additionally, the panel provides the possibility to display a graphical representation of the change of the scores, every time a new version of the system is submitted (see Figure 3). The graph displays a progress line for all metrics, but users are able to turn individual metrics on and off, to allow for easier comparisons.



name	description	BLEU	PRECISION	RECALL	F-MEASURE	
Pilot 0 - 2015-06-10 14:42:45	Code commit 4dc30dbccd9d27008e4f5d4042559407570f5cbb - date 2015-06-09_00:54:23	0.1058	0.1108	0.1055	0.1081	delete
Pilot 1 - 2015-06-10 14:42:45	Code commit 4dc30dbccd9d27008e4f5d4042559407570f5cbb - date 2015-06-09_00:54:23	0.1116	0.1116	0.1135	0.1125	delete

Figure 3: Evaluation panel with graph

[6] This part includes code by Ondrej Klejch: https://github.com/choko/MT-ComparEval.

In order to allow for closer inspection of the particular improvements introduced by updates, the interface gives the possibility to compare single sentences among different systems. The user can select two versions of the system, and the panel can display several translated sentences from system one below the other along with original text, reference text and scores (see 4). Differences between the two versions can be highlighted on demand, for easier comparisons.

Additionally, the panel offers:

- graphs with bootstrap re-sampling (see Figure 5: Evaluation panel for bootstrap resampling)
- statistics of confirmed and unconfirmed n-grams (n-grams found in the translation confirmed or not against the reference)

Source	Try pressing the F11 key.
Reference	Tente carregar na tecla F11.
Pilot 0 - 2015-06-10 14:42:45	Tentar pressionar o F11 chave.
Pilot 1 - 2015-06-10 14:42:45	Tente carregar na chave f11.

	BLEU	BLEU-cis	F-MEASURE	F-MEASURE-cis	PRECISION	PRECISION-cis	RECALL	RECALL-cis
Pilot 0 - 2015-06-10 14:42:45	0	0	0	0	0	0	0	0
Pilot 1 - 2015-06-10 14:42:45	0.4949	0.5318	0.4949	0.5318	0.4949	0.5318	0.4949	0.5318
Diff	-0.4949	-0.5318	-0.4949	-0.5318	-0.4949	-0.5318	-0.4949	-0.5318

Figure 4: Evaluation panel for sentence-level comparison

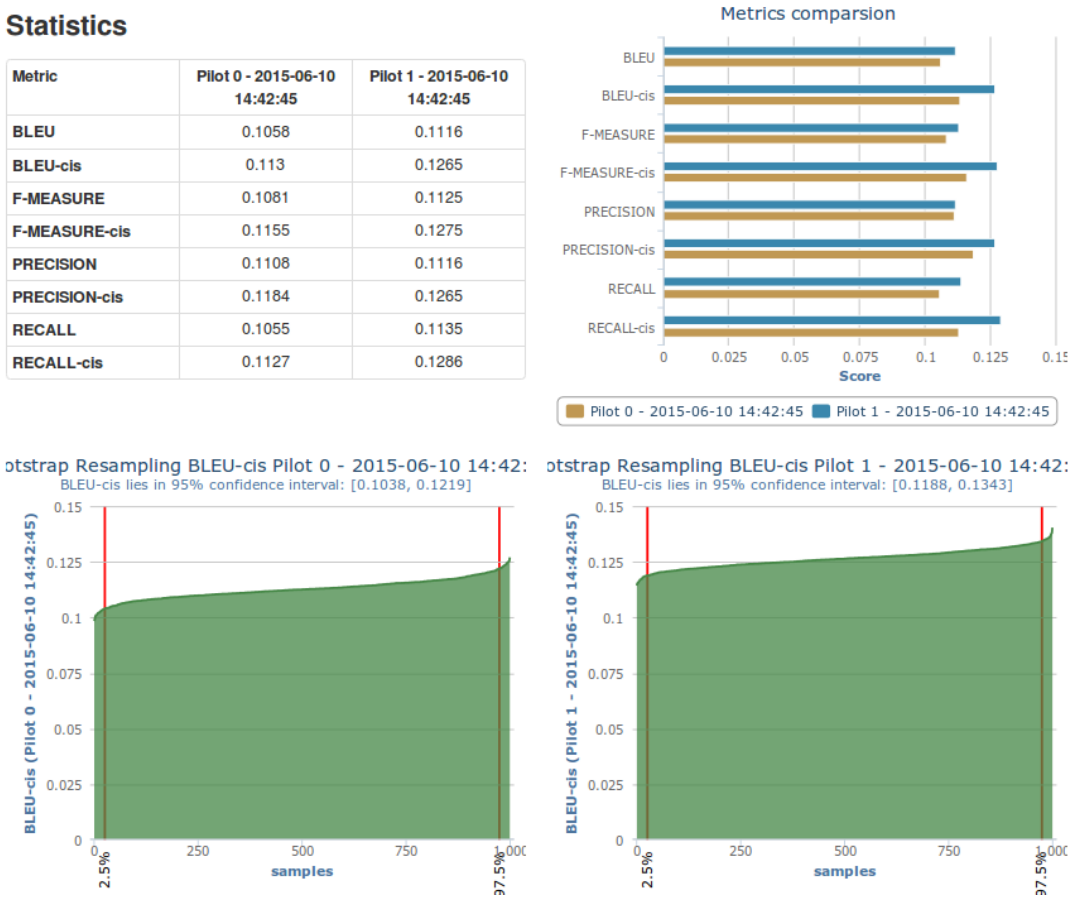


Figure 5: Evaluation panel for bootstrap resampling

4 Appendix

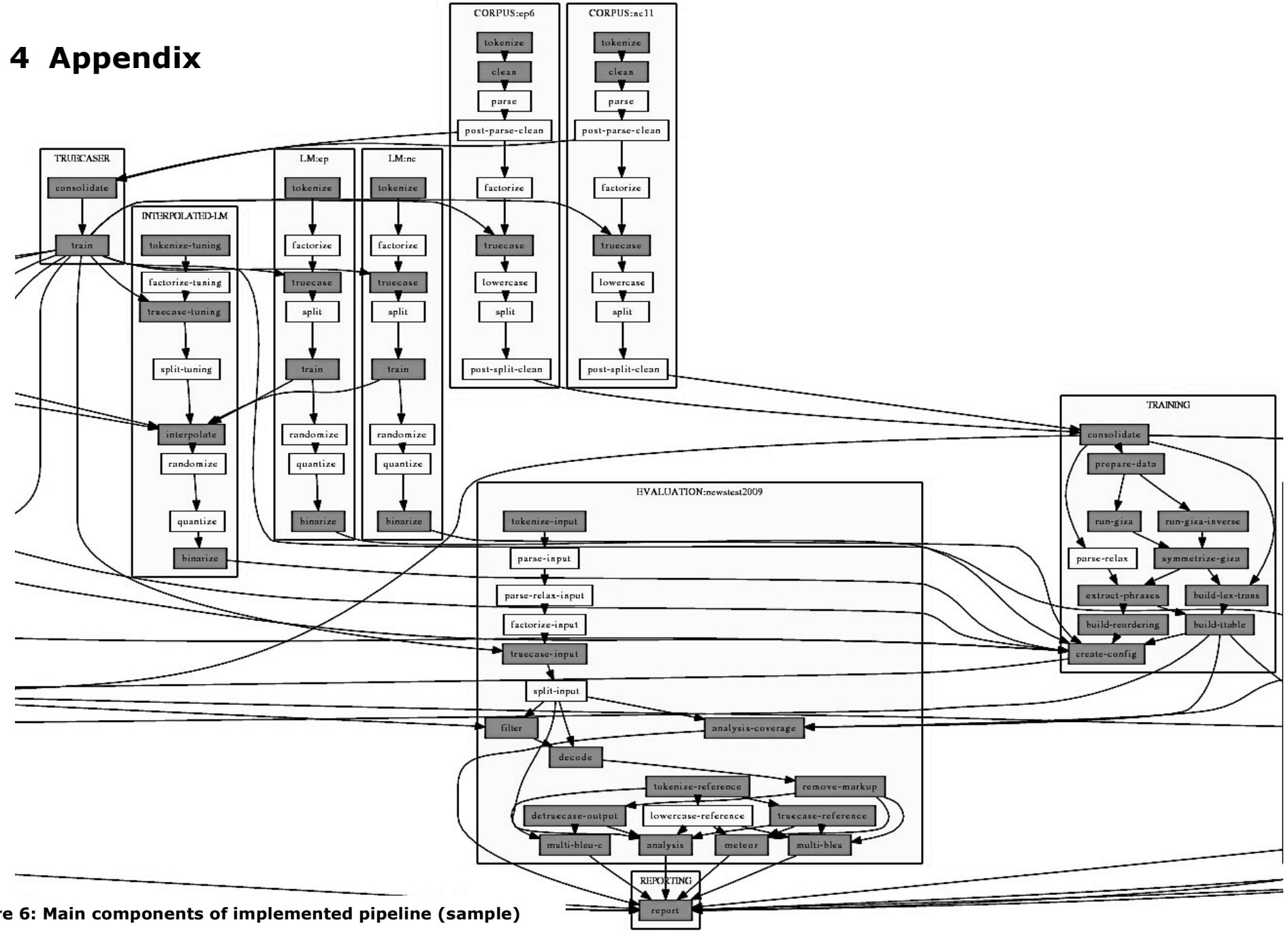


Figure 6: Main components of implemented pipeline (sample)

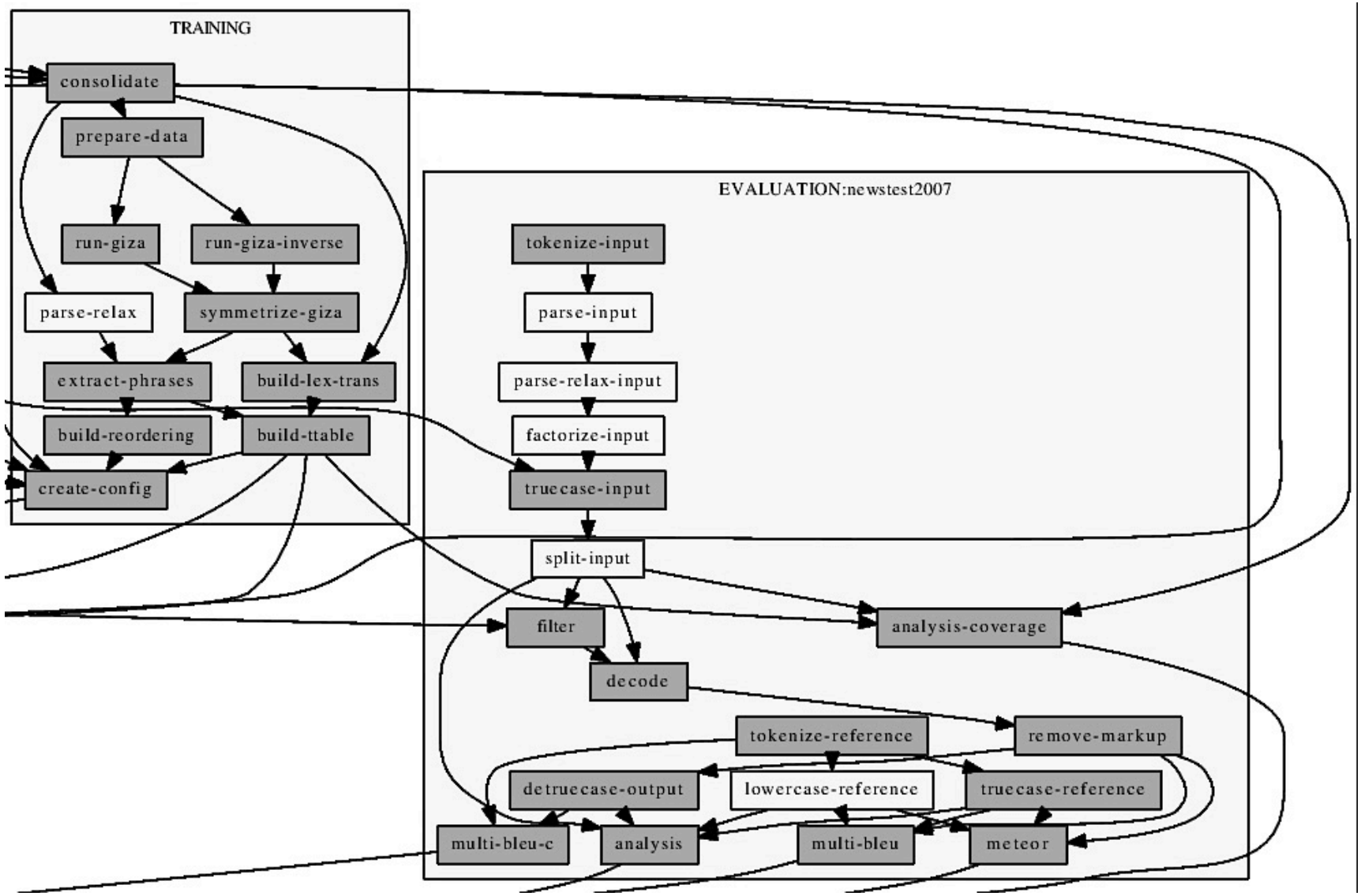


Figure 7: Sample pipeline, focusing on the components of training and evaluation