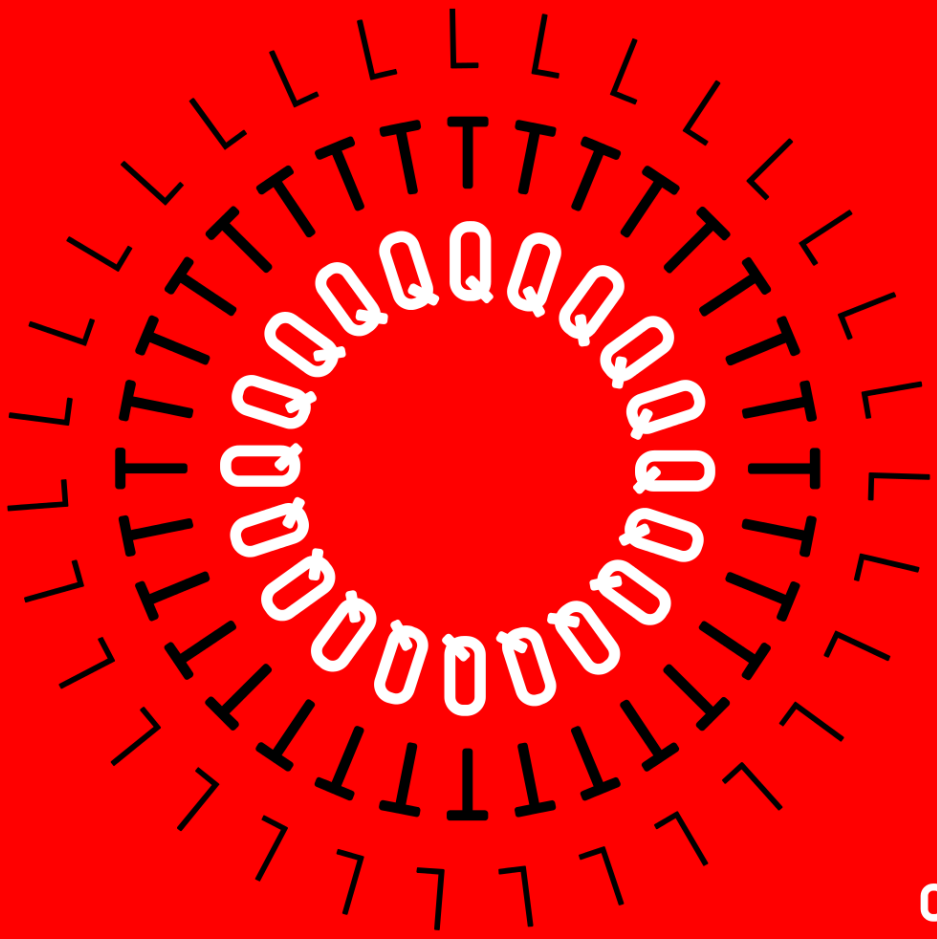


qtleap

quality translation by deep language engineering approaches

Report on the second MT pilot and its evaluation

DELIVERABLE D2.8

VERSION 1.1 | 2015-11-16

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:

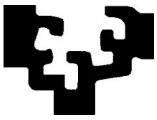
Faculty of Sciences, University of Lisbon

German Research Centre for Artificial Intelligence

Charles University in Prague

Bulgarian Academy of Sciences

Humboldt University of Berlin



University of Groningen

Higher Functions, Lda

Revision history

Version	Date	Authors	Organisation	Description
0.1	Oct 8, 2015	Martin Popel, Ondřej Dušek, Michal Novák	CUNI	First draft
0.2	Oct 21, 2015	Arle Lommel	DFKI	Results of manual evaluation
0.3	Oct 22, 2015	Aljoscha Burchardt	DFKI	Description of German Pilot 2
0.4	Oct 23, 2015	Ondřej Dušek	CUNI	General description of TectoMT, new specifics for EN, CS, and (partial) NL
0.5	Oct 23, 2015	Gertjan van Noord	UG	Description of Dutch Pilot 2
0.6	Oct 25, 2015	Rudolf Rosa	CUNI	Intros, conclusion, TM interpolation
0.7	Oct 27, 2015	Luís Gomes, António Branco	FCUL	EasyAdapt
0.8	Oct 27, 2015	Gorka Labaka	UPV/EHU	new specifics for ES and EU
0.9	Oct 30, 2015	Velislava Todorova, Aleksander Popov	IICT-BAS	Description of Bulgarian Pilot 2
1.0	Oct 30, 2015	Kiril Simov, Petya Osenova	IICT-BAS	Internal Review
1.1	Oct 30, 2015	Luís Gomes	FCUL	new specifics for PT

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 second MT pilot and its evaluation

DOCUMENT QTLEAP-2015-D2.8
EC FP7 PROJECT #610516

DELIVERABLE D2.8

completition
FINAL
status
SUBMITTED
dissemination level
PUBLIC

responsible
Jan Hajič (WP2 Coordinator)
reviewers
Kiril Simov, Petya Osenova
contributing partners
CUNI, FCUL, DFKI, IICT-BAS, UPV/EHU, UG, HF

authors
Martin Popel, Ondřej Dušek, António Branco, Luís Gomes, João Rodrigues, João Silva, Eleftherios Avramidis, Aljoscha Burchardt, Arle Lommel, Nora Aranberri, Gorka Labaka, Gertjan van Noord, Rosa Del Gaudio, Michal Novák, Rudolf Rosa, Jaroslava Hlaváčová, Jan Hajič, Velislava Todorova, Aleksander Popov

Contents

List of Abbreviations

BDT	Basque Dependency Treebank
FGD	Functional Generative Description
HMM	Hidden Markov Model
HMTM	Hidden Markov Tree Model
LM	language model
MT	machine translation
NED	named entity disambiguation
NERC	named entity recognition and classification
NLP	natural language processing
PB-SMT	phrase-based statistical machine translation
PDT	Prague Dependency Treebank
SMT	statistical machine translation
SRL	semantic role labeling
STSG	synchronous tree substitution grammar
UD	Universal Dependencies
WER	word error rate
WSD	word sense disambiguation
TM	translation model
WMT	workshop on statistical machine translation

1 Overview

Deliverable D2.8 describes and evaluates Pilot 2 (the second MT pilot system, enhanced with lexical semantics), which was published in QTLeap deliverable D2.7. We focus on the changes and improvements done since finalizing Pilot 1 (entry-level deep MT system, delivered in D2.3 and described in D2.4).

In Pilot2, we resorted to lexical semantics processing (as described in detail in deliverable D5.7), including the use of concept resolution, via word sense disambiguation to WordNet, and resolution of domain-specific entities, via gazetteers mined from domain-related resources. The techniques used involve linked open data like WordNet and Wikipedia/DBpedia. Pilot2 systems thus incorporate techniques experimented in QTLeap Work Package 5 (WP5).

The overall goal of the project is to produce high-quality translation between English (EN) and another language (X in the following text) by using deep linguistic information. All language pairs follow the same processing pipeline of analysis, transfer and synthesis (generation)[1] and adopt the same hybrid MT approach of using both statistical as well as rule-based components in a tightly integrated way for the best possible results.

Section 2 of this deliverable describes general development and enhancements of the translation systems (Sections 2.1 and 2.2) and important improvements introduced for each of the individual translation directions. The pipeline of language processing tools for each language is described in Sections 2.3–2.10.

As most of the translation systems were newly created within the QTLeap project, many of their components were still very basic in Pilot 1, and a lot of effort has been invested in making the systems more mature in Pilot 2. The language resources and tools enhanced to support these MT systems are referred to in the sections below and their description is provided in Deliverable D2.5. A special focus of Pilot 2 and a major source of translation quality improvements was the incorporation of various semantic linking and resolving techniques; they are described in detail in D5.7, which also reports the BLEU score deltas of each technique.

Section 3 is devoted to empirical evaluation of the Pilot 2 systems in terms of translation quality. The evaluation was performed generally in the same way as Pilot 1 evaluation in D2.4. Moreover, it compares Pilot 2 systems both to their Pilot 1 versions, as well as to the baseline Pilot 0 systems.

2 Pilot 2 Systems

2.1 Bringing Entry-level Systems to Pilot 2 Version

Our base approach to the deep translation is based on the TectoMT system. The general TectoMT pipeline, described in detail for Pilot 1 systems in D2.4 and briefly summed up here in Section 2.2, is language independent, and consists of analysis, deep transfer, and synthesis steps.

A basic version of the TectoMT system for en→cs translation had already been developed by CUNI before the start of the QTLeap project. For Pilot 1, we improved the en→cs system and implemented cs→en, eu↔en, nl↔en, pt↔en, es↔en almost from scratch, thus obtaining 10 entry-level deep MT systems based on TectoMT (English to

[1] The terms *synthesis* and *generation*, as used in this deliverable (and related literature), are synonyms.

and from Czech, Basque, Dutch, Portuguese, and Spanish). For German and Bulgarian, pre-existing non-TectoMT systems were used and improved.

For Pilot 2, we further developed all of these systems, fixing numerous problems of the entry-level systems, not only adapting them to the QTLeap setting but also enhancing the systems in general. Furthermore, we focused on enriching the systems in various ways by lexical semantics, for example by

- **HideIT**, a module for handling of "fixed" entities such as URLs or commands, which should not be translated (see D5.7, Section 2.3.1).
- **Gazetteers**, a module for translation of special types of text (e.g. menu items, button names, software messages) using specialized lexicons (see D5.7, Section 2.3.2).[2]
- **Replacement by concept ids**, the replacement of the words in the translation models by interlingual conceptual representations (see D5.7, Section 2.1).[3]
- **TM interpolation** for the purpose of domain adaptation (see Section 2.2.5 of this deliverable).

Further sections deal with the specifics of individual language pairs, summarizing differences in system training and operation for each language pair and highlighting updates and improvements compared to Pilot 1.

We continue to benefit greatly from the design of TectoMT, which is highly modular, and consists of a language-universal core and language-specific additions. This allowed each of the partners to experiment with enhancing TectoMT in various ways for their language only without influencing the other languages, yet at the same time any improvements that proved useful were then easy to apply to the other languages as well. Thus, each partner had an access to all currently implemented features, and it was easy for them to experiment with switching these on and off, or to further adapt them for their language as necessary, to finally obtain the best-performing setup for their language. Thanks to that, each team was able to work with a great autonomy and only loose need for coordination with others, and yet we were then able to quickly and easily integrate the results of any team into the common system to benefit all. This has proven to be a very efficient approach.

For German and Bulgarian, we decided to keep developing the non-TectoMT deep MT systems for Pilot 2. The improvements that bring these systems from Pilot 1 to Pilot 2 are described in Section 2.9 for the Bulgarian Deep factored MT system, and in Section 2.10 for the German Qualitative system combination.

2.2 General structure of TectoMT-based Systems

TectoMT is a structural machine translation system with deep transfer, first introduced by **?**. It uses two layers of structural description, *a-layer* (shallow, see Section 2.2.1) and *t-layer* (deep, see Section 2.2.2).

The system consists of 3 phases:

1. The analysis phase has two steps and proceeds from a-layer to t-layer (see Section 2.2.3).

[2] As usual in named entity recognition and classification (NERC) research, *gazetteer* means a list of named entities of a given type, not only geographic names.

[3] This was experimented with in Pilot 2 only for the translation pairs involving Portuguese.

2. The transfer phase of the system is based on Maximum Entropy context-sensitive translation models [?] and Hidden Tree Markov Models [?]. It is factorized into three subtasks: t-lemma, formeme, and grammatemes translation (see Section 2.2.4). The transfer phase features techniques newly introduced in Pilot 2 – model interpolation and the "EasyAdapt" technique, described in Sections 2.2.5 and 2.2.6, respectively.

3. The generation phase is a rule-based pipeline that gradually changes the deep target language representation into a shallow one, which is then linearized to text (cf. Section 2.2.7).

All of the aforementioned sections deal with the actual operation of a trained system. The training of the system is briefly described in Section 2.2.8.

2.2.1 Surface syntax analysis

The *a-layer* (analytical layer) is a surface syntax layer which includes all tokens of the sentence, organized as nodes into a labeled dependency tree (*a-tree*).

Each a-layer node is annotated, among others, with the following types of information:

- *word form* – the inflected word form as it appears in the original sentence (including capitalization).
- *lemma* – the base form of the word form, for instance infinitive for verbs, nominative singular for nouns.
- *part-of-speech tag* and *morphological information* – Interset [?] is used to facilitate language-independent rules in TectoMT.
- *afun* – surface dependency label. The labels largely correspond to commonly known syntactic functions such as subject, predicate, object, and attribute (`Sb`, `Pred`, `Obj`, `Atr`).

2.2.2 Deep syntax analysis and transfer layer

The *t-layer* (tectogrammatical layer) is a deep syntactic/semantic layer describing the linguistic meaning of the sentence according to the FGD (Functional Generative Description) theory [?]. Its dependency tree (t-tree) includes only content words (nouns, full verbs, adjectives, adverbs) as nodes (*t-nodes*).

Auxiliary words are not present on the t-layer as separate nodes, but they usually influence the individual attributes of t-nodes. On the other hand, there are nodes on the t-layer that do not correspond to any surface words, e.g., nodes representing pro-dropped subject personal pronouns.

In addition, coreference is marked in the t-layer using special coreference links (non-tree edges).

Each regular *t-node* has the following attributes:

- *t-lemma* – "deep lemma" (mostly identical to surface lemma).
- *functor* – a semantic role label. There are over 60 different semantic role labels based on the FGD theoretical framework, such as `ACT` (actor/experiencer), `PAT` (patient/deep object), `TWHEN` (time adverbial), `RSTR` (modifying attribute) etc.

- *grammatemes* – a set of deep linguistic features integrating features relevant to the meaning of the given sentence (e.g., person, number, tense, modality).

- *formeme* – morpho-syntactic form information [**?**], composed of coarse-grained part-of-speech based on syntactic behavior, prepositions or subordinate conjunctions, and coarse-grained syntactic form (e.g., `v:to+inf` for infinitive verbs or `n:into+X` for a prepositional phrase).

2.2.3 Analysis

As already mentioned, the analysis in TectoMT is two-step: the first step uses standard dependency parsers trained on treebanks to reach the a-layer, while the second step is composed mostly of rule-based modules that convert the a-layer tree into a t-layer tree.

The a-layer parsing is preceded by preprocessing steps which include sentence segmentation, tokenization, lemmatization, and morphological tagging. The a-layer parsing itself can then be performed by various dependency parsers [**?**].

The a-tree is then gradually transformed into a t-tree by modules that build the t-tree by removing auxiliary words (and preserving links to them on the a-layer), changing surface lemmas to t-lemmas, and assigning formemes, functors, and grammatemes to each node. Final stages of the t-layer analysis pipeline involve reconstructing deep subjects (for pro-drop languages, imperatives, and passive) and coreference resolution.

2.2.4 Transfer: Translation Factorization

The transfer on the t-layer is separated into three relatively independent simpler subtasks: the translation of t-lemmas, then the conversion of formemes and grammatemes [**??**]. This approach makes a strong assumption that topology changes to t-trees are rarely needed as t-trees representing the same content in different languages should be very similar. This allows us to model each of these three subtasks by a symmetric source-target one-to-one mapping.

The t-lemma and formeme transfer is treated jointly in the following main steps:

1. Producing an n-best list of translation variants using t-lemma translation model(s)
2. Producing an n-best list of translation variants using formeme translation model(s)
3. Joint re-ranking of the n-best lists using Hidden Markov Tree Models (HMTM)

For each t-lemma/formeme in a source t-tree, the translation model (TM) assigns a score to all possible translations observed in the training data. This score is a probability estimate of the translation variant given the source t-lemma/formeme and other context, and it is calculated as a linear combination of several components:

- Discriminative TMs – prediction is based on features extracted from the source tree, using a maximum entropy (MaxEnt) model [**?**].

- Dictionary TMs – this is only a dictionary of possible translations with relative frequencies (no contextual features are taken into account, called *static* in the source codes).

- Other – backoff components that focus on out-of-vocabulary t-lemmas using hand-crafted rules and various small "derivative" dictionaries.

TM for	TM type Dictionary	Discriminative
Formemes	1.0	0.5
T-lemmas	0.5	1.0

Table 1: Weights of TMs in interpolation; the same set was used both for out-of-domain TMs and in-domain TMs in all translation directions.

2.2.5 Transfer: Model Interpolation

One of the major improvements of the transfer phase in Pilot 2 for all language pairs is the introduction of a general mechanism for translation models interpolation. Part of the mechanism was implemented already in the original TectoMT, but now in Pilot 2, we have included easy-to-use parametrization and training procedures.

Therefore, in Pilot 2 systems, we can use the TM interpolation for two purposes:

1. Same as in Pilot 1, we interpolate Dictionary and Discriminative TMs because of their complementary reliability, precision and coverage (see D2.4 for details).

2. Pilot 2 introduces the interpolation of a general-domain TM and an in-domain TM [**?**]. So the TM interpolation is also used for the purpose of domain adaptation.

In total, TectoMT Pilot 2 systems use four standard TMs trained from parallel data by default – a Dictionary formeme TM, a Discriminative formeme TM, a Dictionary t-lemma TM, and a Discriminative t-lemma TM (plus optionally other special TMs). For Pilot 2, we trained this set of four models on each of the available parallel corpora – both on the general domain one (usually trained on news corpora or Europarl) and an IT domain one (Batch 1 of QTLeap corpus).

In the interpolation of the TMs, each TM is assigned an interpolation weight, as listed in Table 1. We use the same weights for in-domain TMs and out-of-domain TMs.

The standard approach, as applied in phrase-based SMT systems, would be to use tuning on an in-domain development set to find a well-performing set of weights, by employing an optimizer such as MERT or PRO. However, we do not apply interpolation weights tuning for Pilot 2 as the in-domain training dataset, Batch 1, is very small (1000 sentences), and we thus did not want to further divide it into smaller training and development parts. Still, we believe to be able to perform weight tuning for Pilot 3 by tuning on another Batch, which may lead to additional performance gains.

The interpolated TMs give out n-best lists of most probable translations for the t-lemma and formeme of each node. These are subsequently jointly re-ranked by Hidden Markov Tree Models (HMTMs), [**??**]. HMTMs are similar to standard (chain) Hidden Markov Models but operate on trees. Transition probability is modeled by a tree language model, while emission probability is the probability of the particular source-language t-lemma/formeme being a translation of the hidden target-language t-lemma/formeme.

The translation of grammatemes is much simpler than the translation of t-lemmas and formemes since abstract linguistic categories such as tense and number are usually paralleled in the translation. Therefore, a set of relatively simple rules (with a list of exceptions) is sufficient for this task. Such rules are inherently language-specific.

2.2.6 Transfer: EasyAdapt

The 'frustratingly easy domain adaptation' (EasyAdapt) technique [**?**] is a simple feature augmentation technique that can be used in combination with many learning algorithms. EasyAdapt has been applied successfully for various NLP tasks, including Named Entity Recognition, Part-of-Speech Tagging, and Shallow Parsing. Even if EasyAdapt is not directly applicable to the models typically used in Statistical Machine Translation, a similar approach has been shown to improve results for translation as well [**?**].

Although EasyAdapt has been developed in the context of domain adaptation, it is best described as a very simple, yet effective, multi-domain learning technique [**?**]. In EasyAdapt, each input feature is augmented with domain specific versions of it. If we have data from K domains, the augmented feature space will consist of $K + 1$ copies of the original feature space. Each training/testing instance is associated with a particular domain, and therefore two versions of each feature are present for a given instance: the original, general, version and the domain specific version.

The classifier may learn that a specific feature is always important, regardless of the domain (and thus it will rely more on the general version of the feature), or it may learn that a specific feature is relevant only for particular domain(s) and thus rely more on the relevant domain specific features. As a result, we obtain a single model which encodes both generic properties of the task as well as domain specific preferences.

We implemented EasyAdapt in our Discriminative TMs by adding, for each original feature f, a feature f_d if the training/testing instance is from domain d. In our experiments there are only two domains, the IT domain, which we regard as in-domain for the translation system, and the general domain.

While the interpolated models described above (Section 2.2.5) consistently outperform the baseline models, the behavior of EasyAdapted models was less consistent, showing a slight improvement in one translation direction and a slight performance decrease in the other [**?**]. Therefore, we chose TM interpolation instead of EasyAdapt, as the technique for combining in-domain and general domain corpora for Pilot 2.

2.2.7 Synthesis

The synthesis is composed of a series of small, mostly rule-based modules that perform gradual changes on the trees, converting them to a-layer trees that contain inflected word forms and can be linearized to plain text. Generators in this scenario are designed to be domain-independent and known to reach high performance [**???**].

The tasks carried out by the modules in the pipeline are language-specific, but generally include solving the following problems:

- *Word ordering* – word order imposed by the target language is enforced.
- *Agreement* – morphological attributes are deduced based on agreement with properties of the context (as in subject-predicate agreement or noun-attribute agreement).
- *Inserting grammatical words* – a-layer nodes are created for prepositions, subordinate conjunctions, auxiliary verbs, particles, articles, punctuation, and other grammatical words which do not have separate nodes in t-layer trees.
- *Inflection and phonetics* – inflected word forms are produced based on known morphological and phonetic information from the context.

- *Capitalization* – the first word in a sentence is capitalized.

2.2.8 System Training

There have been no major changes in the TectoMT training process since Pilot 1; the only important change is the addition of in-domain translation models, which use the same training pipeline on different data (see Section 2.2.4). We include here only a brief overview of the training process. A more detailed description can be found in D2.4.

While most analysis components (taggers, parsers, see Section 2.2.3) are trained in a standard fashion on annotated corpora and treebanks, training the translation models requires a more complex procedure using automatic annotation to obtain large deep parallel treebanks. This is due to two reasons – first, it follows the real-life scenario where error-prone automatic features are extracted from the data, and second, the TMs require very large parallel treebanks, which are expensive to obtain by manual work.

We obtain parallel deep treebanks by using the analysis pipelines for both respective languages (see Section 2.2.3), starting from sentence-aligned bitexts and going through tokenization, morphology, dependency parsing to a-layer and t-layer conversion. The analysis pipeline is run independently on each of the two languages, taking no advantage of the joint bilingual processing for the time being.

Word-alignment on t-layer nodes is obtained in three steps – first, automatic word alignment using the GIZA++ tool [**?**], second, projection to the corresponding nodes in the t-trees, and third, additional heuristic rules used to align t-nodes that have no counterparts on the surface.

Note that once a parallel treebank for a given language pair has been constructed, it can be used for training translation models in both translation directions.

2.2.9 System Testing

For Pilot 2, we have created a new testing framework, which offers integration with the QTLeap Evaluation Workbench.

In Pilot 1, the TectoMT scenarios (sequences of TectoMT blocks) were stored in plain-text `*.scen` files. In Pilot 2, TectoMT scenarios can be stored also in Perl modules, which has the following benefits:

- These scenarios can be easily parametrized. So for example, we have one scenario for English analysis (from plain text up to tectogrammatical layer),[4] which has parameters `tagger`, `ner` and `functors`, which specify which morphological tagger, named entity recognizer and functor (semantic roles) detector should be used. Parameter `domain` specifies the domain of the analyzed text (so far the only possible values are: general and IT), if it is known. Parameter `gazetteer` specifies which gazetteers (see Section 2.3.2 in QTLeap Deliverable D5.7) should be used. The last two mentioned parameters are applicable also for the transfer and synthesis scenarios. Thanks to the parameters, one scenario can be shared between several applications and it is obvious which blocks are needed vs. optional and which blocks are specific for a given domain/application.

- These scenarios are versioned in the main Treex repository[5] together with the source

[4] https://github.com/ufal/treex/blob/master/lib/Treex/Scen/Analysis/EN.pm
[5] https://github.com/ufal/treex

codes of the Treex blocks. So the scenario can be kept synchronized with the source code of the blocks it includes.

- Running Treex applications from command line is now easier. With scenarios in plain-text files one had to specify a relative or absolute path to the scenario file, so usually each Makefile was accompanied with all the needed scenarios (which were duplicated over several applications). Now, one can run the treex command (e.g. `treex -Len Read::Sentences from=my.txt Scen::Analysis::EN Write::Treex to=my.treex.gz`) from any directory because the scenario (`Scen::Analysis::EN`) will be found as other Perl modules (based on `PERL5LIB` variable).

The new testing framework[6] has a directory for each translation direction (e.g. `en-cs`) and a subdirectory for each test set (`batch1a`, `batch2a`, `batch3a`, `news`). Replicating the Pilot 2 results (after installing Treex as described in D2.7) is easy:

```
git clone https://github.com/ufal/qtleap
cd qtleap/translate/en-cs/batch3a/
make translate eval D="optional description describing this experiment"
# Each experiment has a number, eg. 42 and is stored in runs/042_<date>.
make help # see a list of commands
# Now, copy the experiment #42 to the qtleap-corpus repository
make archive-042
cd ../../../qtleap-corpus/
git status
git commit -a
git push
# After few minutes, the results will be automatically evaluated
# and stored in the QTLeap Evaluation Workbench.
```

2.3 English Components

This section details English-specific features of the TectoMT pipeline, used for all language pairs. The pipeline is an improved version of the Pilot 1 pipeline described in D2.4; therefore, we only give here a very brief overall explanation and focus mainly on the changes.

2.3.1 Analysis

The English analysis follows the annotation pipeline used for the CzEng 1.0 parallel corpus [**?**], using a (rule-based) tokenizer, a statistical part-of-speech tagger [**?**] and dependency parser to a-trees [**?**], with rule-based post-processing.

The t-layer conversion starts from the a-tree and follows the process outlined in Section 2.2.3 very closely (see also D2.4 for details); there have been no significant changes in English analysis since Pilot 1.

[6] https://github.com/ufal/qtleap/tree/master/translate

2.3.2 Synthesis

The English synthesis pipeline also adheres to the general setup presented in Section 2.2.7 (see also D2.4 for details). The following improvements have been implemented since Pilot 1:

- The original English **article insertion** module using heuristic rules based on **?** has been replaced by a much simpler module that assigns articles based on value of the definiteness grammateme. Since most of the QTLeap languages have an explicit notion of definiteness and use articles, it is much more straightforward to use the same definiteness value in English – e.g., in most cases, a definite article in the source language will result in a definite article in English. This change brought a more than 3 BLEU points' gain for Dutch-English and Spanish-English Pilot 2 systems on the Batch1 data.

  The articles are still not copied verbatim from the source language as their usage may not correspond 1:1 to the notion of definiteness. Czech is the only QTLeap language where no articles are used and definiteness is only implicit; it must therefore be handled in transfer (see Section 2.5).

- An in-domain **Hidden Markov Tree Model** (HMTM, see Section 2.2.4) has been trained using a cleaned data dump from the SuperUser website.[7] The corpus is relatively small but matches the target domain perfectly.

  We also performed preliminary experiments on Czech-English translation in the Batch1 data set using a larger in-domain model using data from the Ubuntu Dialogue corpus [**?**] as well as large general-domain models trained on the CzEng corpus [**?**] and WMT'15 News Crawl and WMT'15 News Discussions,[8] but none of the larger models yielded an improvement – on the contrary, using them resulted in a worse BLEU score than using no HMTM. A simple concatenation of SuperUser data with any of the other corpora also did not improve the results.

  The HMTM is only used in the final Dutch-English Pilot 2 setup, where we tuned the parameters on the Batch1 data set, which resulted in a 0.6 BLEU point gain on the Batch2 data, confirmed by manual examination of the results. With HMTM, the Czech-English setup did gain about 0.1 BLEU point on the Batch2 data, but our manual investigation of the translation did not confirm a quality improvement. Therefore, we decided not to use it for the final Pilot 2 setup. A more extensive tuning will probably be required. Adding HMTM to other language pairs is left for Pilot 3.

The Pilot 2 version of English synthesis further contains several small bug fixes to word ordering and negation handling.

[7] http://superuser.com/, dumps available here: https://archive.org/details/stackexchange.

[8] See English News Crawl (articles from 2014) and News Discussions Version 1 under http://www.statmt.org/wmt15/translation-task.html.

2.4 Basque: TectoMT

2.4.1 Analysis

Same as in Pilot 1, the analysis of Basque input uses the ixa-pipes modules. Specifically, for the PoS tagger and lemmatizer,[9] and the dependency parser,[10] details on the system are given in D2.4.

Next, in the rule-based pipeline, the dependency trees are converted to a-layer-compatible trees and finally to t-trees. The t-tree conversion pipeline parallels other languages (see D2.4 and Section 2.2.3). Since Pilot 1, the pipeline has undergone changes on the definition of the Basque formemes and fixes of some problems on the assignment of the verb grammatemes.

2.4.2 Transfer

Both English-to-Basque and Basque-to-English transfer use discriminative and dictionary translation models as described in Section 2.2.4, with the newly added in-domain models. Those translation models have to be retrained due to the changes done on Basque analysis (mainly on the definition of the used formemes). Additionally, in Pilot 2 the following rule-based transfer modules have been added:

- Rules for English-to-Basque translation of relative clauses, which enforce the proper translation of the relative pronouns.
- Rules to address differences in definiteness between Basque and English.
- Proper generation of the “There is" structure in English, from the Basque counterpart structure.

2.4.3 Synthesis

Same as in Pilot 1, the Basque synthesis pipeline adheres to the general synthesis scenario (see Section 2.2.7), including morphology initialization and agreements (subject-predicate and attribute-noun), insertion of prepositions and conjunctions based on formemes, and insertion of punctuation. Pilot 2 features an improved handling of word ordering and the generation of Basque articles and auxiliary verbs.

As it is described in D2.4, the final step of the Basque synthesis, the generation of word forms, is performed by Flect [**?**]. Flect, the morphological generation tool developed by CUNI, uses a morphologically annotated corpus to automatically learn how to inflect word forms based on lemmas and morphological features.

2.5 Czech: TectoMT

2.5.1 Analysis

The Czech analysis has undergone just minor changes since D2.4; therefore, we only give here a very brief description of the whole pipeline. Please refer to D2.4 for more details.

The analysis pipeline is based on the annotation pipeline of the CzEng 1.0 corpus [**?**], starting with a rule-based tokenizer and a statistical part-of-speech tagger [**?**] and

[9] http://ixa2.si.ehu.es/ixa-pipes/eu/ixa-pipe-pos-eu.tar.gz
[10] http://ixa2.si.ehu.es/ixa-pipes/eu/ixa-pipe-dep-eu.tar.gz

dependency parser [**??**]. These steps result in a-layer trees, which are then converted to t-layer using a rule-based process, which follows very closely the description in Section 2.2.3.

All domain-adaptation improvements of Pilot 1 are still in place in Pilot 2 (please see D2.4 for details), we have only added one rule-based improvement – since NameTag (**?**), the Named Entity Recognizer used in Pilot 1, has a relatively low recall on the IT domain, we also recognize as named entities all mid-sentence uppercase words that are missing from the part-of-speech tagger dictionary. This yields a minor improvement of the article assignment further down the Czech-English translation pipeline.

2.5.2 Transfer

Similarly to Pilot 1, the English-to-Czech transfer uses a combination of translation models and tree model reranking. In Pilot 2, we newly include in-domain translation models into the model interpolation, as described in Section 2.2.4.

All additional rule-based changes to the t-tree topology (insertion and deletion of t-layer nodes) and grammateme changes, such as the addition of grammatical gender, as well as specialized translation models for the pronoun *it* and reflexive pronouns [**??**] have not been changed since Pilot 1. The rule-based transfer blocks have been further improved based on errors found in Pilot 1 outputs; this includes mainly non-isomorphic translation (one Czech t-tree node translates to two English nodes and vice versa) as well as translating domain-specific concepts that are not handled well by the translation models.

The Czech-to-English transfer also follows the process described in Section 2.2.4, but it doesn't use a HMTM reranking of the translation options (cf. also Section 2.3 for details). Two improvements to the rule-based part of the transfer have been added since Pilot 1:

- Definiteness detection has been added based on the old English synthesis block used in Pilot 1 (see also Section 2.3). Some rules were adapted to use t-trees only instead of combining t-tree and a-tree information.

  This change itself does not lead to an overall quality improvement, but moving the definiteness detection to the transfer phase corresponds better to the underlying theoretical foundation and allows us to use separate article assignment rules in other translation pairs (which improves their performance significantly, see Section 2.3).

- Noun groups reordering rules have been improved based on performance on Batch1 data.

2.5.3 Synthesis

The Czech synthesis pipeline has remained basically unchanged since Pilot 1. It has been used since the original TectoMT system [**?**] and tuned extensively throughout the years, and its performance is stable. The pipeline roughly conforms to the description in Section 2.2.7; a more detailed description can be found in D2.4.

2.6 Dutch: TectoMT

2.6.1 Analysis

Same as in Pilot 1, the analysis of Dutch input uses the Alpino system.[11] Alpino is an implementation of a stochastic attribute value grammar [**??**], details on the system are given in D2.4. In comparison with Pilot 1, a number of small changes have been implemented in Alpino which are relevant for the project. Those changes include several heuristics to analyze in-domain peculiarities such as the names of menu items, URL's, idiosyncratic named entities, etc. In addition, the disambiguation models of Alpino have been re-trained and perform slightly better than for Pilot 1.

In the following rule-based pipeline, the Alpino-parsed dependency trees are converted to a-layer-compatible trees and finally to t-trees. The t-tree conversion pipeline parallels other languages (see D2.4 and Section 2.2.3). The pipeline has undergone only minor changes since Pilot 1 which fix problems with punctuation and grammateme assignment.

2.6.2 Transfer

Both English-to-Dutch and Dutch-to-English transfer use discriminative and dictionary translation models as described in Section 2.2.4, with the newly added in-domain models. All the Pilot 1 rule-based transfer modules (see D2.7) have been kept in place, with the following ones newly added:

- Non-isomorphic translation of English noun groups (more t-nodes) into a Dutch compound (a single t-node) is improved: A post-processing module has been added that finds cases where TM decisions result in redundancies in the Dutch sentence (e.g., *web page* translates to *webpagina pagina*). The redundant nodes are removed from the translated t-tree (e.g., *web page* translates to *webpagina*).

- English-Dutch translation of relative clauses is now handled by a newly added rule, which enforces that relative clauses translate to relative clauses (the TM often assigns a plain finite clause formeme, causing word order errors).

- Grammatemes are checked for validity after transfer: If a t-tree node is translated to a different semantic part-of-speech, some grammatemes become invalid and may cause unwanted output (nouns inflected as verbs etc.). Invalid grammatemes are now removed in the Dutch-to-English direction, where the problem was most urgent. We are planning to include this module also in the other translation direction, where invalid grammatical features are currently mostly discarded in the Alpino generator.

2.6.3 Synthesis

Same as in Pilot 1, the Dutch synthesis pipeline adheres to the general synthesis scenario (see Section 2.2.7), including morphology initialization and agreements (subject-predicate and attribute-noun), insertion of prepositions and conjunctions based on formemes, and insertion of punctuation. Pilot 2 features an improved handling of possessive pronouns and Dutch pronominal adverbs (e.g., *waarin* 'in which', *ernaar* 'to that' etc.).

The final step of the Dutch synthesis, the generation of an actual sentence including word ordering and word form inflection, is handled by the Alpino generator [**?**]. The a-tree resulting from the previous steps is first converted to an Alpino Abstract Dependency

[11] http://www.let.rug.nl/vannoord/alp/Alpino/

Tree, which is then used as the input for Alpino generation (see D2.4 for more details). Similar to the techniques described in section D2.4 for improved Dutch synthesis, we have implemented further heuristics, inspired by several problems that were encountered in the development data.

2.7 Portuguese: TectoMT

2.7.1 Analysis

Same as in Pilot 1, the Portuguese analysis is performed by the LX-Suite toolchain [**?**] up to the a-layer, and then is rule-based up to the t-layer. The LX-Suite toolchain (earlier described in §3.7.1 of D2.4) comprises tokenization, lemmatization, morphological analysis, PoS tagging and dependency parsing. In comparison with Pilot 1, the Portuguese analysis has been slightly improved by adding missing lemmas to the PoS tagger and fixing some tokenization rules.

2.7.2 Transfer

Both English-to-Portuguese and Portuguese-to-English transfer use the discriminative and dictionary translation models as described in Section 2.2.4. Compared to Pilot 1, the transfer has been improved by interpolating (as described in Section 2.2.5) the general-domain models that were used in Pilot 1 (trained on the Europarl corpus [**?**]) with new IT domain models trained on a small parallel corpus composed of the QTLeap Batch 1, the Microsoft Terminology Collection[12] and the LibreOffice localization data.[13] Furthermore, we added some IT-domain-specific rules:

- Transfer of some IT-domain specific lemmas (such as "tab"↔"separador");
- Word reordering rules for dealing correctly with menu chains such as "go to Tools > Word Count";

2.7.3 Synthesis

Same as in Pilot 1, the Portuguese synthesis pipeline adheres to the general synthesis scenario described in Section 2.2.7. As described in section 3.7.3 of D2.4, the generation of appropriately inflected word forms is performed by LX-Inflector[14] for nominal expressions, and by LX-Conjugator [**?**] for verbal expressions. In comparison to Pilot 1, the Portuguese synthesis has been slightly improved by adding missing lemmas to these tools and by fixing the insertion of quotation marks in quoted expressions.

2.8 Spanish: TectoMT

2.8.1 Analysis

Same as in Pilot 1, the analysis of Basque input uses the IXA pipes tools.[15] So far, we have used Treex tokenization, IXA pipes modules for POS tagging and lemmatization, and Mate tools for dependency parsing. Details on the system are given in D2.4.

[12] Available from http://www.microsoft.com/Language/en-US/Terminology.aspx

[13] Available from https://www.libreoffice.org/community/localization/

[14] LX-Inflector is available online at
http://lxcenter.di.fc.ul.pt/services/en/LXServicesInflector.html

[15] http://ixa2.si.ehu.es/ixa-pipes/

Next, in the rule-based pipeline, the dependency trees are converted to a-layer-compatible trees and finally to t-trees. The t-tree conversion pipeline parallels other languages (see D2.4 and Section 2.2.3). Since Pilot 1, the pipeline has undergone changes on the treatment of reflexive verbs and the identification of multi-word prepositions and conjunctions.

2.8.2 Transfer

Both English-to-Spanish and Spanish-to-English transfer use discriminative and dictionary translation models as described in Section 2.2.4, with the newly added in-domain models. Additionally, in Pilot 2 the following rule-based transfer modules have been added:

- Rules for English-to-Spanish translation of relative clauses, which enforces the proper translation of the relative pronouns.
- Rules to address differences in definiteness between Spanish and English.
- Proper generation of the "There is" structure in English, from the Spanish counterpart.

2.8.3 Synthesis

Same as in Pilot 1, the Spanish synthesis pipeline adheres to the general synthesis scenario (see Section 2.2.7), including morphology initialization and agreements (subject-predicate and attribute-noun), insertion of prepositions and conjunctions based on formemes, and insertion of punctuation. Pilot 2 features an improved handling of word ordering and pronouns in reflexive verbs, and a gender lexicon to assign gender information to Spanish lemmas.

As described in D2.4, the final step of the Spanish synthesis, the generation of word forms, is performed by Flect **?**. Flect, the morphologic generation tool developed by CUNI, uses a morphologically annotated corpus to automatically learn how to inflect word forms based on lemmas and morphological features.

2.9 Bulgarian: Deep Factored MT

The analysis, transfer and generation are similar to Pilot 1 (described in D2.4). For Pilot 2, the main improving efforts were devoted to the analysis and transfer steps. For generation the same language model was used. This step was performed by the Moses setting.

We have conducted experiments with the addition of two types of lexical knowledge: morphological and semantic one. The first type of knowledge is represented by the construction of a bilingual Bulgarian-English inflectional lexicon, aligned on word-form level and grammatical features. The latter is the Bulgarian WordNet aligned to the Princeton WordNet, additionally extended for both languages with domain specific synsets.

2.9.1 Analysis

First, we performed experiments by adding information from the bilingual Bulgarian-English inflectional lexicon. The lexicon was constructed by exploiting the following resources: BTB-Morphological lexicon containing all word forms for more than 110 000

Bulgarian lemmas; BTB-bilingual Bulgarian-English lexicons (with about 8 000 entries); and English Wiktionary. The English word forms were extracted for the English lemmas. Then we mapped the word-form lexicons for both languages to the corresponding parts of the bilingual lexicon. Afterwards, the corresponding word forms were aligned on the basis of their morphological features like *number* (singular, plural); *degree* (comparative, superlative); *definiteness* (definite, indefinite), etc.

Bulgarian	**English**
visok|visok|a	a|a|d high|high|g
visok|visok|a	high|high|g
visok|visok|a	a|a|d tall|tall|g
visok|visok|a	tall|tall|g
naj-visokata|visok|a	highest|highest|g
naj-visokata|visok|a	the|the|d highest|highest|g
naj-visokata|visok|a	tallest|tallest|g
naj-visokata|visok|a	the|the|d tallest|tallest|g

Table 2: Wordform aligned parallel lexicon. It shows the Bulgarian adjective "visok" with its two translations in English: "high" and "tall". The table represents the encoding of singular, masculine, indefinite forms and superlative, singular, feminine, definite forms. Each triple represents *word form* | *lemma* | *part-of-speech*. For example, the triple a|a|d means: word form "a"; lemma "a" and part-of-speech "determiner".

In this preliminary experiment we used only the noun and the adjective parts-of-speech from the word-form aligned bilingual lexicon. Bulgarian language encodes definiteness as an ending to the nouns and adjectives in contrast to English which encodes it as a separate determiner in front of the noun or adjective. For this reason, we also encode the English definite and indefinite articles for the English word forms. Since in some contexts the English articles are not obligatory, the English word forms were encoded with or without them. Table 2 shows an example from the resulting lexicon.

The lexicon represents more than 70 000 aligned word forms. It was added to the training data. Each aligned pair of word forms is added as a pair of sentences with length one or two depending on determiners. We got the results presented in Table 3. They show a positive impact of the aligned word-form parallel lexicon on the translation in both directions. The table shows also that the addition of the definite forms for English does not improve the result.

lexicon	bg→en BLEU	en→bg BLEU
none	32.59	22.86
with indefinite forms only	33.02	23.91
with all forms	32.88	22.97

Table 3: Preliminary experiments with parallel morphological lexicons.

Additionally, we exploited the inflectional lexicon in experiments with semantic knowledge. For the exploitation of synsets from WordNet we used the UKB system provided by the Basque partner. An especially popular knowledge-based disambiguation approach

has been the use of successful graph-based algorithms known under the name of "Random Walk on Graph" **?**. Most methods exploit variants of the PageRank algorithm **?**. **?** apply a variant of the algorithm to Word Sense Disambiguation by translating WordNet into a knowledge graph in which the synsets are represented as vertices and the relations between them are represented as edges between the vertices.

For Bulgarian we exploited the knowledge graph for English WordNet via the mapping from Bulgarian to English WordNet. Additionally, semantic and syntactic relations from the sense annotated BulTreeBank have been extracted and the algorithm has been applied to Bulgarian data. In order to do that, the treebank was first annotated with synsets from the BulTreeBank WordNet[16], aligned to the Princeton WordNet. The word forms annotated with senses at this point are 69,333, consisting of nouns and verbs. Out of these, 12,792 sense-tagged word forms have been used for testing, and the rest have been used for relation extraction.

The WordNet ontological relations that have been used are 252,392, and the relations derived from the synset glosses are 419,387. Additionally, the following relations have been extracted: inferred hypernymy relations; syntactic relations from the gold corpus; extended syntactic relations; domain relations from WordNet. Thus, 590,272 new relations have been added. The newly added relations introduce syntagmatic information into the graph, which was originally constructed out of paradigmatic relations.

The results from the experiments with paradigmatic relations alone (done on the whole corpus) show highest accuracy (0.551) for the combination of: WordNet relations + relations from the glosses + inferred hypernymy relations + domain relations of the kind synset-to-synset + domain hierarchy relations. The results from the experiments with mixed – paradigmatic and syntagmatic – relations (done on a test portion of one fourth of the corpus) show highest accuracy (0.656) for the combination of: WordNet relations + relations from XWN + inferred hypernymy relations + dependency relations from the golden corpus + extended dependency relations starting from one level up + domain relations of the kind synset-to-synset + domain hierarchy relations.

The pipelines for English and Bulgarian were extended with UKB system working with two different knowledge graphs — the best combination of WordNet knowledge graph and semantic relations extracted from semantically annotated corpora and inferred relations.

The pipelines now performed the following tasks for Bulgarian: tokenization, sentence splitting, POS tagging, lemmatization, WSD, factor extraction. For English: tokenization, sentence splitting, POS tagging, lemmatization, WSD, parsing and coreference resolution, factor extraction. The main new addition in comparison to Pilot 1 is the WSD one. It is assigning to words in the corresponding WordNet the ILI identifier for the corresponding synset. For example, the word "plug-in" has ILI 03033986-n.

2.9.2 Transfer

After the annotation of the data we have performed substitution of original word forms with words in the target language. In the direction from Bulgarian to English we established correspondence between the morphological forms of the two languages. For example, plural Bulgarian nouns were substituted by plural English nouns, similarly for the other parts-of-speech. A given synset is always replaced by the word forms of the same lemma. This lemma is selected to be representative for the concept defined by the synset.

[16]The Core WordNet is freely available at: http://compling.hss.ntu.edu.sg/omw/. The extended one will be released soon. For more details about the sense annotated BulTreeBank, see **?**.

The selection was based on frequency. Thus, synsets with similar meanings like in cases of regular polysemy (city as a location and as a set of the people living in a city) have the same representative lemma in the target language. In many cases the translations of these related meanings are the same. In this way the sparseness of the word sense annotation representation was reduced.

2.10 German: Quality system combination

The fact that German is relatively well-resourced, in comparison to other language pairs, has allowed MT researchers to build strong statistical systems with very good performance on a lexical or a local level [**?**]. At the same time, rule-based MT systems like Lucy [**?**] are used successfully, yet only in certain niches today. The reason is that they lack recall: for example, parsing failure or gaps in the lexicon typically lead to a dead-end where the only option is to manually code the missing information, which is too resource intensive especially if one wants to take care of those less frequent items and phenomena in the "long tail". If one has the goal to use deep information for building an MT system that improves translation quality, there are two extreme options: (1) to start from a purely knowledge-driven approach and try to arrive at the same performance found in current SMT systems; (2) to start from an SMT system and try to arrive at higher precision by modifying it so that knowledge drives the search process. The system architecture we will describe below starts in the middle of both extreme options. It is a hybrid architecture that we use for experiments and extensions to increase MT quality by more knowledge-driven processing.

2.10.1 Overview of German Pilot 2

The German QTLeap Pilot 2 system is based on the Pilot 1 system documented in the project Deliverable D2.4. To make this Deliverable self-contained, we will briefly repeat information about all modules of the system.

The major change from Pilot 1 to Pilot 2 is the inclusion of a WSD module for English based on the system of [**?**] that feeds into a Moses that uses WSD information in an alternative path. Figure 2.10.1 shows the architecture of the German Pilot 2. The main components are:

- A WSD system,
- A Moses SMT system trained on WSD data,
- the transfer-based system Lucy,
- the serial system combination of Lucy and a Moses system, and
- an informed selection mechanism ("ranker").

As German is not part of WP5 in QTLeap and has thus not been able to produce WSD resources for German, we have focused the Pilot 2 developments on the "outbound" language direction en→de that is more relevant in the project and have left improvements of translation into English for Pilot 3.

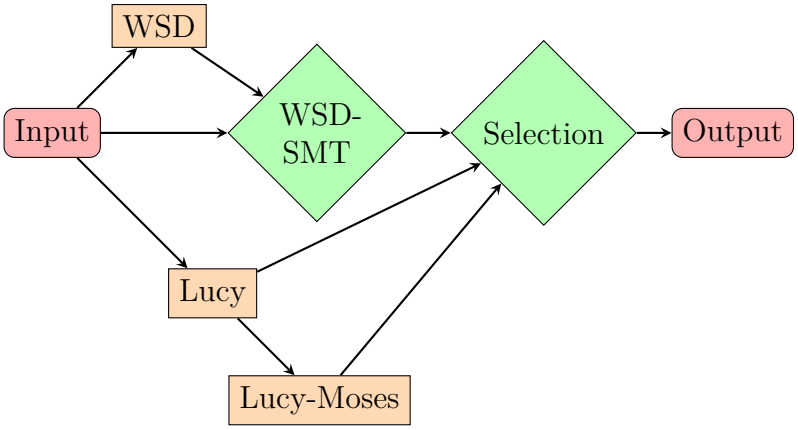


Figure 1: Architecture of German Pilot 2 Hybrid System

2.10.2 Translation systems

WSD and Moses. For Pilot 2, the simple phrase-based SMT of Pilot 1 is replaced by word-sense-disambiguated phrase-based SMT. It is a phrase-based system with two decoding paths, one basic and one alternative. In the basic path, all nouns of the source language (English) have been annotated by a Word Sense Disambiguation system [**?**] that assigns BabelNet-senses to nouns and has recently shown improvements over state-of-the-art results on several corpora. For use in QTLeap, we have developed and installed a REST-server version of the WSD system for online use.

The senses used for training and decoding by Moses are estimated based on the disambiguation analysis on the sentence level by choosing the best ranked sense from the WSD system. The produced WSD labels are concatenated with the respective base word forms. In the alternative path, non-annotated input is used. The alternative path allows for decoding phrases when there are no WSD labels or the decoder cannot form a translation with a good probability. Due to the high complexity of the WSD annotation, this model was trained on less data than the respective phrase-based models for Pilot 0 and Pilot 1. We ran Pilot 2 experiments where the senses are used as alternative paths in Moses with four translation setting:

1. Baseline w/o WSD
2. Sense → Word
3. Word → word, Sense → word (alt path)
4. Sense → word, Word → word (alt path)

The results of pilot experiments were promising as we got 1 BLEU score improvement for Batch3q and 0.4 BLEU score improvement for Batch1q for setting 4. Interestingly enough, the best results were achieved on questions rather than answers. For the QTLeap test set Batch3a, we obtain the results in table Table 4. On this set, WSD does not show a positive effect in terms of BLEU. Please refer to Deliverable 5.7 for more details on the contribution of WSD.

Moses variant	BLEU
0. Pilot 0	34.82
1. Pilot 2 Moses baseline	27.41
2. Pilot 2 Sense → word	22.56
3. Pilot 2 Word → word, Sense → word (alt path)	26.80
4. Pilot 2 Sense → word, Word → word (alt path)	27.05

Table 4: BLEU scores for Moses variants used in German (en→de) hybrid architecture (tested on Batch3a)

Lucy. The transfer-based Lucy system [**?**] includes the results of long linguistic efforts over the last decades and that has been used in previous projects including EuroMatrix, EuroMatrix+ and QTLaunchPad, while relevant hybrid systems have been submitted to WMT [**???**]. The transfer-based approach has shown good results that compete with pure statistical systems, whereas it focuses on translating according to linguistic structures. Its functionality is based on hand-written linguistic rules and there are no major empirical components. Translations are processed in three phases:

- the **analysis phase**, where the source-language text is parsed and a tree of the source language is constructed,
- the **transfer phase**, where the analysis tree is used for the transfer phase, where canonical forms and categories of the source are transferred into similar representations of the target language,
- the **generation phase**, where the target sentence is formed out of the transferred representations by employing inflection and agreement rules.

LucyMoses. As an alternative way of automatic post-editing of the transfer-based system, a serial transfer+SMT system combination is used, as described in **?**. For building it, the first stage is translation of the source language part of the training corpus by the transfer-based system. In the second stage, an SMT system is trained using the transfer-based translation output as a source language and the target language part as a target language. Later, the test set is first translated by the transfer-based system, and the obtained translation is translated by the SMT system. In previous experiments, however, the method on its own could not outperform Moses trained on a large parallel corpus (Pilot 0).

2.10.3 Empirical machine learning classifier for sentence selection

The machine learning selection mechanism is based on encouraging results of previous projects including EuroMatrix+ [**?**], META-NET [**?**], QTLaunchPad [**??**]. It has been extended to include several features that can only be generated on a sentence level and would otherwise blatantly increase the complexity of the transfer or decoding algorithm. In the architecture at hand, automatic syntactic and dependency analysis is employed on a sentence level, in order to choose the sentence that fulfills the basic quality aspects of the translation: (a) assert the fluency of the generated sentence, by analyzing the quality of its syntax (b) ensure its adequacy, by comparing the structures of the source with the structures of the generated sentence.

All produced features are used to build a machine-learned ranking mechanism (ranker) against training preference labels. Preference labels are part of the training data and rank different system outputs for a given source sentence based on the translation quality. Preference labels are generated either by automatic reference-based metrics, or derived from human preferences. The ranker was a result of experimenting with various combinations of feature sets and machine learning algorithms and choosing the one that performs best on the development corpus.

The implementation of the selection mechanism is based on the "Qualitative" toolkit that was presented at the MT Marathon, as an open-source contribution by QTLeap [**?**].

Feature sets. We experimented with feature sets that performed well in previous experiments. In particular:

- Basic syntax-based feature set: unknown words, count of tokens, count of alternative parse trees, count of verb phrases, PCFG parse log likelihood. The parsing was performed with the Berkeley Parser [**?**] and features were extracted from both source and target. This feature set has performed well as a metric in WMT-11 metrics task [**?**].

- Basic feature set + 17 QuEst baseline features: this feature set combines the basic syntax-based feature set described above with the baseline feature set of the QuEst toolkit [**?**] as per WMT-13 [**?**]. This feature set combination got the best result in WMT-13 quality estimation task [**?**]. The 17 feature set includes shallow features such as the number of tokens, LM probabilities, number of occurrences of the target work within the target probability, average numbers of translations per source word in the sentence, percentages of unigrams, bigrams and trigrams in quartiles 1 and 4 of frequency of source words in a source language corpus and the count of punctuation marks.

- Basic syntax-based feature set with Bit Parser: here we replace the Berkeley parser features on the target side with Bit Parser.

- Advanced syntax-based feature set: this augments the basic set by adding IBM model 1 probabilities, full depth of parse trees, depth of the 'S' node, position of the VP and other verb nodes from the beginning and end of the parent node, count of unpaired brackets and compound suggestions (for German, as indicated by LanguageTool.org).

Machine Learning. As explained above, the core of the selection mechanism is a ranker which reproduces ranking by aggregating pairwise decisions by a binary classifier [**?**]. Such a classifier is trained on binary comparisons in order to select the best one out of two different MT outputs given one source sentence at a time. As a training material, we used the evaluation dataset of the WMT shared tasks (years 2008-2014), where each source sentence was translated by many systems and their outputs were consequently ranked by human annotators. These preference labels provided the binary pairwise comparisons for training the classifiers. Additionally to the human labels, we also experimented on training the classifiers against automatically generated preference labels, after ranking the outputs with METEOR [**?**]. In each translation direction, we chose the label type (human vs. METEOR) which maximizes if possible all automatic scores on our development set, including document-level BLEU.

We exhaustively tested all suggested feature sets with many machine learning methods, including Support Vector Machines (with both RBF and linear kernel), Logistic Regression, Extra/Decision Trees, k-neighbors, Gaussian Naive Bayes, Linear and Quadratic Discriminant Analysis, Random Forest and AdaBoost ensemble over Decision Trees. The binary classifiers were wrapped into rankers using the *soft pairwise recomposition* [**?**] to avoid ties between the systems. When ties occurred, the system selected based on a predefined system priority (Lucy, Moses, LucyMoses). The predefined priority was defined manually based on preliminary observations in order to prioritize the transfer-based system, due to its tension to achieve better grammaticality. Further analysis on this aspect may be required.

Best combination. The optimal system for Pilot 2 has been trained with Support Vector Machines against METEOR scores. METEOR was chosen since for this language pair, the empirical mechanism trained on human judgments had very low performance in term of correlation with humans.

A variant of the Pilot 1 system (using the same selection mechanism as Pilot 2) trained on WMT data has participated in WMT 2015 (**?**). The paper also contains a detailed evaluation and some additional experiments we performed on the WMT data.

3 Intrinsic Evaluation

This section describes the intrinsic evaluation of the Pilot 2 results, starting with automatic measures and then describing a manual evaluation study.

3.1 Automatic Evaluation

We provide (case-insensitive) BLEU and (case-sensitive) word-level F-scores (wordF) in Tables 5 through 9 for the test corpus (which is QTLeap Batch 3 for the evaluation of Pilot2). Similarly to Pilot1 evaluation in D2.4, scores have been computed using the official BLEU script `mteval-v13a.pl -international-tokenization` and `rgbF.py` as implemented in the QTLeap Evaluation Workbench. The best system in each column (for BLEU tables only) is marked in bold if it is significantly ($p < 0.05$, using bootstrap resampling) better than the remaining two systems; otherwise (insignificantly best) it is marked in italics.

The Pilot0 results for en→es and es→en reported here are Pilot0-comparable, that is Pilot0 trained on Europarl only, so it can be fairly compared with Pilot1 and Pilot2, which are also trained on Europarl only.

In order to test the pilots in another usage scenario, we have decided to use WMT data ("News part of the QTLeap corpus", see D3.13) as out-of-domain test data for our pilots. BLEU results are reported in Tables 7 and 10.

Translation into English

system	bg→en	cs→en	de→en	es→en	eu→en	nl→en	pt→en
Pilot0	*22.56*	19.03	**32.02**	**24.47**	**11.94**	25.57	**17.14**
Pilot1	22.06	17.01	27.47	12.77	4.43	20.79	8.94
Pilot2	21.81	**20.53**		18.64	6.79	**27.09**	11.59

Table 5: BLEU scores of Pilot0 (baseline), Pilot1 and Pilot2 on translations into English of Batch3q (questions) part of the QTLeap Corpus.

system	bg→en	cs→en	de→en	es→en	eu→en	nl→en	pt→en
Pilot0	28.91	26.34	37.21	31.08	20.00	32.52	25.03
Pilot1	28.53	26.19	33.97	21.38	11.69	28.21	18.05
Pilot2	28.25	28.91		26.66	15.64	33.43	21.19

Table 6: F-scores of Pilot0 (baseline), Pilot1 and Pilot2 on translations into English of Batch3q (questions) part of the QTLeap Corpus.

system	bg→en	cs→en	de→en	es→en	eu→en	nl→en	pt→en
Pilot0	**18.05**	**24.03**	**26.26**	**27.53**	**8.84**	**23.50**	**21.85**
Pilot1	17.72	12.07	17.77	8.78	2.40	11.33	6.69
Pilot2	17.30	13.04		13.55	3.07	19.40	7.55

Table 7: BLEU scores of Pilot0 (baseline), Pilot1 and Pilot2 on translations into English of News part of the QTLeap Corpus

Translation from English

system	en→bg	en→cs	en→de	en→es	en→eu	en→nl	en→pt
Pilot0	**17.72**	21.34	**34.82**	16.23	**18.59**	**25.98**	13.75
Pilot1	16.36	20.44	31.56	10.73	9.62	18.15	12.86
Pilot2	16.91	*21.89*	29.57	**24.32**	11.27	19.66	**15.51**

Table 8: BLEU scores of Pilot0 (baseline), Pilot1 and Pilot2 on translations from English of Batch3a (answers) part of the QTLeap Corpus.

system	en→bg	en→cs	en→de	en→es	en→eu	en→nl	en→pt
Pilot0	23.77	27.85	39.78	22.92	25.46	31.33	20.70
Pilot1	22.51	27.31	36.80	18.40	16.83	25.03	20.16
Pilot2	22.90	28.73	34.61	31.52	17.92	26.38	23.49

Table 9: F-scores of Pilot0 (baseline), Pilot1 and Pilot2 on translations from English of Batch3a (answers) part of the QTLeap Corpus.

system	en→bg	en→cs	en→de	en→es	en→eu	en→nl	en→pt
Pilot0	**15.45**	**17.57**	**17.41**	**29.96**	**5.36**	**19.66**	**21.85**
Pilot1	14.48	12.40	16.73	9.24	2.09	12.72	8.64
Pilot2	14.65	14.36	12.89	13.60	2.10	13.72	7.60

Table 10: BLEU scores of Pilot0 (baseline), Pilot1 and Pilot2 on translations from English of News part of the QTLeap Corpus.

In terms of automatic scores in the project's technical domain, the vast majority of Pilot2 systems have improved over Pilot1. Still, it remains difficult for some deeper pilots to beat the baseline of Pilot0. Comparison between the QTLeap technical domain corpus and news corpus shows generally considerably lower scores than the Pilot0 baselines on the latter. One reason might be missing domain optimization, another reason might be that the sentences in the news domain are generally long and complex and the number of topics touched is broad.

Effect of individual modules

In Tables 11 and 12, we analyze the effect of individual modules in Pilot2 systems on the BLEU score. These results were evaluated on Batch2, which was used as a development dataset (unlike the results reported above, which were evaluated on Batch3, which was kept secret until the final evaluation of Pilot2). Tables 11 and 12 are reported also in D5.7, but we report them here as well, to keep this deliverable self-contained.

The row denoted as Pilot2-minus-LS shows BLEU scores of the Pilot 2 systems, if all the lexical semantics components are switched off. Each row that follows presents one of the lexical semantics component and what is the effect of switching on just this single component in the Pilot2-minus-LS system. The "Δ total LS" row shows the effect of switching on all the components (except for nl→en gazetteers; see below). This difference in scores is usually not a sum of the differences for individual components, as the effects of these components may overlap. The final performance of the full Pilot 2 systems can be found in the last row of the tables. Not all lexical semantics components have been switched on in the full Pilot 2. Using a gazetteer in the nl→en translation deteriorate the score, so we decided to deactivate the gazetteer module in full Pilot 2 for nl→en.

system	bg→en	cs→en	es→en	eu→en	nl→en	pt→en
Pilot0	29.66	26.44	39.30	25.29	36.45	22.59
Pilot1	27.70	26.81	16.05	4.75	34.46	10.14
Pilot2-minus-LS		27.78	26.21	13.30	44.01	11.94
Δ "fixed" entities (HideIT)		−0.01	+0.01	+0.03	+0.00	+0.01
Δ specialized lexicons (gazetteers)		+0.77	+0.62	+0.00	−0.09	+0.02
Δ adaptation by TM interpolation		+1.67	+0.42	+0.71	+1.91	+1.50
Δ total LS		+2.50	+0.94	+0.77	+1.92	+1.57
full Pilot2	25.69	30.28	27.15	14.07	45.93	13.51

Table 11: Translations to English (Batch2q). Effect of various lexical semantic modules on BLEU performance.

system	en→bg	en→cs	en→es	en→eu	en→nl	en→pt
Pilot0	25.11	31.07	25.11	28.37	32.94	19.36
Pilot1	24.15	30.68	16.92	14.39	23.10	19.34
Pilot2-minus-LS		28.07	26.25	20.87	23.38	19.82
Δ +synset&supersense(node,parent)						+0.25
Δ "fixed" entities (HideIT)		+0.84	+0.46	+0.56	+0.48	+0.34
Δ specialized lexicons (gazetteers)		+3.49	+3.19	+0.91	+1.49	+0.94
Δ adaptation by TM interpolation		+0.74	+5.10	+0.06	+0.75	+1.98
Δ total LS		+4.97	+7.91	+1.46	+2.45	+2.60
full Pilot2	22.76	33.04	34.16	22.33	25.83	22.42

Table 12: Translations from English (Batch2a). Effect of various lexical semantic modules on BLEU performance.

For some language pairs, the highest improvement (e.g. about 10 BLEU points for en→es, eu→en and nl→en) stems from the "from Pilot1 to Pilot2-minus-LS" improvements (which are described in Sections 2.8, 2.4 and 2.6, respectively). For other language pairs, the highest improvement stems from the three lexical-semantics-related modules described in D5.7 (HideIT and gazetteers) and Section 2.2.5 (TM interpolation).

3.2 Manual Evaluation

In this task, volunteers from within the project were asked to annotate sentences from both Pilot 1 and Pilot 2 using a selection of issues types taken from the Multidimensional Quality Metrics (MQM) framework. The issues are a modification from the selection used in D2.4, updated to reflect knowledge and experience gained in the QT21 project. The primary difference in the issues are:

1. *Terminology* was moved to become a high-level issue, separate from *Mistranslation*;
2. *Overly literal* was added as a subtype of *Mistranslation* to account for cases in which the translation was a possible literal translation of the words but nevertheless failed to capture the intended meaning;
3. *Locale convention* was added as a main branch to account for instances in which items such as dates, times, addresses, telephone numbers, or names were rendered contrary to the conventions of the target locale.

Despite these differences, the results of D2.8 and D2.4 are largely comparable.

Annotators were asked to annotate a minimum of 25 segments (they could do more if they wished to), annotating the results from both Pilot 1 and Pilot 2 using translate5, where they appeared in separate columns. To prevent any bias that might come if the annotators knew which segment was from which pilot, we randomized which column the results appeared in, so that each column had roughly equal numbers of results from each pilot. The number of annotators and number of segments annotated for each language are as follows:

- Basque: 1 (26 segments annotated)
- Bulgarian: 4 (28 segments double annotated + 66 single annotated)
- Czech: 1 (30 segments annotated)
- Dutch: 1 (25 segments annotated)
- German: 1 (28 segments annotated)
- Portuguese: 2 (25 segments double annotated)
- Spanish: 1 (25 segments annotated)

The annotators were instructed to skip sentences that were too difficult to annotate or that required no annotation and to indicate their reason using the Notes feature of translate5. In most cases, however, the annotators did not indicate why they skipped specific sentences: we only found out after annotation was complete that there had been a misunderstanding about how the notes feature works, and it is likely that most such

notes were lost as a result. Therefore we do not know if skipped sentences were too difficult to annotate or needed no annotation. However, based on feedback from the annotators, in most cases it would appear that skipped sentences were simply too difficult to annotate.

As it was an option to skip annotating one of the options if it was too difficult or if it was perfect, the number of segments annotated from each Pilot varies.

- In four of the seven cases more segments from Pilot 2 were skipped:
  - Basque (Pilot 1: 26; Pilot 2: 25)
  - Czech (Pilot 1: 30; Pilot 2: 25)
  - German (Pilot 1: 28; Pilot 2: 27)
  - Spanish (Pilot 1: 25; Pilot 2: 23)
- In two of the seven cases more segments from Pilot 1 were skipped:
  - Bulgarian (Pilot 1: 94; Pilot 2: 96)[17]
  - Dutch (Pilot 1: 24; Pilot 2: 25)
- In the case of Portuguese both annotators annotated both results for all 25 segments.

As the option of skipping segments has been used frequently, the annotation results provide information about the relative occurrence of errors in translations from Pilot 1 and Pilot 2 on segments that belong to the better segments. This information is relevant, e.g., if one thinks of the usage of MT in a production setting (possibly with post-editing options) where only the better MT segments can be used while the absence of certain errors, the overall number of errors, etc. must be tightly controlled. In our development setting, this information provides insights into the qualitative nature of errors and provides starting points for system improvements.

To compare the pilots we were interested in the "density" of each error type rather than the total number (which would vary with the number of segments annotated). To calculate this number we counted the total number of instances of each MQM issue type, multiplied it by 100 and divided by the total number of segments annotated. This calculation does *not* provide the percentage of segments exhibiting a given error (since a segment can have multiple instances of an error), but rather gives us an approximation of how many instances of the error would occur in 100 segments. It is thus only superficially similar to a percentage.

The density figures for each language for the top-level MQM categories are presented in Figure 2. In this case a lower number for a given column between Pilot 1 and Pilot 2 represents an improvement in this particular category (i.e., fewer instances of an issue type were found in Pilot 2); conversely a higher number can represent a decrease in performance. Because most issues appeared only in very small numbers, statistical significance cannot be demonstrated and small changes generally cannot be interpreted as significant in any way. The results are thus only indicative in nature.

It is critical to note that numbers cannot *be compared across languages. For example, it might appear that Portuguese has many more errors than the other languages, but differences in annotation style such as when to skip segments and other factors render*

[17]Note that three of the four annotators in Bulgarian did not annotate both columns, but only one instead, so these numbers cannot be directly compared with the other cases.

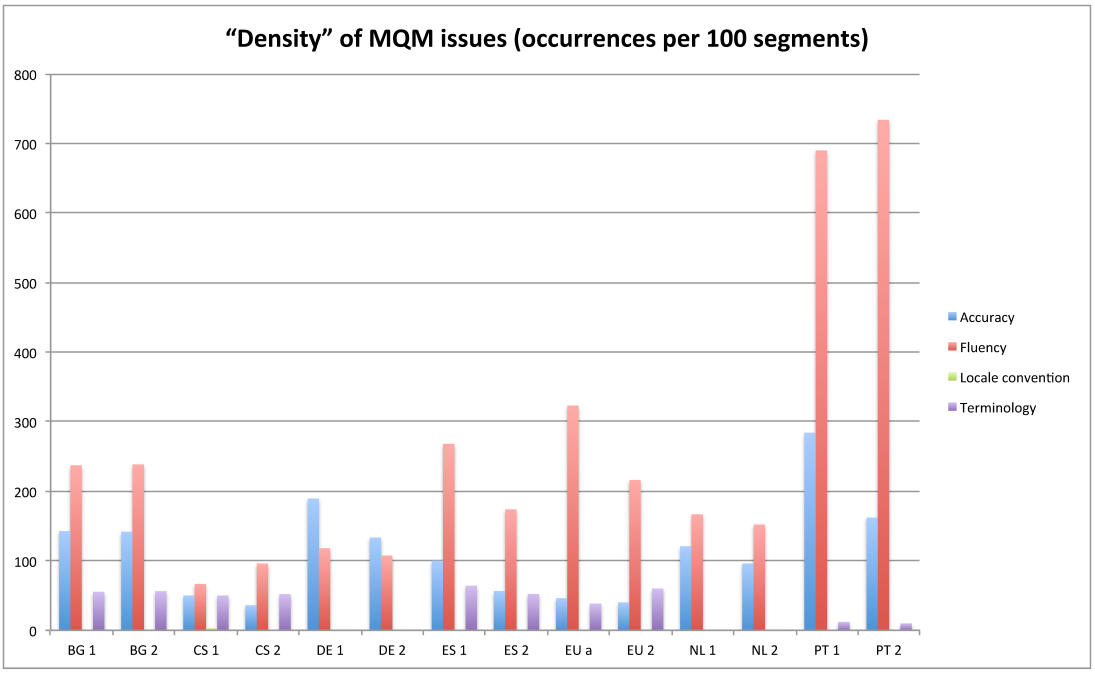


Figure 2: Density of high-level MQM issues for segments annotated from Pilot 1 and Pilot 2 for all languages.

any such comparison pointless: the only valid comparison is between Pilot 1 and Pilot 2 within a given language.

To compare relative performance, we then subtracted the number of instances of each type in Pilot 1 from the number in Pilot 2. Negative numbers thus show a relative improvement for a specific issue type and positive numbers show an increase in the occurrences of the issue. The numbers are also realized as percentage changes (see Figure 3). Percentages cannot be calculated for instances in which no instances of a given issue type were found in Pilot 1, but if both numbers were 0, a value of 0% is shown. Care should be taken in using these percentage changes since statistically insignificant differences in issues with relatively low numbers can appear as large percentages.

In this view, it can be seen that, with limited exceptions, Pilot 2 systems show improvements over Pilot 1. The prominent increase in *Terminology* for Basque represents an absolute increase from 10 to 15 instances and indicates how this small study may magnify relatively small differences. Similarly, the seemingly dramatic improvement in Locale convention for Czech is actually a difference of 1 (Pilot 1 had one instance and Pilot 2 had 0). Despite the limitations of these figures when based on small numbers, they do indicate overall improvements between Pilot 1 and Pilot 2. The full annotation results can be found in Tables 13 and 14.

When inspecting the annotations qualitatively, it becomes obvious that sometimes improvements of certain issues come with a certain loss of performance at other places, e.g., when comparing the German Pilot2 and Pilot1 (see Figure 4 and 5) on the following example:

Source Go to Tools and then choose 'Delete browsing history..', you can then choose to delete your Internet cookies.

Reference Gehen Sie zu Extras und wählen Sie dann 'Löschen der Browser Geschichte .. ', dann können Sie wählen, ob Sie Ihre Internet-Cookies löschen möchten.

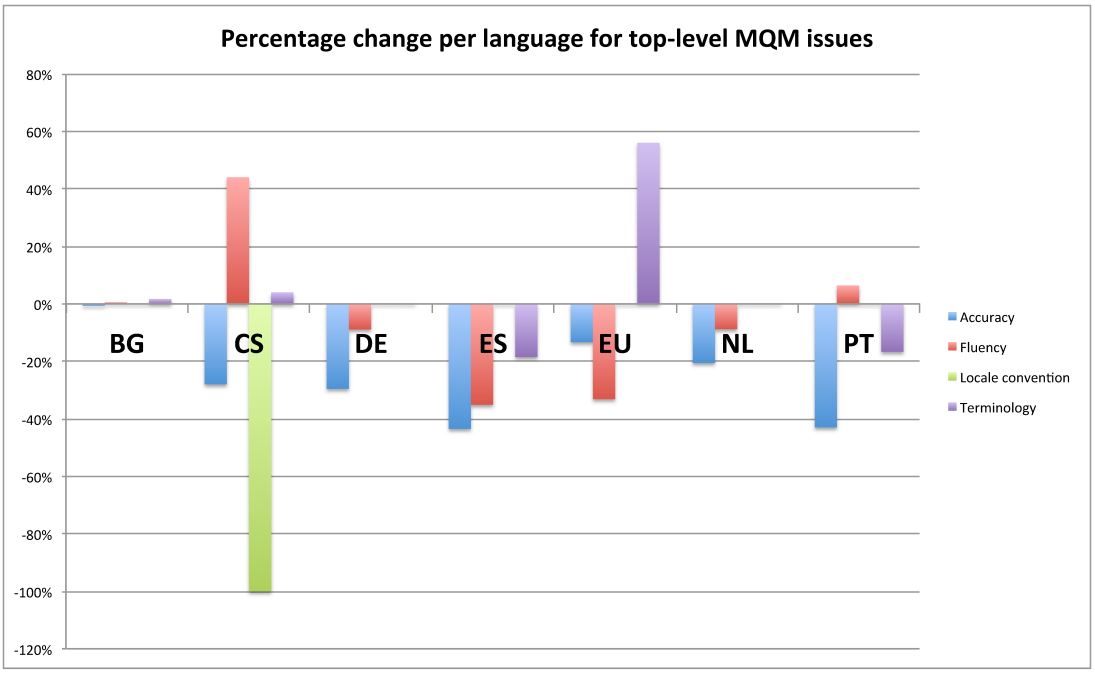


Figure 3: Percentage change in density of high-level MQM issue types from Pilot 1 to Pilot 2 (negative numbers indicate better performance).

one can see that Pilot2 gets the term "Tools" right ("Extras") while at the same time failing on the imperative *Go to* and omitting the information that one can *choose* to delete cookies.

3 (DE)	**Gehen Sie zu Tools und wählen Sie dann Browsingchronik Löschen.., können Sie dann vorziehen, Ihre Internet-Cookies zu löschen.**		
	deA [[1] Gehen Sie zu] [[2] Tools] und wählen Sie dann [[3] Browsingchronik] [[4] Löschen].., [[5] können] Sie dann [[6] vorziehen], Ihre Internet-Cookies zu löschen.	6	1. **Mistranslation** [Gehen Sie zu] 2. **Untranslated** [Tools] 3. **Mistranslation** [Browsingchronik] 4. **Part of speech** [Löschen] 5. **Word order** [können] 6. **Mistranslation** [vorziehen]

Figure 4: MQM annotated output of Pilot1.

3 (DE)	**Sprung zu Extras und wählen Sie dann Browserverlauf löschen,.., Sie können dann Ihre Internet-Cookies löschen.**		
	deA [[1] Sprung zu] Extras und wählen Sie dann Browserverlauf löschen,..[[2] ,] Sie können dann[[3]]Ihre Internet-Cookies löschen.	3	1. **Mistranslation** [Sprung zu] 2. **Typography** [,] 3. **Omission** [ ]

Figure 5: MQM annotated output of Pilot2.

4 Conclusion

In this deliverable, we described the improvements of the entry-level deep MT systems (Pilot 1, see D2.4) that led to the current Pilot 2 systems (enhanced with lexical-semantics modules). The changes were pushed in two directions. This deliverable describes general development and enhancements of the systems, as most of them were newly created within the QTLeap project and therefore many of their components were still very basic or even rudimentary in Pilot 1. The other direction of improvements followed in Pilot 2 was

MQM issue	Bulgarian (BG)				Czech (CZ)				German (DE)				Spanish (ES)			
	P1	P2	Diff.	%	P1	P2	Diff.	%	P1	P2	Diff.	%	P1	P2	Diff.	%
Accuracy	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Addition	16.0	7.3	-8.7	-54%	13.3	20.0	6.7	50%	3.6	11.1	7.5	211%	0.0	0.0	0.0	0%
Mistranslation	73.4	70.8	-2.6	-4%	30.0	8.0	-22.0	-73%	121.4	111.1	-10.3	-8%	4.0	17.4	13.4	335%
Omission	3.2	7.3	4.1	128%	0.0	0.0	0.0	0%	46.4	7.4	-39.0	-84%	0.0	0.0	0.0	0%
Overly literal	14.9	14.6	-0.3	-2%	3.3	0.0	-3.3	-100%	3.6	3.7	0.1	4%	0.0	4.3	4.3	NA
Untranslated	35.1	41.7	6.6	19%	3.3	8.0	4.7	140%	14.3	0.0	-14.3	-100%	96.0	34.8	-61.2	-64%
Accuracy sub.	142.6	141.7	-0.9	-1%	50.0	36.0	-14.0	-28%	189.3	133.3	-56.0	-30%	100.0	56.5	-43.5	-43%
Fluency	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Grammatical register	2.1	2.1	0.0	-2%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Spelling	12.8	12.5	-0.3	-2%	6.7	8.0	1.3	20%	14.3	14.8	0.5	4%	48.0	8.7	-39.3	-82%
Typography	12.8	7.3	-5.5	-43%	0.0	0.0	0.0	0%	0.0	3.7	3.7	NA	0.0	0.0	0.0	0%
Grammar	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	3.7	3.7	NA	0.0	0.0	0.0	0%
Word form	20.2	29.2	9.0	44%	3.3	8.0	4.7	140%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Part of speech	18.1	24.0	5.9	32%	3.3	4.0	0.7	20%	10.7	0.0	-10.7	-100%	36.0	4.3	-31.7	-88%
Agreement	61.7	51.0	-10.7	-17%	0.0	8.0	8.0	NA	14.3	14.8	0.5	4%	36.0	8.7	-27.3	-76%
Tense/aspect/mood	14.9	17.7	2.8	19%	6.7	8.0	1.3	20%	7.1	3.7	-3.4	-48%	8.0	26.1	18.1	226%
Word form sub.	114.9	121.9	7.0	6%	13.3	28.0	14.7	110%	32.1	18.5	-13.6	-42%	80.0	39.1	-40.9	-51%
Word order	14.9	18.8	3.9	26%	6.7	20.0	13.3	200%	25.0	11.1	-13.9	-56%	52.0	30.4	-21.6	-41%
Function words	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Extraneous	11.7	26.0	14.3	123%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	4.0	30.4	26.4	661%
Incorrect	20.2	16.7	-3.5	-18%	3.3	0.0	-3.3	-100%	28.6	33.3	4.8	17%	20.0	17.4	-2.6	-13%
Missing	19.1	19.8	0.6	3%	10.0	12.0	2.0	20%	17.9	18.5	0.7	4%	64.0	47.8	-16.2	-25%
Function words sub.	51.1	62.5	11.4	22%	13.3	12.0	-1.3	-10%	46.4	51.9	5.4	12%	88.0	95.7	7.7	9%
Unintelligible	28.7	13.5	-15.2	-53%	26.7	28.0	1.3	5%	0.0	3.7	3.7	NA	0.0	0.0	0.0	0%
Fluency sub.	237.2	238.5	1.3	1%	66.7	96.0	29.3	44%	117.9	107.4	-10.4	-9%	268.0	173.9	-94.1	-35%
Locale convention	0.0	0.0	0.0	0%	3.3	0.0	-3.3	-100%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Terminology	55.3	56.3	0.9	2%	50.0	52.0	2.0	4%	0.0	0.0	0.0	0%	64.0	52.2	-11.8	-18%

Table 13: MQM Density data for Bulgarian, Czech, German, and Spanish (negative numbers in columns Diff. and % indicate improved performance)

MQM issue	Basque (EU)				Dutch (NL)				Portuguese (PT)			
	P1	P2	Diff.	%	P1	P2	Diff.	%	P1	P2	Diff.	%
Accuracy	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Addition	15.4	8.0	-7.4	-48%	4.2	12.0	7.8	188%	8.0	6.0	-2.0	-25%
Mistranslation	0.0	0.0	0.0	0%	100.0	68.0	-32.0	-32%	178.0	82.0	-96.0	-54%
Omission	23.1	28.0	4.9	21%	4.2	0.0	-4.2	-100%	46.0	36.0	-10.0	-22%
Overly literal	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	30.0	16.0	-14.0	-47%
Untranslated	7.7	4.0	-3.7	-48%	12.5	16.0	3.5	28%	22.0	22.0	0.0	0%
Accuracy sub.	46.2	40.0	-6.2	-13%	120.8	96.0	-24.8	-21%	284.0	162.0	-122.0	-43%
Fluency	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	2.0	2.0	0.0	0%
Grammatical register	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Spelling	80.8	24.0	-56.8	-70%	0.0	0.0	0.0	0%	48.0	40.0	-8.0	-17%
Typography	3.8	0.0	-3.8	-100%	0.0	0.0	0.0	0%	46.0	54.0	8.0	17%
Grammar	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Word form	3.8	0.0	-3.8	-100%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Part of speech	34.6	28.0	-6.6	-19%	4.2	0.0	-4.2	-100%	34.0	34.0	0.0	0%
Agreement	11.5	12.0	0.5	4%	66.7	32.0	-34.7	-52%	56.0	52.0	-4.0	-7%
Tense/aspect/mood	19.2	12.0	-7.2	-38%	4.2	0.0	-4.2	-100%	56.0	100.0	44.0	79%
Word form sub.	69.2	52.0	-17.2	-25%	75.0	32.0	-43.0	-57%	146.0	186.0	40.0	27%
Word order	42.3	16.0	-26.3	-62%	58.3	68.0	9.7	17%	74.0	66.0	-8.0	-11%
Function words	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Extraneous	11.5	40.0	28.5	247%	4.2	8.0	3.8	92%	112.0	110.0	-2.0	-2%
Incorrect	42.3	36.0	-6.3	-15%	12.5	28.0	15.5	124%	52.0	32.0	-20.0	-38%
Missing	73.1	48.0	-25.1	-34%	8.3	8.0	-0.3	-4%	210.0	244.0	34.0	16%
Function words sub.	126.9	124.0	-2.9	-2%	25.0	44.0	19.0	76%	374.0	386.0	12.0	3%
Unintelligible	0.0	0.0	0.0	0%	8.3	8.0	-0.3	-4%	0.0	0.0	0.0	0%
Fluency sub.	323.1	216.0	-107.1	-33%	166.7	152.0	-14.7	-9%	690.0	734.0	44.0	6%
Locale convention	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%	0.0	0.0	0.0	0%
Terminology	38.5	60.0	21.5	56%	0.0	0.0	0.0	0%	12.0	10.0	-2.0	-17%

Table 14: MQM Density data for Basque, Dutch, and Portuguese (negative numbers in columns Diff. and % indicate improved performance)

incorporation of semantic linking and resolving techniques; they are described in D5.7, which also reports the BLEU score deltas of each technique.

A thorough evaluation of the Pilot 2 systems on Batch 3 test set from the QTLeap corpus showed tremendous increases in translation quality achieved by the systems. Not only do nearly all Pilot 2 systems provide higher quality translation than their Pilot 1 versions, but for four translation directions (cs→en, nl→en, en→es, en→pt), the improvement over the Pilot 1 system is large enough to also newly significantly outperform Pilot 0 (note that evaluation of the Pilot 1 versions of these four systems on Batch 3 does not indicate them outperforming Pilot 0, i.e. this is a new achievement). For the systems that seem to still be below Pilot 0, we generally observed steady improvements in translation quality across multiple dimensions, and we believe that most of them will reach this goal in Pilot 3.