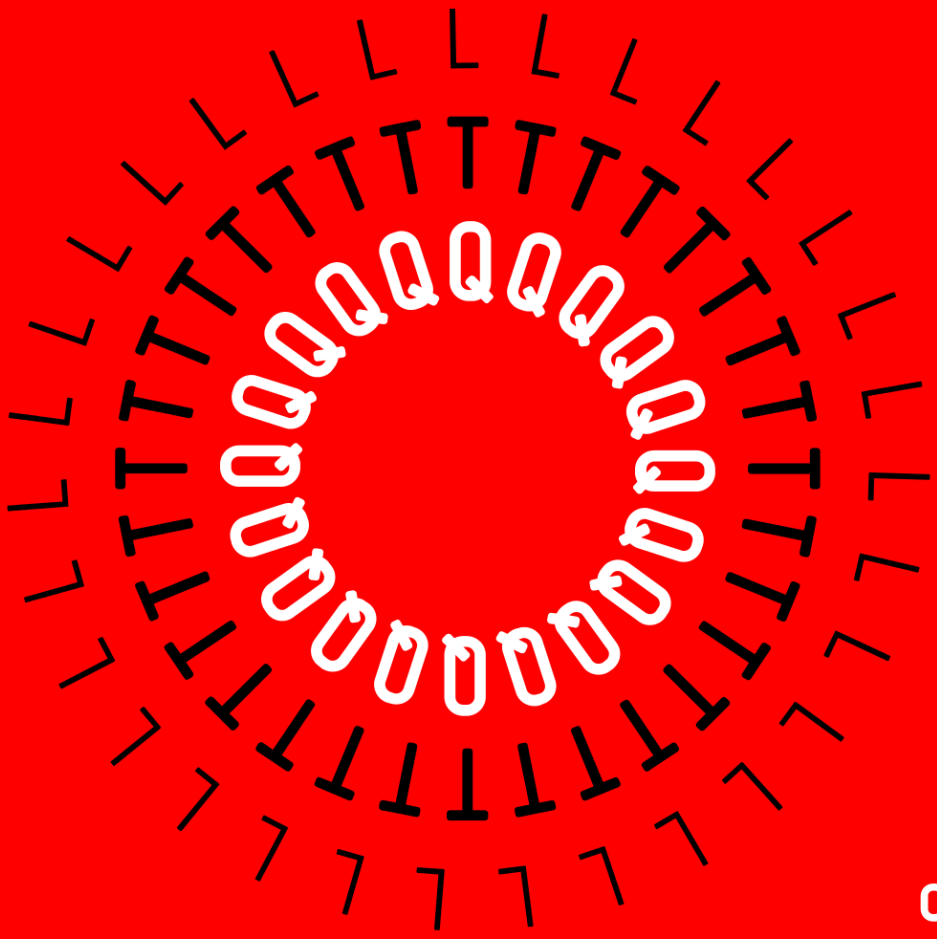


qtleap
quality translation by deep language engineering approaches

Report on the first MT pilot and its evaluation

DELIVERABLE D2.4

VERSION 1.8 | 2015-04-02

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:

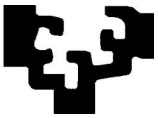
Faculty of Sciences, University of Lisbon

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, 2014	Ondřej Bojar, Ondřej Dušek, Michal Novák, Martin Popel, Rudolf Rosa, Aleš Tamchyna	CUNI	First draft of Sections 1, 3.1. Czech and English components (3.5,3.3).
0.2	Oct 9, 2014	Ondřej Dušek	CUNI	Dutch components (3.6).
0.3	Oct 9, 2014	Eleftherios Avramidis, Aljoscha Burchardt, Maja Popovic	DFKI	first draft of Section 4. German components (3.10).
0.4	Oct 10, 2014	Gorka Labaka, Nora Aranberri	UPV/EHU	Basque (3.4) and Spanish (3.8) components.
0.5	Oct 12, 2014	Iliana Simova	IICT-BAS	Bulgarian components (3.9).
0.6	Oct 12, 2014	Jaroslava Hlaváčová	CUNI	First revision sent for internal review.
0.7	Oct 14, 2014	Gertjan van Noord	UG	Alpino part of Dutch components (3.6).
0.8	Oct 22, 2014	Rosa Del Gaudio	HF	Feedback from HF.
0.9	Oct 30, 2014	António Branco	FCUL	Portuguese components (3.7).
1.0	Oct 31, 2014			Preliminary version submitted for M12 review.
1.1	Mar 10, 2015	Rudolf Rosa	CUNI	Restructuring the deliverable.
1.2	Mar 16, 2015	Ondřej Dušek	CUNI	Updating Czech and Dutch.
1.2	Mar 16, 2015	Eleftherios Avramidis, Aljoscha Burchardt, Arle Lommel, Maja Popovic	DFKI	Updating German and evaluation results.
1.2	Mar 17, 2015	Iliana Simova	IICT-BAS	Updating Bulgarian.
1.2	Mar 23, 2015	Nora Aranberri	UPV/EHU	Review and update of Spanish evaluation results and examples.
1.3	Mar 27, 2015	Luís Gomes and António Branco	FCUL	Updated Portuguese components (3.7).
1.4	Mar 30, 2015	Luís Gomes	FCUL	Added appendix A.
1.5	Mar 30, 2015	Arle Lommel	DFKI	Added results of analysis for Basque.
1.6	Mar 30, 2015	Kiril Simov and Petya Osenova	IICT-BAS	Comments from internal reviewers.
1.7	Mar 31, 2015	Nora Aranberri, Gorka Labaka	UPV/EHU	Update of Basque (3.4) and Spanish (3.8) sections. Review and update of Basque manual evaluations.
1.7	Mar 31, 2015	Jan Hajič, Jaroslava Hlaváčová, Martin Popel, Rudolf Rosa	CUNI	M12 review comments addressed. Integrated comments from the internal review. Checked for missing pieces.
1.8	Apr 1, 2015	Jan Hajič and António Branco	CUNI and FCUL	Final review before submission.

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 first MT pilot and its evaluation

DOCUMENT QTLEAP-2015-D2.4
EC FP7 PROJECT #610516

DELIVERABLE D2.4

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, Jaroslava Hlaváčová, Ondřej Bojar, Ondřej Dušek, António Branco, Luís Gomes, João Rodrigues, João Silva, Andreia Querido, Nuno Rendeiro, Marisa Campos, Diana Amaral, Eleftherios Avramidis, Aljoscha Burchardt, Maja Popovic, Arle Lommel, Iliana Simova, Nora Aranberri, Gorka Labaka, Gertjan van Noord, Rosa Del Gaudio, Michal Novák, Rudolf Rosa, Aleš Tamchyna, Jan Hajič

© all rights reserved by FCUL on behalf of QTLeap

Contents

List of Abbreviations

BDT	Basque Dependency Treebank
FGD	Functional Generative Description
HMM	Hidden Markov Model
HMTM	Hidden Markov Tree 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
USD	Universal Stanford Dependencies
WER	word error rate
WSD	word sense disambiguation
TM	translation model

1 Overview

Deliverable D2.4 aims at describing MT Pilot 1, which encompasses the entry-level deep MT systems for all language pairs in the project, including their empirical evaluation, as introduced in D2.3. The language resources and tools resorted to and enhanced to support these MT systems are referred to in the sections below and their description is provided in Deliverable D2.5.

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. The MT approaches we are using are hybrid, combining statistical and rule-based processing. In line with the objectives of the usage scenario (WP3), the focus of direction X→EN is aimed at supporting cross-lingual information retrieval.

All language pairs follow the same processing pipeline of analysis, transfer and synthesis (generation)[1] as depicted in Figure 1 and adopt the same methodology of using both statistical as well as rule-based components in a tightly integrated way for the best possible results.

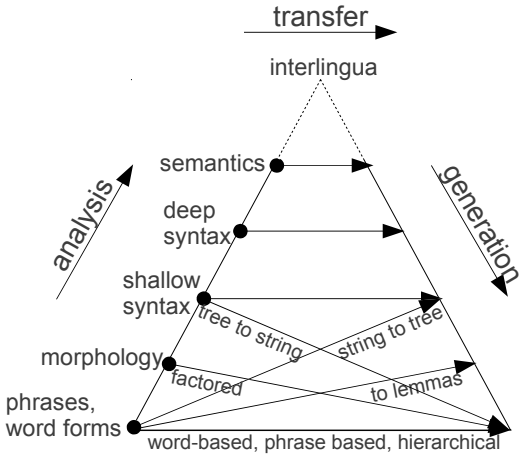


Figure 1: Vauquois diagram for transfer-based machine translation (i.e. consisting of three phases: analysis, transfer and generation). Most current approaches to MT (word-based, factored phrase-based, hierarchical, tree-to-string, string-to-tree) operate only on the morphological or shallow-syntax layer and are not based on transfer. QTLeap focuses on transfer via deep syntax and semantics.

This deliverable is structured as follows. Section 2 describes and explains the overall approach we took in implementing the entry-level deep MT systems. In Section 3, we describe in detail each of the particular systems developed for the individual language pairs. Finally, Section 4 is devoted to the evaluation of the Pilot 1 systems in terms of translation quality.

2 Introduction to Systems and their Architecture

Our base approach to the entry-level deep translation is the TectoMT system, described in Section 3.2. TectoMT is implemented in the Treex[2] NLP framework.

[1] Terms *synthesis* and *generation*, as used in this deliverable (and related literature), are synonyms.
[2] https://ufal.mff.cuni.cz/treex

A basic version of the TectoMT system for English-to-Czech translation had already been developed by CUNI, who is the main partner responsible for WP2, before the start of the QTLeap project. In WP2, we ported the TectoMT system to all language pairs for which there was no other pre-existing deep MT system, for both X-to-EN and EN-to-X translation. This concerns Basque, Czech,[3] Dutch, Portuguese, and Spanish.

A strong feature of TectoMT is the fact that it is language-universal in many aspects, making it easy to port to new languages. Furthermore, it is modular, enabling smooth incorporation of various pre-existing language-specific tools that had been developed by other partners. Thus, TectoMT was ported to each new language pair by starting with the language-universal parts only, then incorporating pre-existing language-specific tools, and finally implementing missing language-specific components and fine-tuning the system to the QTLeap setting.

Conveniently, for the remaining two languages, there already had been a basic deep MT system – for Bulgarian, this is the Deep factored MT system (Section 3.9), and for German, it is the quality system combination (Section 3.10) that combines a transfer-based MT system with a phrase-based SMT both in a linear combination and using a quality estimation module trained on labeled corpora that uses deep features to select the best output. Thus, instead of porting TectoMT to these languages and then developing their language-specific components basically from scratch, we took the decision to reuse the pre-existing systems, to further develop them and to adjust them to the QTLeap setting, as there was a clear potential of achieving a high quality translation for Pilot 1.

Still, preliminary work has already been done in porting TectoMT to Bulgarian and German as well – committed to achieve the highest possible quality of MT, we plan to investigate the possibilities of combining multiple approaches to deep MT in further pilot systems.

3 Pilot 1 Systems

3.1 TectoMT-based Systems in Pilot 1

The general TectoMT pipeline, described in Section 3.2, is language-universal, and consists of analysis, deep transfer, and synthesis steps. Further sections deal with the specifics of individual language pairs, summarizing differences in system training and operation for each language pair.

3.2 General structure of TectoMT-based Systems

TectoMT is a structural machine translation system with deep transfer, first introduced by Žabokrtský et al. [2008].

This system uses two layers of structural description, *a-layer* (shallow, see Section 3.2.1) and *t-layer* (deep, see Section 3.2.2). The analysis phase is two-step and proceeds from a-layer to t-layer (see Section 3.2.3).

The transfer phase of the system is based on Maximum Entropy context-sensitive translation models [Mareček et al., 2010] and Hidden Tree Markov Models [Žabokrtský and Popel, 2009]. It is factorized into three subtasks: t-lemma, formeme and grammatemes translation (see Section 3.2.4).

[3] The pre-existing English-to-Czech TectoMT was improved within QTLeap. Czech-to-English TectoMT was created anew.

The subsequent generation phase consists of rule-based components that gradually change the deep target language representation into a shallow one, which is then converted to text (cf. Section 3.2.5).

All of the aforementioned sections deal with the actual operation of a trained system. The training of the system is detailed in Section 3.2.6.

3.2.1 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.
- *morphological tag* – morphological description of the word form – all morphology information describing the actual word form used in the sentence.
- *afun* – surface dependency label. The labels largely correspond to commonly known syntactic functions such as subject, predicate, object, and attribute (`Sb`, `Pred`, `Obj`, `Atr`).

In order to facilitate language independent solutions, QTLeap decided to use *Interset morphological features* [Zeman, 2008] in addition to the *morphological tag*. Interset provides a way to map morphological (and some syntactic) features of all 8 languages involved in QTLeap into one common scheme. QTLeap developers also help to improve Interset definition and implementation as it plays a key role in the MT pilots.

3.2.2 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 as nodes (*t-nodes*).

Auxiliary words, such as prepositions, subordinating conjunctions or auxiliary verbs, 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". This is usually identical to the surface lemma, but some related surface lemmas are merged (personal pronouns, possessive adjectives derived from nouns) or modified (t-lemmas of reflexiva tantum verbs[4] include the reflexive

[4] Reflexiva tantum are reflexive verbs that do not have non-reflexive counterparts. Reflexiva tantum verbs are frequent in Slavic languages. See https://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/t-layer/html/ch04s03s01.html for other examples of multi-word t-lemmas in Czech.

particle; in phrasal verbs, they include the phrasal preposition/particle; t-lemmas of verbs with separable prefixes always contain the prefix, regardless of whether it is separated).

- *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. semantic part-of-speech, number for semantic nouns, grade for semantic adjectives and adverbs, or person, tense, and modality for semantic verbs.

- *formeme* – morpho-syntactic form information, composed of coarse-grained part-of-speech based on syntactic behavior (verb, noun, adjective, adverb), prepositions or subordinate conjunctions (for prepositional phrases and verbs in dependent clauses), and coarse-grained syntactic form (e.g., finite/infinitive/gerund for verbs or case/syntactic position for nouns). This adds up to a simple human-readable string, such as `v:to+inf` for infinitive verbs or `n:into+X` for a prepositional phrase. The result can easily be used both in hand-written rules and statistical systems [Žabokrtský, 2010].

  The set of formemes is language-dependent, but formemes are easy to assign using simple rules (which can be partially language-independent). The formeme attribute is not based on the FGD theory, but proved very useful for translation in practice. TectoMT uses a formeme-based transfer instead of functor transfer (see Section 3.2.4).

3.2.3 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 must be 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 [Popel et al., 2011], such as the Maximum Spanning Tree parser of McDonald et al. [2005a] or the Malt parser of Nivre et al. [2006].

The a-tree is then gradually transformed into a t-tree by modules that perform the following tasks:

1. *Removal of auxiliary words from the tree* – only content words, such as nouns, lexical verbs, or adjectives have their own nodes on the t-layer. Therefore, nodes of auxiliary words (such as prepositions, subordinate conjunctions, auxiliary verbs, punctuation) are "collapsed", i.e. removed from the tree while links to auxiliaries are retained in t-nodes to which they relate (e.g., prepositions are linked from nouns, auxiliary verbs from the lexical verb).

2. *Changing surface lemmas to t-lemmas* – while most lemmas remain identical, lemmas of personal pronouns are changed to `#PersPron` while reflexiva tantum verbs,

separable and phrasal verbs as well as multi-word surnames now include multiple words in their lemmas, e.g. `screw_up`.

3. *Formeme assignment* – formemes are assigned by a rule-based block that uses both t-tree and a-tree context. The rules are language-specific, but some parts are shared among languages. In general, the "syntactic part-of-speech" (syntactic behavior) of a node is determined, then based on this, important morphosyntactic features and auxiliary words are included in the formeme.

4. *Functor assignment* – semantic roles are now detected and marked for each node. The actual implementation of this step varies across languages (with supervised machine learning models for English and Czech and rules for other languages, where no t-layer gold-standard annotation is available).

5. *Grammateme assignment* – important linguistic properties of each node are stored, based on its part-of-speech. We use the following set of grammatemes:

   - *semantic part-of-speech*
   - *tense, aspect, diathesis, deontic, verbal, and sentence modality* for semantic verbs
   - *number, person*, and *gender* for semantic nouns
   - *degree of comparison* for semantic adjectives and adverbs
   - *negation* for semantic verbs, nouns, adverbs and adjectives

6. *Reconstructing actors (deep subjects)* – this concerns cases where the subject/actor personal pronoun is not present on the surface, such as:

   - *Pro-dropped subjects* – here, the subject is a personal pronoun whose person and number are usually indicated by the morphology of the main verb of the clause. This is used to reconstruct and insert a new node that bears a `#PersPron` t-lemma and grammatemes corresponding to the verbal morphology.
   - *Imperative* – imperative clauses typically do not express the actor explicitly; a 2nd person generic actor is assumed.
   - *Passive* – in passive and reflexive passive clauses, the actor may not be present on the surface; in that case, a generic 3rd person singular actor node is generated.

7. *Coreference* – coreference links are introduced to connect anaphors with their antecedents. For different languages, various types of coreference (e.g. with relative, reflexive or personal pronouns as anaphors) using different approaches are resolved.

The implementation details as well as the order and/or presence of all of these steps vary across languages, as detailed in the later language-specific sections.

3.2.4 Transfer: Translation Factorization

Using the t-layer representation in structural MT allows separating the problem of translating a sentence into three relatively independent simpler subtasks: the translation of t-lemmas, formemes and grammatemes [Bojar and Týnovský, 2009, Žabokrtský, 2010].

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. The t-layer as implemented in TectoMT makes such assumption possible, albeit still problematic [Popel, 2009, p. 64]. Nonetheless, it allows us to model each of these three subtasks by a symmetric source-target one-to-one mapping, thus simplifying the initial n-to-m mapping of word phrases or surface treelets.

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
2. Producing an n-best list of translation variants using formeme translation model
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 TM – prediction is based on features extracted from the source tree.
- Dictionary TM – this is only a dictionary of possible translations with relative frequencies (no contextual features are taken into account). This brings in the valuable information on the relative frequency of a translation variant. In source code, dictionary TM are called *static* TM.
- Other – backoff components that focus on out-of-vocabulary t-lemmas using handcrafted rules and various small "derivative" dictionaries. This component is language-dependent and only used in t-lemma TM.

The discriminative TM [Mareček et al., 2010] is in fact an ensemble of maximum entropy (MaxEnt) models [Berger et al., 1996], each trained for one specific source t-lemma/formeme. However, as the number of types observed in the parallel treebank may be too large, infrequent source t-lemmas/formemes are covered only by a Dictionary TM.

The 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), [Crouse et al., 1998, Žabokrtský and Popel, 2009]. HMTMs are similar to standard (chain) Hidden Markov Models but operate on trees.

In HMTMs, the *transition probability* says how likely a node v (with hidden state variable $X(v)$) occurs given its parent $\rho(v)$ (with hidden state variable $X(\rho(v))$). Emissions correspond to nodes (with hidden states) emitting the observed output. The *emission probability* then describes how likely a hidden state $X(v)$ is to emit the observed output $Y(v)$.

HMTMs are well suited for describing generation/translation from syntactic trees – in particular, dependency trees are captured quite naturally by HMTMs. In the context of deep MT transfer, observed variables correspond to nodes in the parsed source-language t-tree. The task is then to find the most probable assignment of hidden variables, which in turn correspond to target-language t-nodes. 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.

3.2.5 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 [Ptáček and Žabokrtský, 2006, Žabokrtský et al., 2008, Dušek et al., 2012].

The small tasks carried out by the modules in the pipeline are language specific but in general include solving the following problems (not necessarily in the order listed):

- *Word ordering* – word order imposed by the syntax of the target language is enforced.
- *Agreement* – morphological attributes are deduced based on agreement with properties of the context (such as in subject-predicate agreement or noun-attribute agreement).
- *Prepositions and conjunctions* – nodes are created for prepositions and subordinate conjunctions, which do not have separate nodes in tectogrammatical trees.
- *Compound verb forms* – additional nodes are added for verbal particles (infinitive, reflexive, or phrasal verbs) and for compound expressions of tense, mood, and modality.
- *Grammatical words* – negation particles, articles, and other grammatical words are added into the sentence.
- *Punctuation* – nodes for commas, final punctuation, quotes, and brackets are introduced.
- *Inflection* – inflected word forms are produced based on known morphological information from the context.
- *Phonetic changes* – word forms are changed according to phonetic combinatorial rules.
- *Capitalization* – words that start a sentence are capitalized.

The implementation is a mix of rule-based and statistical components. Most higher-level rules are very general and can be transferred to code in a straightforward way, which makes their implementation relatively simple. Grammar rules with lexical dependencies, such as inflection, can be solved using dictionaries or statistical modules such as "Flect" [Dušek and Jurčíček, 2013].

3.2.6 System Training

The statistical components use supervised learning and require training data. While the procedure for training most analysis components (such as morphological taggers or dependency parsers, see Section 3.2.3) is fairly standard and involves annotated corpora and treebanks, training translation models requires a more complex procedure using automatic annotation to obtain deep parallel treebanks.

Automatic annotation is used to train translation models due to two reasons:

1. It follows the real-life scenario where features are extracted from automatically annotated data. Training the components on manually annotated data would result in more errors in real usage, particularly for the components operating on the t-layer (which includes translation models).

2. Translation models require very large parallel treebanks including node-to-node alignment, which would be too expensive to obtain by manual work.

We obtain parallel deep treebanks by using automatic annotation (analysis) up to t-layer on both languages (based on analysis tools which use both rule-based components and statistical modules trained on manually annotated deep treebanks) and unsupervised word alignment. The annotation pipeline starts with a bitext aligned on a sentence level and ends with a parallel treebank containing pairs of t-trees aligned on the level of t-nodes.

The analysis phase of the pipeline mimics the one used in a translation process, including tokenization, lemmatization, morphological tagging, dependency parsing to a-layer and a conversion to t-trees. The analysis pipeline is run independently on each of the two languages, taking no advantage of the joint bilingual processing for the time being.

The word-alignment stage of the pipeline interlinks the constructed pair of t-trees in three steps – first, automatic word alignment is obtained on lemmatized bitexts (strings of word base forms) using the GIZA++ tool [Och and Ney, 2003]. Second, the word-alignment links obtained by GIZA++ are projected to the corresponding nodes in the t-trees. Third, additional heuristic rules are used to align t-nodes that have no counterparts on the surface (e.g., pro-dropped pronominal subjects).

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.

3.3 English Components

This section details English-specific features of the TectoMT pipeline, used for all language pairs.

3.3.1 Analysis

The English analysis follows the annotation pipeline used for the CzEng 1.0 parallel corpus [Bojar et al., 2012]. The Morče tagger [Spoustová et al., 2007] is currently used for morphological tagging; we plan to replace Morče with MorphoDiTa [Straková et al., 2014] in near future, as MorphoDiTa is much faster and easier to install.

Dependency parsing to a-trees is performed by the Maximum Spanning Tree parser [McDonald et al., 2005a] trained on the CoNLL-2007 conversion of Penn Treebank [Nilsson et al., 2007]. Since the set of dependency labels in the CoNLL-2007 data is different from the one used on the a-layer, the labels are converted by a rule-based block after the parsing.

The English t-layer conversion starts from the a-tree and follows the process outlined in Section 3.2.3 closely; in the following, we detail the parts specific for English:

- The t-lemma assignment focuses on phrasal verbs, personal pronouns, and the negation particle "no", which obtains the t-lemma `#Neg`.
- Formeme assignment reuses language-independent code with the help of the Interset morphology abstraction layer [Zeman, 2008]. Formemes for English nouns include their syntactic position (subject, direct object, indirect object, attribute, possessive).
- Functors are assigned in a rule-based fashion, based on auxiliary words, lemmas, and formemes. A more advanced functor assignment module based on logistic regression [Fan et al., 2008] and trained on the Prague Czech-English Dependency Treebank (PCEDT) 2.0 data [Hajič et al., 2012] is available, but is currently not used in the translation pipeline since it only provides modest benefits and comes at a much greater computational cost.
- Grammateme assignment for English uses language-specific rules which are mainly based on surface morphology and presence of auxiliaries.
- Generated actor (deep subject) nodes are added in imperative clauses and in control constructions (with a verb that governs an infinitive clause). In the latter case, the added deep subjects of infinitive clauses have a coreference link to the subject or the object of the governing verb, based on the type of the control construction.
- Simple modules for coreference resolution in English are available. They aim at coreference of relative and possessive pronouns.

The English analysis includes further modules whose purpose is purely technical and which are omitted from the description for the sake of brevity.

A significant issue we encountered when applying the analysis tools to the QTLeap domain was their unpreparedness for imperative sentences. This is mainly due to the fact that the tools are typically trained on news domain corpora, which contain very few imperative sentences. In Pilot 1, we use a set of heuristic rules that try to detect imperative constructions based on various cues, especially the verb form and the position of the subject as identified by the tools. We will investigate possibilities of directly adapting the analysis tools in future pilots.

3.3.2 Synthesis

The English synthesis pipeline also adheres to the general setup presented in Section 3.2.5 and tries to reuse language-independent code where possible. The parts that were specifically designed for English involve, ordered as they appear in the generation pipeline:

- a rule-based word ordering module that enforces the SVO order in indicative clauses, as well as the vSVO order in interrogative clauses,
- detection of surface morphology based on grammatemes, including the enforcement of subject-predicate agreement,[5]

[5]This is only language-specific due to technical reasons (the implementation predated switching to Interset for morphology).

- modules that insert English-specific auxiliary words – infinitive and phrasal verb particles, possessive markers (*'s*), and articles,
- insertion of auxiliary verbs based on grammatemes,
- heuristic rules for the necessary English punctuation (clause-initial punctuation and punctuation for some dependent clauses),
- word form generation, which uses MorphoDiTa[6] dictionary-based generator and the Flect statistical generator [Dušek and Jurčíček, 2013] trained on English PCEDT 2.0 data,
- rules for indefinite article phonetics (distinguishing *a* and *an*).

All blocks, with the exception of word form generation, are rule-based. We plan to switch to statistical article assignment as the performance of the current module (based on an older English generator by Ptáček [2008]) is suboptimal.

3.4 Basque: TectoMT

3.4.1 Analysis

The linguistic tools that were integrated into Pilot 1 (tokenization, POS tagging, lemmatization and dependency parsing) had been already developed and available, as well as a number of modules that might be integrated in later pilots (NERC, NED, WSD, Coreference and SRL). The approach we took in Pilot 1 was to develop wrappers to invoke our tools directly from Treex. For future pilots, we will investigate whether these tools perform well in TectoMT, or whether it would be better to consider training models for the tools developed at CUNI instead (MorphoDiTa, for instance).

Eustagger [Alegria et al., 2002] is a robust and wide-coverage morphological analyzer and a Part-of-Speech tagger for Basque. The analyzer is based on the two-level formalism and has been designed in an incremental way with three main modules: (1) the standard analyzer, (2) the analyzer of linguistic variants, and (3) the analyzer without lexicon which can recognize word-forms without having their lemmas in the lexicon.

Eustagger provides the lemma, POS and morphological information for each token. In the tagger, combination of stochastic (HMM models) and rule-based disambiguation (Constraint Grammar) methods are applied to Basque.

The dependency parser is based on the MATE-tools [Björkelund et al., 2010], a pipeline of linguistic processors that performs lemmatization, part-of-speech tagging, dependency parsing, and semantic role labeling of a sentence. As the input of the module is already tagged with lemmatization and pos-tagging, the module only implements the dependency parsing [Bohnet, 2010]. Basque models have been trained using the Basque Dependency Treebank (BDT) corpus [Aduriz et al., 2003]. BDT treebank is already included in HamleDT,[7] so we use the harmonization rules already developed by CUNI to convert our analyzes into harmonized parses (i.e. the a-layer style used in HamleDT).

Transformation from the a-level analysis into t-level is partially performed with language-independent blocks thanks to the support of Interset [Zeman, 2008]. This includes the

[6] http://ufal.mff.cuni.cz/morphodita [Straková et al., 2014]
[7] http://ufal.mff.cuni.cz/hamledt

initial construction of t-trees, handling relative clause and reflexive pronoun coreference, and (partially) assigning grammatemes.

Additional language-specific modules that have been created for Basque analysis involve:

- selection of meaning-bearing elements to be included as nodes in the t-tree,
- functor detection (rule-based, based on t-lemmas, formemes, and auxiliary words),
- verbal grammateme detection based on morphology as well as auxiliary and modal verbs.

These language-specific modules reuse language-independent code where appropriate.

3.4.2 Transfer

Both English-to-Basque and Basque-to-English transfer uses discriminative and dictionary translation models as described in Section 3.2.4.

3.4.3 Synthesis

The Basque synthesis pipeline follows the general scenario and reuses many language-independent modules, including morphology initialization and morphological agreements (subject-predicate and attribute-noun) and insertion of punctuation. Language-specific synthesis modules for Basque include:

- pro-drop subject and object pronouns are deleted from the surface,
- insertion of auxiliary verbs based on grammatemes,
- basic word ordering.

Basque is an agglutinative language, therefore, the generation of the inflected word forms is a considerably hard task – even harder than for other highly inflected languages, such as Spanish. In order to guide this process, we trained models for Flect, the morphologic generation tool developed by CUNI [Dušek and Jurčíček, 2013]. Flect uses a morphologically annotated corpus to automatically learn how to inflect word forms based on lemmas and morphological features. The system is able to inflect previously unseen words based on *edit scripts*.

3.5 Czech: TectoMT

3.5.1 Analysis

The Czech analysis is similar the annotation pipeline of the CzEng 1.0 corpus [Bojar et al., 2012]. Original CzEng annotation used Morče tagger [Spoustová et al., 2007]. In QTLeap, we have replaced Morče with MorphoDiTa tagger [Straková et al., 2014], which is better both in terms of accuracy (for Czech) and speed. Parsing to a-layer is done with a version of the non-projective Maximum Spanning Tree parser adapted for Czech [McDonald et al., 2005b, Novák and Žabokrtský, 2007] and trained on the Prague Dependency Treebank (PDT).

The conversion to t-layer follows the process described in Section 3.2.3. Language-specific phenomena include:

- In addition to handling reflexiva tantum verbs and personal pronouns, t-lemma assignment includes the usage of noun t-lemmas for possessive adjectives – e.g. *matčin (mother's)* is converted to *matka (mother)*.
- Special rules handle the inconsistent surface-syntactic behavior of Czech numerals, where some ordinal numbers behave like nouns and some like adjectives.
- Czech formemes are set by entirely language-specific rules. They capture finiteness and subordinating conjunctions for verbs, preposition and morphological case for nouns, and attributive/possessive function for adjectives.
- Functor assignment is performed by a logistic regression classifier [Fan et al., 2008] trained on PDT data and achieves 78% accuracy.
- Grammatemes are assigned by an extensive language-specific set of rules.
- Generated actor (deep subject) nodes are added for pro-dropped subject pronouns as well as deep subjects of passive and reflexive passive clauses which are missing on the surface.
- Coreference resolution of relative and reflexive pronouns is carried out by simple rules. Personal pronouns in 3rd person are targeted by a system introduced in [Bojar et al., 2012].

To achieve higher accuracy of the analysis tools on the QTLeap domain, we introduced several approaches to domain adaptation. As the data used for training the tools are mainly news text, we observed a significant decrease in their performance when applied to the QTLeap domain, for which there is a lack of annotated corpora.

To correctly identify the base form of domain-specific unknown words, we use data mined from Wikipedia, where words that appear in the text of the article and are similar to the title of the article are regarded as its inflections, and the title is regarded as the base form.

The tools by themselves are also generally unable to identify text sequences that are not to be analyzed and translated, but should be kept intact throughout the whole process, such as URLs, e-mail addresses, file paths, or computer commands. A simple but relatively powerful approach we took is to "hide" such elements in the input by replacing them with a special unique token that passes unchanged through the whole pipeline, keeping the information about the original contents separately, and only replacing the token with the original contents at the end of the synthesis phase.

3.5.2 Transfer

The English-to-Czech transfer uses the combination of translation models and tree model reranking described in Section 3.2.4. In addition, simple rule-based blocks handle changes to t-tree topology (insertion and deletion of t-layer nodes) and grammateme changes, such as the addition of grammatical gender. Also, specialized translation models for pronoun *it* and reflexive pronouns [Novák et al., 2013a,b] are employed. The models were trained on manually labeled examples of these English pronouns, since unsupervised alignment performs poorly for them.

The Czech-to-English direction is very similar, except that no tree model reranking is done. Rule-based blocks include removal of grammatical gender, other grammatical changes, and small changes to t-tree topology.

3.5.3 Synthesis

Czech synthesis blocks come from the original TectoMT system [Žabokrtský et al., 2008], which in turn is based on Ptáček and Žabokrtský [2006]'s Czech t-layer generator and roughly adheres to the general pipeline described in Section 3.2.5.

The Czech setup in particular is composed of the following steps:

1. Inconsistent syntactic behavior of numerals in Czech is accounted for.
2. Surface morphology information is filled based on semantic properties of the tectogrammatical tree nodes.
3. This information is enriched based on various agreement rules of Czech:
   - Subject-predicate agreement
   - Agreement of relative pronouns with their antecedents
   - Agreement of adjectives with the governing nouns
   - Agreement of a predicative complement with the subject
4. Pro-drop subject pronouns are deleted from the surface.
5. Grammatical words, such as prepositions, conjunctions, and particles are introduced.
6. All Czech compound verb forms are generated.
7. Clause-level and other punctuation is included.
8. Inflected word forms are generated based on information available so far.
9. Clitics are moved to the Wackernagel position (second member of the clause).
10. Capitalization and detokenization are performed.

All of the above-mentioned tasks are implemented as rules based on linguistic expertise, except for word form generation (8), which uses a large-coverage morphological dictionary [Hajič, 2004] and lemma-tag-form frequency information from corpora.

3.6 Dutch: TectoMT

3.6.1 Analysis

The analysis of Dutch input uses the Alpino system. Alpino is an implementation of a stochastic attribute value grammar [van Noord, 2006, de Kok et al., 2011]. The grammar contains over 800 grammatical rules, expressed in the attribute value grammar notation. Disambiguation is performed by a statistical model (a maximum entropy model) trained on a gold standard treebank containing about 10 thousand sentences. A very large lexicon (over 300 thousand entries), along with a very large set of heuristics to recognize named entities as well as unknown words and word sequences, provides attribute value structures for the words in the input. A tightly integrated part-of-speech tagger, implemented as a trigram Hidden Markov Model, uses the forward-backward algorithm to compute the posterior probability of each potential lexical category, and removes the most unlikely ones

for improved efficiency [Prins and van Noord, 2003]. Further efficiency gains are obtained using a technique of statistical guides, described in van Noord [2009]. The parser has been carefully tuned to real world input, using error mining techniques [de Kok et al., 2009]. The current accuracy of the parser is over 90% (labeled dependency) for newspaper text, and experiments have shown that the parser is more robust against changes in domain and text genre in comparison with popular statistical dependency parsers [Plank and van Noord, 2010].

Alpino is available under an open source license, and its sources as well as binary versions of the system can be downloaded from `http://www.let.rug.nl/vannoord/alp/Alpino/`.

The Alpino-parsed dependency trees first undergo a simple rule-based conversion that makes them more similar to dependency trees (a-trees) used in Treex (and TectoMT). This involves decoding parts-of-speech using a driver for Interset [Zeman, 2008] which we created for the Alpino part-of-speech tagset.

The rest of the t-tree conversion pipeline parallels other languages. In some cases, language-independent blocks can be used thanks to Interset support. This includes the initial construction of t-trees, handling relative clause and reflexive pronoun coreference, and (partially) assigning grammatemes.

Additional language-specific modules that have been created for Dutch analysis involve:

- selection of meaning-bearing elements to be included as nodes in the t-tree,
- functor detection (rule-based, based on t-lemmas, formemes, and auxiliary words),
- t-lemma handling of negation, personal pronouns, and reflexiva tantum particles,
- handling of multi-word surnames in Dutch,
- formeme assignment,
- verbal grammateme detection based on morphology as well as auxiliary and modal verbs.

These language-specific modules reuse language-independent code where appropriate.

3.6.2 Transfer

Both English-to-Dutch and Dutch-to-English transfer uses discriminative and dictionary translation models as described in Section 3.2.4. Rule-based modules in the English-to-Dutch direction handle changes in t-tree topology and Dutch grammatical gender.

3.6.3 Synthesis

The Dutch synthesis pipeline adheres to the general scenario and reuses many language-independent modules, including morphology initialization and agreements (subject-predicate and attribute-noun), insertion of prepositions and conjunctions based on formemes, and insertion of punctuation.

Language-specific synthesis modules for Dutch include:

- insertion of infinitive particles (*om–te*) and reflexive particles (*zich*),

- insertion of articles,
- insertion of auxiliary verbs based on grammatemes,
- basic word ordering (moving predicates to clause end).

The final step of the Dutch synthesis, the generation of the actual sentence including inflected word forms, is handled by the Alpino generator [de Kok and van Noord, 2010]. The a-tree resulting from the previous steps is first converted to an Alpino Abstract Dependency Tree, which is then used as the input for Alpino generation. The Alpino generator tasks include detailed word ordering, separable verb prefixes, inflection, and punctuation. The Alpino generator is based on the Alpino grammar and lexicon which is also used in parsing. In addition, a statistical fluency model based on maximum entropy selects the most fluent realization out of the set of realizations that are compatible with the input.

In order that the Alpino genearor be used with TectoMT, the generator has been made more robust to handle noisy input from incorrect parses and/or translation model decisions. This includes a general back-off strategy to apply the generation algorithm to the parts of the input structure in case no full generation is possible, as well as a number of heuristics which rearrange the input Alpino Abstract Dependency Tree to match the expectations implicit in the Alpino grammar and lexicon.

3.7 Portuguese: TectoMT

The language pairs Portuguese–English and English–Portuguese in MT Pilot 1 are developed along the common approach assumed by the language pairs in the project, viz. a transfer-based approach. Their development relies on the TectoMT system, and progresses through the implementation of the components that integrate the different phases of this approach.[8]

3.7.1 Analysis

For the analysis phase, the project resorts to the reference pipeline of LX processing tools for Portuguese, which cover from raw text to graphs of grammatical dependencies. This pipeline includes a sequence of steps undertaken by a number of tools focused on the typical self-contained tasks in natural language processing, namely tokenization, lemmatization, morphological analysis, POS tagging, recognition of multi-word expressions from closed classes, and dependency parsing.

These tools are being adapted for their integration in our application, namely LX-Suite (tokenization, lemmatization, morphological analysis, POS tagging), LX-DepParser (dependency parsing). They offer state-of-the-art performance and have the advantage of covering complementary portions of the analysis step up to the computation of grammatical dependencies.

LX-Suite [Branco and Silva, 2006b] is composed by the set of shallow processing tools briefly described in the following lines of text:

LX-Tokenizer: Besides the separation of words, this tool expands contractions.: do → |de_|o|. It detaches clitic pronouns from the verb and the detached pronoun is marked

[8] The evaluation scores reported for Pilot 1 by the EN-PT system below in Section 4 are based on `tectomt` revision 14 386.

with a "-" (hyphen) symbol (dá-se-lho → |dá|-se|-lhe|-o|, afirmar-se-ia → |afirmar-CL-ia|-se|, vê-las → |vê#|-las|). This tool also handles ambiguous strings. These are words that, depending on their particular occurrence, can be tokenized in different ways. For instance: deste → |deste| when occurring as a verb, deste → |de|este| when occurring as a contraction (Preposition + Demonstrative). This tool achieves an f-score of 99.72% [Branco and Silva, 2003].

LX-Tagger: This tool assigns a single morpho-syntactic tag to every token: um exemplo → um/IA exemplo/CN. Each individual token in multi-token expressions gets the tag of that expression prefixed by "L" and followed by the number of its position within the expression: de maneira a que → de/LCJ1 maneira/LCJ2 a/LCJ3 que/LCJ4. This tagger was developed over Hidden Markov Models technology and an accuracy of 96.87% was obtained [Branco and Silva, 2004].

LX-Featurizer (nominal): Assigns inflection feature values to words from the nominal categories, namely Gender (masculine or feminine), Number (singular or plural) and, when applicable, Person (1st, 2nd and 3rd): os/DA gatos/CN → os/DA#mp gatos/CN#mp. It also assigns degree feature values (diminutive, superlative and comparative) to words from the nominal categories: os/DA gatinhos/CN → os/DA#mp gatinhos/CN#mp-dim. This tool has 91.07% f-score [Branco and Silva, 2006a].

LX-Lemmatizer (nominal): Assigns a lemma to words from the nominal categories (Adjectives, Common Nouns and Past Participles): gatas/CN#fp → gatas/GATO/CN#fp, normalíssimo/ADJ#ms-sup → normalíssimo/NORMAL/ADJ#ms-sup. This tool has 97.67% f-score [Branco and Silva, 2007].

LX-Lemmatizer and Featurizer (verbal): Assigns a lemma and inflection feature values to verbs. escrevi/V → escrevi/ESCREVER/V#ppi-1s This tool disambiguates among the various lemma-inflection pairs that can be assigned to a verb form, achieving 95.96% accuracy [Martins, 2008, Nunes, 2007, Branco and Henriques, 2003].

LX-DepParser: This is a MaltParser [Nivre et al., 2007] trained with the CINTIL-DependencyBank [Branco et al., 2011]. For the training of the parser, 14,052 sentences were used (comprising 161,772 word tokens). Its labeled attachment score (LAS) is 91.21.

In benefit of the adoption of de facto standards and its harmonization within the project, this pipeline is completed with two further analysis steps, namely two converters between grammatical dependencies graphs complying with different linguistic formats. One of these tools converts from the linguistic CINTIL format into the Universal Stanford Dependencies (USD) format. The other tool closes the analysis step by converting from the USD format to the tectogrammatical layer (*t-layer*) described in Section 3.2.2.

CINTIL format follows a strict discipline of encoding only information on grammatical dependencies in the arcs of the graph representing grammatical dependencies, and thus separating this type of information from any linguistic information of other sorts, including POS tagging (which are encoded attached to the nodes of the graph).

USD in turn can be seen as a less-principled linguistic format and mixes, in the same representation level (arcs of labeled dependencies graphs), the encoding of different sorts of linguistic information, including information on grammatical dependencies, but also information on PoS and constituency, and on linguistic phenomena typology.

The tecto linguistic format conforms to the design options already described in Section 3.2.2.

The CINTIL-USD converter is a rule-based tool that handles the mapping from the CINTIL to the USD formats. The subsequent converter USD-Tecto is also rule-based and takes the processing flow from the USD to the Tecto linguistic formats. Given the similar

expressive power of the different linguistic formats involved, modulo eventual residual slips of implementation, there is no information loss in the conversion process. These conversions are undertaken in order to be possible to benefit from, and build on tools, development frameworks and datasets that are available for some of the formats but not for some of the others.

Modal verbs are not represented as *t-nodes* in the tectogrammatical layer, but instead they are encoded as three *grammateme* components[9] of their governing verbal *t-nodes*, namely, verbal modality, deontic modality and verb tense.

3.7.2 Transfer

The transfer step is responsible for mapping tectogrammatical attributes of source-language `t-nodes` into adequate attributes of the other language. Unlike *t-lemmas*, which are determined by the language, the domain of *formemes* is largely determined by the needs of the synthesis step. Consequently, the Portuguese t-layer representation, in particular the formemes, had to be adjusted in an iterative manner as Pilot 1 progressed, to accommodate all necessary information for word reordering and insertion of functional words in the synthesis step. For example, initially all attributive adjectives would have formeme `adj:attr`, but later, when developing reordering rules it became clear that a stochastic approach is more appropriate than a rule-based one since adjective-noun order is a lexical property of each adjective. Consequently, to support stochastic transfer of noun-adjective ordering, attribute adjectives were thereafter labeled with one of these two different formemes, `adj:prenom` and `adj:postnom`, according to their position with respect to the governing noun (prenominal or postnominal respectively).

Portuguese verbs with null-subject require insertion of a personal pronoun (`#PersPron`) t-node, to warrant a personal pronoun in the synthesized English sentence, which was provided.

URLs and other non-linguistic constructs such as menu paths (see example below) are very common in the QTLeap corpus, and they require special (non-linguistically motivated) treatment, which for Pilot 1 have not yet been treated to full extent. Example sentence containing a menu path, emphasized, which requires special treatment:

> In windows you should click *Start >Control Panel >Programs and Functionalities >Add or Remove Programs* and uninstall the respective application.

The eight models used in the transfer phase (2x2x2: *discriminative* and *dictionary* models for *t-lemma* and *formeme* transfer for both translation directions) were learned from the Europarl parallel corpus [Koehn, 2005]. To train the transfer models, both sides of the corpus (English and Portuguese) were analyzed up to the tectogrammatical layer, using the aforementioned analysis pipelines for English (Section 3.3) and Portuguese (described above).

Whenever the domain of attributes changes or the analysis of either language is changed, corpora has to be re-analyzed and the transfer models re-learned. This is a computationally demanding task but easily parallelized by sub-dividing the corpora into smaller work-units which are analyzed independently. QTLeap Manager (*qtlm* for short) was created to make iterative development and re-training more agile. The user manual for this tool can be found in Appendix A below.

[9] Unlike *t-lemma* and *formeme* attributes, which are unidimensional, *grammatemes* are multidimensional attributes

3.7.3 Synthesis

For the insertion of functional words and to ensure proper word order, rule-based components were created:

Negation is encoded in the *t-layer* as a grammateme of verbal *t-nodes* and it is synthesized by inserting "não" before the verb, which basically covers the needs for the QTLeap domain.

Like negation, modal verbs are not represented as *t-nodes* but instead they are encoded in grammatemes of verbal *t-nodes* as described earlier in Section 3.7.1. Portuguese modal verbs are synthesized by retrieving the appropriate verb from a hand-created table indexed by verbal modality, deontic modality and verb tense.

For the generation of surface forms of expressions appropriately inflected, the project resorts to the LX-Inflectorfor nominal expressions, and to the LX-Conjugator [Costa, 2004] for verbal expressions.

LX-Inflector: Takes a Portuguese (nominal) lemma, gender, number and degree and delivers a properly inflected noun or adjective. LX-Inflector processes simple forms as well as compounds (e.g. "trabalhador-estudante"), both lexically known and unknown ones (such as neologisms). It also handles nominal forms with prefixes (e.g. "anti-constitucional").

LX-Conjugator: Takes a Portuguese infinitive verb form, mood, tense, person and number and delivers the corresponding conjugated form. Verbal inflection is a complex part of the Portuguese morphology, given the high number of conjugated forms for each verb (ca. 70 forms in non pronominal conjugation), the number of productive inflection rules involved and the number of non regular forms and exceptions to such rules. LX-Conjugator handles both known verbs and unknown verbs (such as neologisms).

3.8 Spanish: TectoMT

3.8.1 Analysis

The analysis of Spanish input uses IXA pipes tools http://ixa2.si.ehu.es/ixa-pipes/. So far, we have used Treex tokenization, IXA pipes modules for POS tagging and lemmatization, and Mate tools for dependency parsing. The PoS tagging module implements different algorithms, but we have obtained the best results so far with Perceptron models and the same feature set as in [Collins, 2002]. Lemmatization is currently performed via 2 different dictionary lookup methods:

1. Simple Lemmatizer: It is based on HashMap lookups on a plain text dictionary. Currently we use dictionaries from the LanguageTool project (http://languagetool.org/) under their distribution licenses;

2. Morfologik-stemming (https://github.com/morfologik/morfologik-stemming): The Morfologik library provides routines to produce binary dictionaries, from dictionaries such as the one used by the Simple Lemmatizer above, as finite state automata. This method is convenient whenever lookups on very large dictionaries are required because it reduces the memory foot-print to 10% of the memory required for the equivalent plain text dictionary.

The dependency parsing module is based on the MATE-tools [Björkelund et al., 2010], a pipeline of linguistic processors that performs lemmatization, part-of-speech tagging, dependency parsing, and semantic role labeling of a sentence. As the input of the module

already includes lemmatization and pos-tagging, the module only implements the dependency parser [Bohnet, 2010]. Spanish models have been trained using the Ancora corpus (http://clic.ub.edu/corpus/en/ancora). Finally, the dependency trees obtained are harmonized to follow the HamleDT annotation guidelines. This process involves the modification of the tagset and some tree structures.

The t-tree conversion pipeline parallels other languages. In some cases, language-independent blocks can be used thanks to the support of Interset. This includes the initial construction of t-trees, handling relative clause and reflexive pronoun coreference, and (partially) assigning grammatemes.

Additional language-specific modules that have been created for Spanish analysis involve:

- selection of meaning-bearing elements to be included as nodes in the t-tree,
- functor detection (rule-based, based on t-lemmas, formemes, and auxiliary words),
- verbal grammateme detection based on morphology as well as auxiliary and modal verbs.

These language-specific modules reuse language-independent code where appropriate.

3.8.2 Transfer

Both English-to-Spanish and Spanish-to-English transfer use discriminative and dictionary translation models as described in Section 3.2.4.

3.8.3 Synthesis

The Spanish synthesis pipeline adheres to the general scenario and reuses many language-independent modules, including morphology initialization and agreements (subject-predicate and attribute-noun), insertion of prepositions and conjunctions based on formemes, and insertion of punctuation.

Language-specific synthesis modules for Spanish include:

- insertion of articles and prepositions,
- insertion of comparatives,
- insertion of auxiliary verbs based on grammatemes,
- basic word ordering.

At the moment, the morphological generation/synthesis module is very simple so we plan to train generation models based on Flect [Dušek and Jurčíček, 2013].

3.9 Bulgarian: Deep Factored MT

In this section we present the entry point deep MT for Bulgarian to English and English to Bulgarian based on Minimal Recursion Semantics (MRS). Minimal Recursion Semantics [Copestake et al., 2005] is a formalism for underspecified semantic representation specially developed for the aims of Machine Translation – see also Deliverable D4.1. The main assumption of using MRS is that the underspecified phenomena (quantification scope, for example) are not so important for Machine Translation.

3.9.1 MRS for Machine Translation

There are two main approaches in using MRS for supporting MT. The first is implemented within the Norwegian LOGON project and further developed with the DELPH-IN initiative via transfer rules [Bond et al., 2011]. The transfer rules in this framework are rewriting rules over MRS structures. The basic format of the transfer rules is:

$$[C :]I[!F] \rightarrow O$$

where I is the *input* of the rule, O is the *output*, C determines the *context* and F is the *filter* of the rule. C selects positive context and F selects negative context for the application of a rule – for more details see [Oepen, 2008].

The second approach uses MRS-based factors for a factor-based MT module as implemented in the EuroMatrixPlus Project [Wang et al., 2012a,b]. In this previous work we have investigated only Bulgarian to English machine translation using factors for Bulgarian text.

The first approach is much more accurate because it uses specially designed rules. But it requires much richer set of language resources than the currently available for Bulgarian. Thus, for the entry deep MT for Pilot1 we have chosen to reuse and extend the factors generated within EuroMatrixPlus project.

In the QTLeap project we have used a version of MRS – Robust Minimal Recursive Semantics (RMRS) – which is designed to be created from non-deep syntactic grammar like dependency grammars. In deliverable D4.1 we have presented the type of rules used to implement a module to extract RMRS structures from Bulgarian Dependency parses. These rules connect each word within a source sentence with elements of RMRS.

Currently we have implemented a module which determines the elementary predicate of the corresponding word, its main variable, and the main variables of the argument words. We have considered them as deep factors. In addition, we have used as factors information from the dependency parses themselves – POS tags, grammatical features, lemma, dependency labels and arcs. We have defined similar factors for English. English RMRS factors are built using the analysis produced by *ixa-pipes* [Agerri et al., 2014].

3.9.2 Pilot 1 System

For our entry level deep machine translation system, we make use of the Moses open source toolkit [Koehn et al., 2007] to build a factored SMT model [Koehn and Hoang, 2007]. Factored translation models are an extension of the standard phrase-based models which allow for additional linguistic information to be integrated into the translation process. Translation is based on vectors of linguistic factors, such as surface word form, lemma, part-of-speech tag, morphological specifications, etc. Thus, different properties of the source and target language can easily be represented and used for more linguistically informed translation.

We build upon previous related work for Bulgarian [Wang et al., 2012a,b], which we summarize in the following sections. Factored translation shares similarities with the TectoMT architecture, and we thus describe it in terms of analysis (Section 3.9.3), transfer (Section 3.9.4), and synthesis (Section 3.9.5) steps. The most significant improvements of the system that were made for Pilot 1 are described in the last section (3.9.6).

3.9.3 Analysis

In the analysis step, we create a representation of the text which encodes various levels of linguistic information as factors. These include morphological, syntactic and semantic abstractions in the source.

The linguistic analysis of the data starts with a preprocessing step. For Bulgarian this includes the application of a pipeline with modules for sentence splitting, tokenization, lemmatization, part of speech tagging, and dependency parsing. Several of these tools employing statistical, as well as rule-based techniques. Linguistic knowledge in the form of lexicons and gazetteers is used to improve the performance of purely statistical modules.

The Bulgarian pipeline's tagging module assigns tags from a rich tagset, which encodes detailed information about the morphosyntactic properties of each word [Simov et al., 2004]. The task of choosing the correct tag is carried out by GTagger, which is a guided learning system described in [Georgiev et al., 2012], and by a rule-based module which utilizes a large morphological lexicon and disambiguation rules [Simov and Osenova, 2001]. Lemmatization is also based on rules, generated using this morphological lexicon. These preprocessing steps are necessary to generate the input for the next step, viz. dependency parsing.

The dependency analysis is performed by MATE-tools parser [Bohnet, 2010]. This choice is motivated by previous experiments, in which the performance of several state-of-the-art dependency parsers was compared in a 10-fold cross validation experiment using Bulgarian data. MATE-tools parser achieved the best results with 92.9% UAS [Simov et al., 2014]. For training the parsing model we use the default settings of MATE-tools parser and the dependency conversion of the HPSG-based Treebank of Bulgarian – the BulTreeBank.[10]

For processing the English data we used the *ixa-pipes* system by selecting corresponding modules for tokenization, lemmatization, part of speech tagging and dependency parsing.

3.9.4 Transfer

The next translation steps involve the mapping between the linguistic layers represented as factors in the source and target language, followed by the generation of the target sentence based on this mapping. Transfer is carried out by translation models for the different factors, similarly to the translation of t-lemmas and formemes in TectoMT.

We have experimented with various combinations of factors derived from the preprocessing with the Bulgarian and English analysis pipelines, together with semantic factors based on the Minimal Recursion Semantics analysis. MRS structures are created on the basis of the morphosyntactic annotation and the dependency parses as it was done within the EuroMatrixPlus project [Wang et al., 2012a,b].

The following are some examples of factors for this model: word form, lemma, and morphosyntactic tags, factors modeling the parent word (lemma of the parent word, part of speech of the parent word) as well as the type of dependency relation (syntactic factors), and MRS-based factors (elementary predicate and variable type).

[10] www.bultreebank.org/dpbtb/

3.9.5 Synthesis

In the default setting, the Moses system performs transfer and synthesis jointly – the transfer component (translation model) provides multiple phrase translation options, and the synthesis component (language model) joins a subset of the phrase translations into an output sentence. Optionally, target language word forms can be generated based on the translations obtained during transfer. Multiple translation and generation step combinations can be defined by specifying decoding paths.

3.9.6 Improvements of the Pre-existing System

We extended the system of Wang et al. [2012a,b] mainly in two directions – better analysis with an improved pipeline for Bulgarian and for English, and different more complex types of factored models to explore successful factor combinations.

We have experimented with a number of combinations of the listed factors, language model types (word and POS), translation and generation steps. The best performing model for the direction from Bulgarian to English that we have found so far includes four factors – word form, lemma, POS and variable type – and a word and POS language model.

In the transfer step, two alternative approaches are used. If possible, we perform a mapping of the source word form and variable type to target word form candidates and POS candidates. However, if the source word form has not been seen during training, we take an alternative approach of using the source lemma together with the variable type instead.

For the translation direction from English to Bulgarian, the model includes three factors: word form, part of speech, and variable type. In the translation step, the source word, POS, and variable type are translated into the target word form.

For future pilots, we are modifying our data and pipeline to make it compatible with the TectoMT system, as we plan to use it in future pilot systems in addition to the factored translation model. For the same reason, we are currently constructing a mapping from the dependency version of the BulTreeBank to deep syntactic analysis.

3.10 German: Quality system combination

3.10.1 System Combination for German – Introduction

The German version of the deep MT system aims to effectively incorporate deep linguistic processing into existing successful machine translation methods for this language pair. Previous research in the field has shown great progress in developing systems that overcome obvious barriers of shallow natural language processing and pure empirical methods. Meanwhile, the fact that German has been 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 [Bojar et al., 2014]. Since our main goal is to achieve a high quality system that competes with the state-of-the-art and can be useful in the real use-case scenario (WP3), we use a system implementation that takes advantage of deep transfer and also includes a statistical mechanism that enhances performance by keeping the best parts from each employed method.

3.10.2 Analysis, transfer and generation

The core of the system for Pilot 1 is based on a transfer-based approach implemented in the Lucy system [Alonso and Thurmair, 2003] that includes the results of long linguistic efforts over the last decades and that has successfully been used in previous projects including Euromatrix+ and QTLaunchPad. The transfer-based approach has shown good results that compete with pure statistical systems, whereas its focus on translating according to linguistic structures sets the basis for the aims of this WP. In accordance with the overall MT paradigm of the project, translation occurs in three phases, namely *analysis*, *transfer*, and *generation*. All three phases consist of hand-written linguistic rules which have shown to perform well for capturing the structural and semantic differences between German and other languages.

During the **analysis** phase, a parsing algorithm constructs a tree of the source language, by using a monolingual lexicon and the included grammar rules. The analysis algorithm backs off to a shallower analysis on the phrasal level, when the engine is not able to process the full tree. This might be because the input does not conform with the internal grammar, because the constituents chosen are not consistent, if a grammatical phenomenon is not covered by the grammar, or in case the grammar misses the required lexical entries.

The analysis tree is consequently used for the **transfer** phase, where deep representations of the source are transferred into deep representations of the target language, by using a bilingual lexicon based on canonical forms and categories. The **generation** phase creates the target sentence on the lexical level, by employing inflection and agreement rules between the dependent target language structures.

3.10.3 Deep features for empirical enhancement

Whereas deep techniques indicate good coverage of several linguistic phenomena, each of the three phases may often come across serious robustness issues and the inability to fully process the given sentence. Erroneous analysis from the early phases aggregated along the pipeline may cause further sub-optimal choices in the later phases and severely deteriorate the quality of the produced translation. Preliminary analysis [Federmann and Hunsicker, 2011] has shown that this is the case for source sentences that are in the first place ungrammatical or have a very shallow syntax with many specialized lexical entries, as is the case for the industry-oriented paradigm of WP2. These conclusions are strengthened by the outcome of the intrinsic evaluation, detailed in Section 4.

For this purpose, we take the stance to use output of a supportive statistical machine translation engine. This is done in two aspects: (a) train statistical machine translation to automatically post-edit the output of the transfer-based system (b) use the post-edited or the SMT output in cases where the transfer-based system is of lower performance. This is done through an empirical selection mechanism that performs real-time analysis of the produced translations and automatically selects the output that is predicted to be of a better quality [Avramidis, 2011]. Figure 2 shows the overall architecture of Pilot 1 for DE-EN.

3.10.4 Automatic post-editing using SMT

For automatic post-editing of the transfer-based system, a serial Transfer+SMT system combination is used, as described in [Simard et al., 2007]. The first stage is translation of

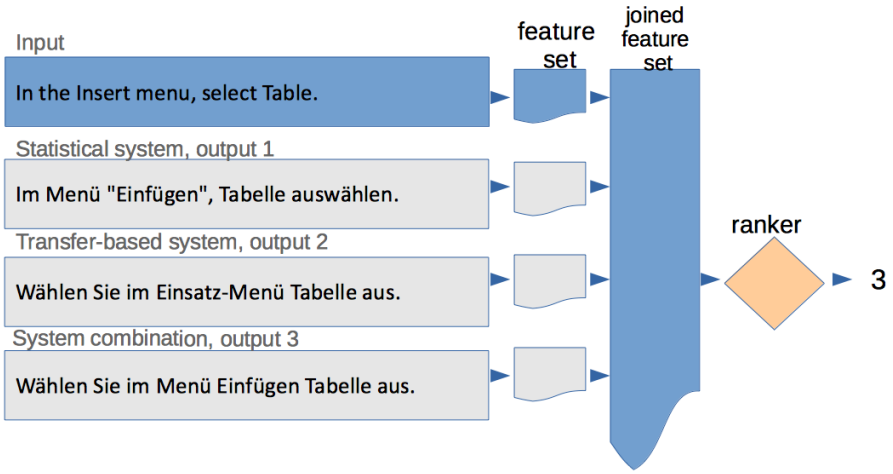


Figure 2: Architecture of System Combination.

the source language part of the training corpus by the transfer-based system. The second stage is training an SMT system with 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 this way, the improvement of up to 6 absolute BLEU points is achieved for both translation directions. Nevertheless, the method on its own could not outperform the SMT system trained on a large parallel corpus (Pilot 0). The example in Figure 2 (which is actually the first answer in the test set) nicely illustrates how the statistical post-editing operates. While the original SMT output used the right terminology ("Menü Einfügen" – "insert menu"), the instruction is not formulated in a very polite manner. In contrast, the output of the transfer-based system is formulated politely, yet mistranslating the menu type. The serial system combination produces a perfect translation. In this particular case, the machine translation is even better than the human reference ("Wählen Sie im Einfügen Menü die Tabelle aus.") as the latter is introducing a determiner for "table", which is not justified by the source.

3.10.5 Selection Mechanism

The selection mechanism is based on encouraging results of previous projects including Euromatrix Plus [Federmann and Hunsicker, 2011], T4ME [Federmann, 2012], QTLaunchPad [Avramidis, 2013, Shah et al., 2013]. It has been extended to include several deep features that can only be generated on a sentence level and would otherwise blatantly increase the complexity of the transfer or decoding algorithm. In Pilot 1, 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 deep features are used to build a machine-learned ranker against training preference labels. Preference labels are part of the training data and indicate which system

output for a given source sentence is of optimal quality. Preference labels are generated either by automatic reference-based metrics, or derived from human preferences. The ranker is a result of experimenting with various combinations of feature set and machine learning algorithms and choosing the one that performs best on the QTLeap corpus.

The selection mechanism is based on the "Qualitative" toolkit that was presented in MT Marathon, as an open-source contribution by QTLeap. Eleftherios et al. [2014]

Feature sets We started from feature sets that performed well in previous experiments and we experimented with several extensions and modifications. In particular:

- Basic syntax-based feature set: unknown words, count of tokens, count of alternative parse trees, count of verb phrases, parse log likelihood. Parse was done with Berkeley Parser and features were extracted from both source and target. This feature set has performed well as a metric in WMT11 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. This featureset combination got the best result in WMT13 quality estimation task.
- 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 We 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.

The classifiers where trained on MT outputs of all systems that participated in the translation shared tasks of WMT (years 2008-2014). We also experimented on several sources of sentence level preference labels, in particular human ranks, METEOR and F-score. We chose the label type which maximizes if possible all automatic scores, including document-level BLEU.

Best combination The optimal systems are using:

1. the *basic syntax-based feature set* for English-German, trained with Support Vector Machines against METEOR scores.
2. the *advanced syntax-based feature set* for German-English, trained with Linear Discriminant Analysis against METEOR scores as well.

Table 1 shows the results of the selection mechanism on the test set in terms of the contribution of the three systems: Transfer-based, SMT, and the linear Transfer+SMT combination. It is notable that the mechanism in many cases favors transfer-based output, which is an indication that the deep features are active as one would have expected a bias towards SMT for a shallower selection mechanism. However, this first hypothesis needs to be confirmed by further studies that are planned.

Lang.	Transfer	SMT	Transfer+SMT
de-en questions	45.2%	33.3%	23.8%
en-de answers	42.5 %	16.3%	50.5%

Table 1: Percentages chosen automatically by the selection mechanism from each of the systems. Percentages which sum more than 100% indicate ties. When ties occur, there is a preset order of preference SMT, Transfer, Transfer+SMT.

3.10.6 Steps towards further pilots

Depending on the results and insights gained from further evaluation of Pilot 1, future Pilots may in cooperation with developments in WP4 and 5 include modules for improved handling of lexical items such as terminology, MWEs, untranslatables, etc. and also for integrating deep structural information from external parsing with the selection mechanism, the rule-based analysis, or the statistical system.

4 Intrinsic Evaluation

This section describes the intrinsic evaluation of the Pilot 1 results, starting with automatic measures and then describing a manual evaluation study.

4.1 Automatic Evaluation

In an earlier version of this Deliverable, translation outputs from MT Pilot 0 (baselines) for the entire test set (questions and answers) from the project languages into English were evaluated using three automatic metrics: BLEU,[11] word-level F-score (wordF) and character-level F-score (charF).[12] F-scores are calculated on 1-grams, 2-grams, 3-grams and 4-grams and then averaged using arithmetic mean. The final score is obtained in the usual way, it is the harmonic mean of precision and recall. Although BLEU is certainly the most used automatic metric, F-score has been shown to correlate better with human judgments, especially if n-grams are averaged using arithmetic instead of geometric mean. We also calculated character level F-score because all the target languages are morphologically rich to more or less extent.

It has been decided in the consortium to slightly extend the evaluation method starting with the first "deeper" Pilot 1 by following the extrinsic (User) evaluation scenario in the project, where both language directions are evaluated separately on different parts of the test set, namely on the questions only for the direction into English and on the answers

[11] The previous version of this Deliverable used `multi-bleu.perl`. This version uses the official BLEU script `mteval-v13a.pl --international-tokenization`, which standardizes also the tokenization to allow replication (`ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13a.pl`).

[12] Using `rgbF.py` (`http://www.dfki.de/~mapo02/rgbF/`)

only for the direction from English to the project languages. Table 2 shows the scores for the baselines (Pilot 0) based only on the questions (translation into English) and contrasts them with the results for Pilot 1. Table 3 shows the scores for the baselines (Pilot 0) based on only the answers (translation from English) and again contrasts them with the results for Pilot 1.

Note that the scores for the Spanish systems have been modified to display a fairer comparison. The scores in deliverable D2.2 claimed a BLEU score of 52.3 for the es→en direction and a BLEU score of 46.4 for the en→es direction for Pilot 0. This system was trained with a larger set of corpora whereas Pilot 1 has been trained with the Europarl corpus only due to time efficiency reasons. We here show results for Pilot 0 trained with Europarl only, although we will train the deep MT with a larger corpora when the deep MT stabilizes.

questions		bg→en	cs→en	de→en	es→en	eu→en	nl→en	pt→en
Pilot 0	BLEU	29.7	26.4	43.0	39.2	25.2	36.9	22.6
	wordF	22.8	30.3	44.6	53.4	29.3	39.3	26.6
	charF	46.7	55.2	64.9	71.7	51.5	59.6	51.0
Pilot 1	BLEU	27.7	26.9	43.3	16.1	4.7	34.1	12.0
	wordF	22.4	31.2	43.8	23.2	12.8	37.3	18.5
	charF	47.4	55.5	63.4	47.4	41.1	58.2	40.8

Table 2: BLEU scores, word-level and character-level F-scores for baseline (Pilot 0) and Pilot 1 English translation outputs.

answers		en→bg	en→cs	en→de	en→es	en→eu	en→nl	en→pt
Pilot 0	BLEU	25.3	30.2	41.7	47.2	24.3	31.0	19.3
	wordF	25.6	27.1	42.2	47.0	22.9	31.6	21.7
	charF	46.7	52.9	64.7	67.3	56.0	55.8	47.3
Pilot 1	BLEU	24.5	30.6	33.0	17.1	12.8	23.0	19.7
	wordF	25.0	32.8	30.2	21.2	14.2	24.8	21.5
	charF	46.6	57.1	57.4	47.5	47.0	52.5	46.6

Table 3: BLEU scores, word-level and character-level F-scores for baseline (Pilot 0) and Pilot 1 translation outputs from English.

The results show that a majority of the Pilot 1 systems perform at least comparable to their Pilot 0 counterparts.[13] The entry-level systems for Spanish and Basque are weak at this point. However, given the fact that only one partner, UPV/EHU, was responsible for developing them basically from scratch, we see it as a progress that these systems are operational, even though the quality of translations they provide is currently low. It should be noted that all other partners were only responsible for one language, and some of them also had the advantage of a basic pre-existing system that they could build upon.

[13]The differences in automatic scores are rather small, typically less than 1 point, and cannot be reliably interpreted as one of the systems being clearly better than the other. This is a well known problem of the automatic metrics, which we address by using other more fine-grained evaluation methods, as described in later sections. Thus we consider 8 out of the 14 Pilot 1 systems (bg→en, cs→en, de→en, nl→en, en→bg, en→cs, en→pt and en→nl), even if the en→nl only in in the charF metric, to perform on par with Pilot 0.

It is clear that close and intense cooperation of UPV/EHU with CUNI and other partners will be undertaken to achieve high-quality translation in future pilots.

Several other systems achieved low scores in the evaluation (pt→en, en→de, en→nl), but they perform well in the opposite translation directions, which shows that some components of the systems still need significant improvements, but the general approach seems to be suitable for these languages.

4.1.1 Further results

In addition to the scores discussed above, the translation errors were analyzed using Hjerson [Popović 2011], an automatic tool for error analysis which provides a categorization into five classes:

- word form (agreement errors, tense, capitalization, part of speech)
- word order
- omission
- addition
- mistranslation (general mistranslations, terminology errors, style, punctuation and any changes to wording)

For each error class, the tool provides raw error counts together with error rates (raw counts normalized over the total number of words in the translation output). In addition, block error counts and block error rates are calculated as well, where the block refers to a group of successive words belonging to the same error class.

The tool is language independent. It requires the translation output and a reference, both in full form and lemmatized.

During the evaluation experiments, it has been observed that there is a number of capitalization errors (or inconsistencies between the reference and the translation), such as “OpenOffice” vs. “openOffice”, “VLC” vs “vlc”, etc. Therefore we subsequently calculated capitalization error rates as difference between word form error rate of true-cased texts and word form error rates of lower-cased texts that are displayed in the table below. The pure morphological errors are those obtained with lower-cased texts. In order to arrive at a fair treatment of the prevalent items in the input such as “File > Save As” or URLs, we have reported block error rates instead of word-level error rates.

The results for all language pairs are presented in Tables 4 and 5. The error classification results are presented below, in the form of block error rates (lower is better). The error rates read as follows. For example, for Pilot 0 bg->en, 14.9% of the word groups in the translation output are mistranslated in comparison to the human reference (i.e. these words are different than the reference words). So, if the system has translated 100 words, ca. 15 (consecutive blocks of) words consist of other words than found in the reference.

It can be seen that mistranslations are the most frequent errors in all translation outputs. In Pilot 1 the number of mistranslations is reduced for Czech and Portuguese outputs and largely increased for the Spanish output. For the Basque language, a very large portion of errors is morphological, although a considerable number of morphological errors is also present for other languages (except English).

Reordering errors are improved for Czech-English, German-English and Dutch-English. For the rest of the systems either no change or an increase of reordering errors can be

	questions	bg→en	cs→en	de→en	es→en	eu→en	nl→en	pt→en
Pilot 0	form	1.8	1.6	1.2	0.9	1.9	1.3	1.2
	order	7.1	9.7	6.5	5.0	9.5	6.3	10.6
	omission	7.6	4.8	4.4	3.9	7.0	4.9	4.4
	addition	2.9	4.9	2.8	2.5	3.8	4.6	5.1
	mistranslation	14.9	15.1	12.2	10.3	15.9	13.0	18.7
Pilot 1	form	1.9	1.7	1.1	2.0	4.6	1.6	2.4
	order	7.7	9.3	5.6	9.5	15.0	5.8	11.9
	omission	7.3	4.3	3.4	4.7	8.8	3.5	2.8
	addition	3.4	5.0	3.6	6.6	3.1	4.3	8.9
	mistranslation	15.2	15.4	12.8	18.4	26.0	14.8	21.6

Table 4: Class error rates for Pilot0 and Pilot1 English translation outputs.

	answers	en→bg	en→cs	en→de	en→es	en→eu	en→nl	en→pt
Pilot 0	form	6.4	7.0	4.4	3.4	11.1	4.4	6.0
	order	4.0	5.9	5.7	4.2	8.1	6.4	8.0
	omission	4.5	5.1	4.6	2.9	5.7	4.3	5.3
	addition	2.9	4.4	3.7	3.7	4.4	5.3	4.6
	mistranslation	18.9	15.4	11.9	12.1	14.5	14.8	19.8
Pilot 1	form	6.0	7.3	4.0	7.3	13.0	5.7	4.8
	order	4.7	7.0	5.6	8.6	8.8	8.5	8.1
	omission	4.5	5.3	3.0	8.8	3.3	6.0	5.5
	addition	3.4	3.1	7.4	2.0	7.7	5.2	6.0
	mistranslation	18.9	14.7	13.5	20.5	17.5	17.2	18.8

Table 5: Class error rates for Pilot0 and Pilot1 translation outputs from English.

observed. A largest increase can be noted for Spanish-English systems in both directions as well as for the Basque-English system. Although this observation has been confirmed by the manual error annotation (see Section 4.2), the reason for this might be that deeper systems produce structurally different translations that might still be correct, yet have a different word order. This needs to be analyzed in more detail.

All in all, the numbers are in line with comparable previous studies. It should be taken into account that the described differences are not related only to the language characteristics, but also to the nature of the (different) systems. Note that it is not a goal of the project to compare between languages, but rather to show an improvement between Pilots for each language pair.

4.2 Manual Evaluation of Pilot 1 Results

One of the goals of the QTLeap project is to analyze the performance of the deep MT Pilots (starting with Pilot 1) with the help of manual markup and error analysis to get better insights into the systems weaknesses and see what could be improved from Pilot to Pilot.

We built upon experience with error annotation acquired in the QTLaunchPad project. One insight is that it is of great advantage if explicit error markup is performed in combination with post-editing as the corrections often help to better understand what "ideal" translation an annotator had in mind.

For Pilot 1, all QTLeap partners have volunteered to post-edit ca. 200 answers from the test set (Batch 2 of the QTLeap corpus) and to provide an explicit error markup of ca. 100 answers using the MQM (Multidimensional Quality Metrics) framework. As the Pilot 1 systems are first deep systems that exhibit many known issues, the QTLeap partners have decided to follow the best practice in industry by performing single annotation to save resources.[14] The annotators have been trained in a webinar[15] and they have received detailed annotation guidelines.[16] Annotation was done using an instance of the translate5 tool[17] hosted at DFKI.

The corpus for annotation has been prepared so that the top 100 interactions shown to the annotators were the same segments that are used in the extrinsic user evaluations of QTLeap (see for instance Deliverable D3.6), followed by the remaining 900 interactions of the test set. An excerpt of the instructions from the introductory slides[18] shown in the webinar is presented below:

- First post-edit a segment and then do the error annotation (so that both are coherent)
- You can skip post-editing and annotation for trash segments
- You can do post-editing for a segment, but skip MQM if it seems unfeasible
- Start with the first 100 interactions and continue through the rest until you've reached the target numbers

[14] Double annotation was performed only for Portuguese.
[15] https://www.youtube.com/watch?v=cY_pBR9OvkI
[16] http://qtleap.eu/wp-content/uploads/2015/03/Annotation-Guidelines-Pilot1.pdf
[17] http://www.translate5.net/
[18] http://qtleap.eu/wp-content/uploads/2015/03/Annotation-Pilot1.pdf

- By default, you can annotate every segment and only skip completely unintelligible segments (or error annotations where it is simply pointless to try and identify individual errors)

As can be seen, some freedom was left to the annotators to skip or select segments. The rationale for this was that partners had pointed out that some of the Pilot 1 annotations are still prone for (known) errors and thus difficult to annotate. Moreover, interpreting the annotations is also difficult if too many errors are involved. This freedom comes with the cost of a certain subjectivity in the error annotation. For the future comparison of Pilot 1 and Pilot 2, one option is to annotate the same segments for Pilot 2 that have been annotated for Pilot 1. Alternatively, one could do a blind (randomized) annotation of identical segments from both pilots. This would be more effort, but allow for a better comparison and also to cross-check the given annotations for Pilot 1.

The post-edits and annotations will serve as reference for future improvements. For the time being, we will present some first statistics on the post-edits and MQM annotations.

4.2.1 Post-Editing

Table 6 shows the edit distance between Pilot 1 translation outputs and their post-edits.

WER	true case	lowercase
en-bg	33.3	29.7
en-cs	24.7	24.3
en-de	28.4	28.1
en-es	56.1	54.3
en-eu	72.5	68.9
en-nl	23.8	23.5
en-pt1	52.4	50.2
en-pt2	57.5	55.8

Table 6: Edit distance (Word Error Rate – WER) between Pilot 1 translation outputs and their post-edits.

We plan to correlate these with the results of the user evaluation where volunteers will compare the results of the baseline systems with the output of Pilot 1. One obvious hypothesis to be confirmed is that Pilot 1 output is preferred when little post-editing would be needed to make it good. In the course of that correlation, we also plan to compare the post-editing distance between the post-edit results and the human reference translations as well as between the raw baseline output and the human reference.

4.2.2 Annotation with an MQM Error Metric

In this section, we provide an overview of the explicit analytic error markup performed for all languages for Pilot 1.

As described above, annotators were asked to annotate 100 segments. For segments to be annotated, they had to pass a two-stage filtering process. First, only those segments that were selected for post-editing were considered for subsequent annotation. Second, from among those sentences that were post-edited, annotators were allowed to skip those that were too difficult to annotate. For example, a sentence might be easy to post-edit but the exact nature of the errors was unclear, in which case the annotator would skip

it. As a result, no "perfect" sentences were considered for annotation and most extremely bad sentences were excluded as well.

Annotations were done using a set of Multidimensional Quality Metrics (MQM) issues selected specifically for the evaluation of machine translation results. Originally developed in the QTLaunchPad project and augmented with the "Terminology" issue type that is crucial in QTLeap, this set of issues is shown in Figure 3.

It is important to note that MQM issues are hierarchical in nature. For example, the MQM issue type Grammar has three subtypes (Word form, Word order, and Function words) and is itself a subtype of Fluency. As a result, the number of errors for any given branch in the hierarchy includes the branch itself and all subtypes beneath it.

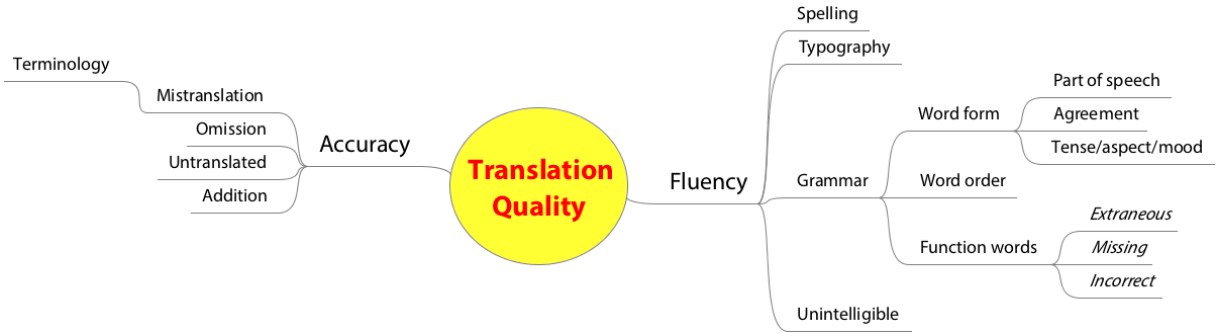


Figure 3: Hierarchy of error types for MT diagnosis in QTLeap.

		Target Language						
MQM Issue Type		Basque	Bulgarian	Czech	Dutch	German	Spanish	Portuguese
Accuracy								
Accuracy (general)		0.0%	0.6%	0.4%	0.0%	0.0%	0.0%	1.0%
Mistranslation		1.6%	16.3%	24.9%	23.8%	21.8%	2.5%	13.6%
Terminology		11.9%	2.1%	15.5%	0.6%	14.6%	17.6%	14.2%
Mistranslation (subtotal)		***13.5%***	***18.3%***	***40.4%***	***24.4%***	***36.4%***	***20.0%***	***27.8%***
Addition		4.7%	0.8%	1.2%	7.3%	12.4%	6.1%	1.5%
Omission		3.8%	0.2%	4.9%	7.9%	8.0%	3.3%	2.6%
Untranslated		4.9%	6.0%	0.8%	7.3%	1.7%	18.8%	4.9%
Accuracy (subtotal)		***26.9%***	***25.9%***	***47.8%***	***47.0%***	***58.4%***	***48.3%***	***37.7%***
Fluency								
Fluency (general)		0.0%	1.6%	8.2%	0.0%	0.0%	0.0%	0.0%
Typography		8.7%	10.1%	0.8%	6.1%	11.8%	0.0%	10.7%
Spelling		1.0%	15.6%	0.4%	0.0%	0.3%	8.2%	2.0%
Grammar	Grammar (general)	1.5%	0.2%	4.5%	0.0%	0.0%	2.5%	0.0%
	Word order	15.9%	3.5%	9.4%	14.6%	8.3%	9.8%	9.7%
	Word form (general)	0.6%	9.5%	5.7%	0.0%	2.5%	0.0%	0.3%
	Agreement	9.0%	12.1%	1.2%	12.8%	6.9%	5.9%	1.5%
	Part of speech	4.4%	4.3%	4.5%	5.5%	1.1%	7.2%	3.6%
	Tense/aspect/mood	4.9%	0.2%	9.0%	9.8%	1.1%	3.7%	7.0%
	Word form (subtotal)	***18.8%***	***26.1%***	***20.4%***	***28.0%***	***11.6%***	***16.8%***	***12.4%***
	Function words (general)	0.0%	0.2%	0.0%	0.0%	0.0%	0.0%	0.1%
	Extraneous	0.0%	4.1%	0.8%	0.0%	0.3%	0.0%	8.7%
	Incorrect	5.9%	8.0%	0.4%	3.7%	7.2%	5.5%	4.7%
	Missing	19.9%	3.7%	2.9%	0.6%	1.9%	8.2%	13.1%
	Function words (subtotal)	***25.7%***	***16.0%***	***4.1%***	***4.3%***	***9.4%***	***13.7%***	***26.6%***
	Grammar (subtotal)	***61.9%***	***45.9%***	***38.4%***	***47.0%***	***29.2%***	***42.7%***	***48.7%***
Unintelligible		1.5%	0.8%	4.5%	0.0%	0.3%	0.8%	0.8%
Fluency (subtotal)		***73.1%***	***74.1%***	***52.2%***	***53.0%***	***41.6%***	***51.7%***	***62.3%***

Figure 4: Overview of MQM issue types from Pilot 1 annotations.

An overview of the issues identified in this annotation task can be seen in Figure 4. Two important notes apply in interpreting this table: (1) The Portuguese column represents the average of two annotators while all others are from single annotators. (2) The results summarized are not intended to be compared among the languages. As a result of system and language differences, considerable variations are to be expected. The figures will be useful, however, in determining how system performance changes with regard to specific issues.

4.3 Manual Evaluation: Results

By going over some of the translations, a number of frequent problems for the various language pairs can be identified immediately.

A large amount of errors in the translations are domain specific issues. The various monolingual and translation components are general purpose, and trained on generic corpora. For most language pairs, only a limited amount of tuning to the QTLeap domain has been performed.

- Identification of named entities. In the QTLeap domain a lot of named entities occur. These include classical named entities such as product names (Google Play, Itunes Store), but also several domain specific, that are different from those normally treated by named entity tools: labels of menu items, buttons, tabs, etc. Those named entities often are not easily identified by capitalization, typography, or punctuation.

- Terminology, and other mistakes for lexical translations. In the domain of QTLeap, a word such as 'driver' has a very specific meaning: a piece of software which manipulates a hardware device. The translation component often produces the translation of the generic meaning of the word – in this example, the translation might be equivalent to 'chauffeur'. A similar example is the 'main menu' which gets translated into the equivalent of 'important menu'.

- In some cases, the translation component generates translations of a word with an inappropriate part-of-speech. For instance, we noted cases where a phrase such as 'to slide' on the English side received a translation for the noun reading of 'slide', as in 'overhead slides'.

- For the systems which employ the TectoMT pipeline architecture, a mistake in English analysis will lead to further problems in the further components in the pipeline, and ultimately will cause errors in the translation output. Similarly, it is hard to recover from mistakes in the transfer component. For future experiments, it may be fruitful to consider a set-up in which transfer can suggest multiple alternative translations so that the synthesis component can decide upon the best fitting translation.

Clearly, most of these generic problems are not very novel or surprising, and in fact in the QTLeap project, work packages are devoted to Named Entity recognition and Word Sense Disambiguation to help overcome some of the problems.

The following per-language descriptions present figures only for high-level distinctions and do not break down all MQM categories presented in Figure 4.

4.3.1 Annotation Results for Basque

The Basque results (Figure 5) show that Fluency was a considerable problem for the system, with Fluency errors comprising almost three quarters of all identified issues. Within Accuracy, Terminology errors were the largest group, and within Fluency, the overwhelming majority of errors were Grammar errors. Over 40% of the Grammar errors belong to the Function words category, where a high number of missing postpositions were identified. This is a well-known difficulty when translating from English into an agglutinative language and a clear path for improvement for the next Pilots. Errors in Word formation, particularly those emerging from agreements and verb phrases, and incorrect word orderings have also been highlighted as clear weaknesses.

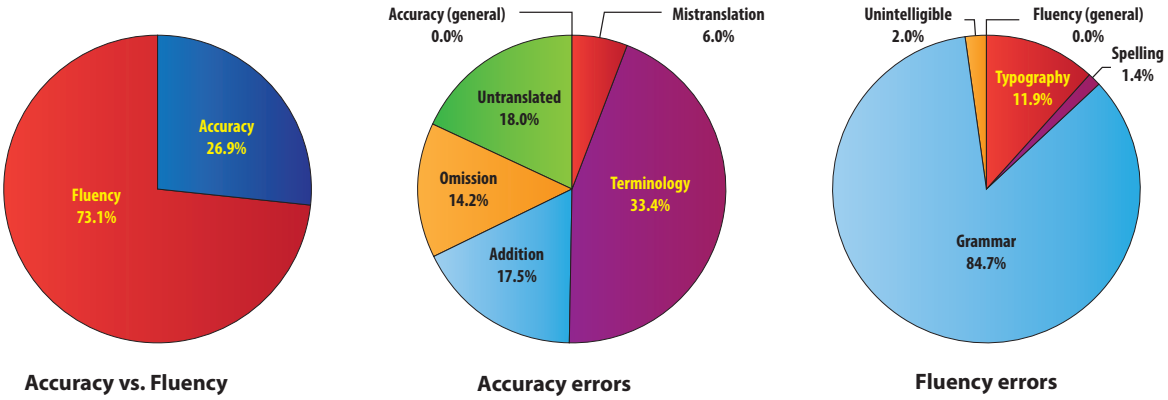


Figure 5: Error distribution for Basque.

4.3.2 Annotation Results for Bulgarian

An examination of the Bulgarian results (Figure 6) indicates that general Fluency is considerably more of a relative issue than Accuracy. Within Accuracy errors, general Mistranslation is by far the most common problem, followed by a significant number of instances of Untranslated (i.e. text remains in the source language). Within Fluency, Grammar is the most significant error type, followed by a large number of Spelling problems and Typography errors.

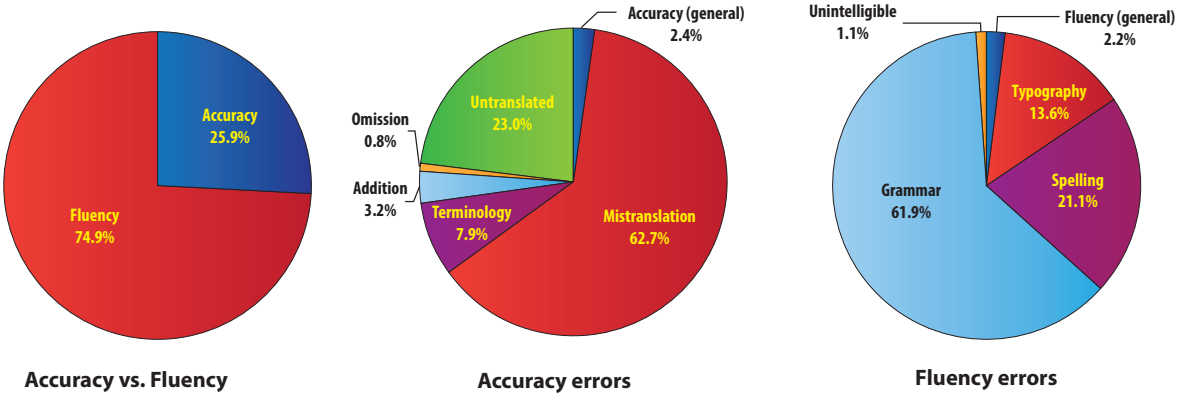


Figure 6: Error distribution for Bulgarian.

4.3.3 Annotation Results for Czech

For Czech (Figure 7) the number of Accuracy and Fluency errors are quite similar, indicating that problems in Czech are evenly distributed between the two categories. Within Accuracy, general Mistranslation accounts for over half of the errors and Terminology for approximately one third. As Terminology is relatively tractable, the importance of Terminology errors indicates one potential avenue for the system improvement. In addition, the Czech system was relatively likely to omit content present in the source (versus leaving it untranslated). Within Fluency, Grammar is the most significant error type, with very few Spelling or Typography errors. The Czech evaluator was particularly likely to mark issues with general Fluency.

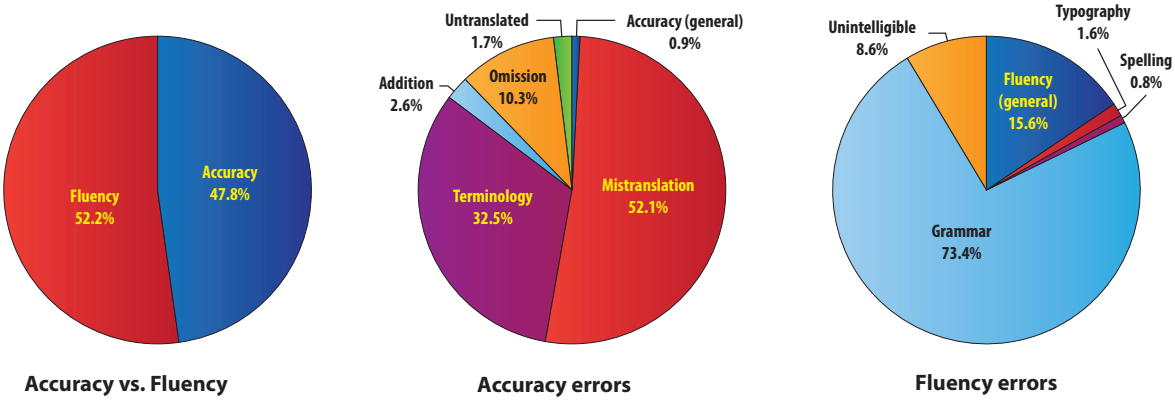


Figure 7: Error distribution for Czech.

4.3.4 Annotation Results for Dutch

For Dutch (Figure 8) the number of Accuracy and Fluency errors are quite similar, indicating that problems in Dutch are evenly distributed between the two categories. Note that Dutch had fewer annotated segments than the other languages – see Figure 4. Within Accuracy, approximately one half of all issues are simple Mistranslation, with very few Terminology errors. The remaining almost one half of issues is evenly distributed among Additions, Omissions, and Untranslated text. Fluency errors in Dutch consisted entirely of Grammar (almost 90%) and Typography, with no other Fluency issues noted. This result indicates that the system seems to perform very well in Spelling and produces very little "Unintelligible" results.

4.3.5 Annotation Results for German

For German (Figure 9) there are slightly more Accuracy than Fluency errors. Within Accuracy, no single issue type predominates, with approximately 37% of issues in Mistranslation, 25% in Terminology, 21% in Addition, and 14% in Omission. Untranslated text was relatively infrequent. Fluency errors for German were largely Grammar and Typography. Although the data from these annotations do not support proper cross-language or cross-system comparisons, the German system performed particularly poorly in Typography, indicating one area where the system could be readily improved.

Among first observations for Pilot 1 there is a systematic problem with English imperative form of verbs as in "Go to X" that are often misinterpreted as nouns. Another

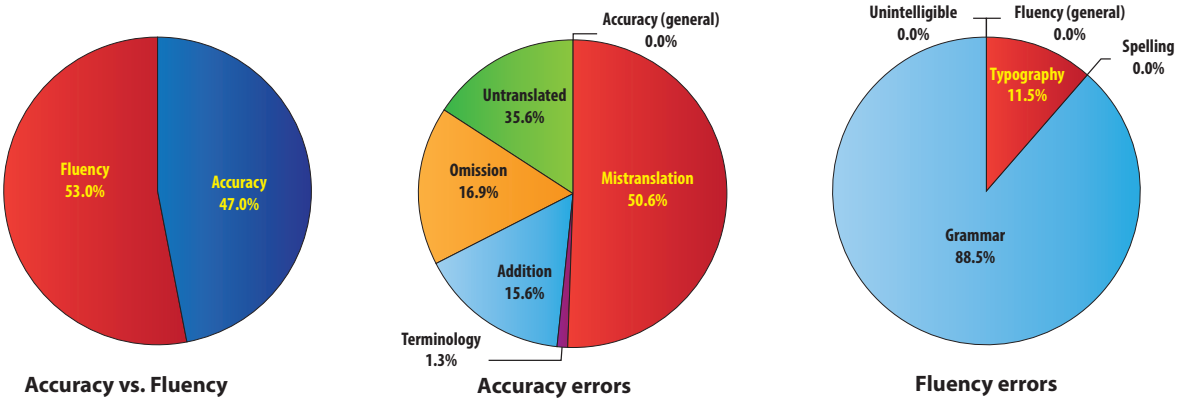


Figure 8: Error distribution for Dutch.

observation is that some problems in the translations relate to the semi-formal nature of some source formulations like the prevalent "Click where it says Y" that is translated too literally into German "Klicken Sie, wo es sagt Y" instead of "Klicken Sie Y" ("Click Y"). The same feature holds for Czech. Partial interpretation of MWEs (leading to parsing errors of the transfer-based component) and missing hyphenation of MWEs in German is another frequently observed pattern.

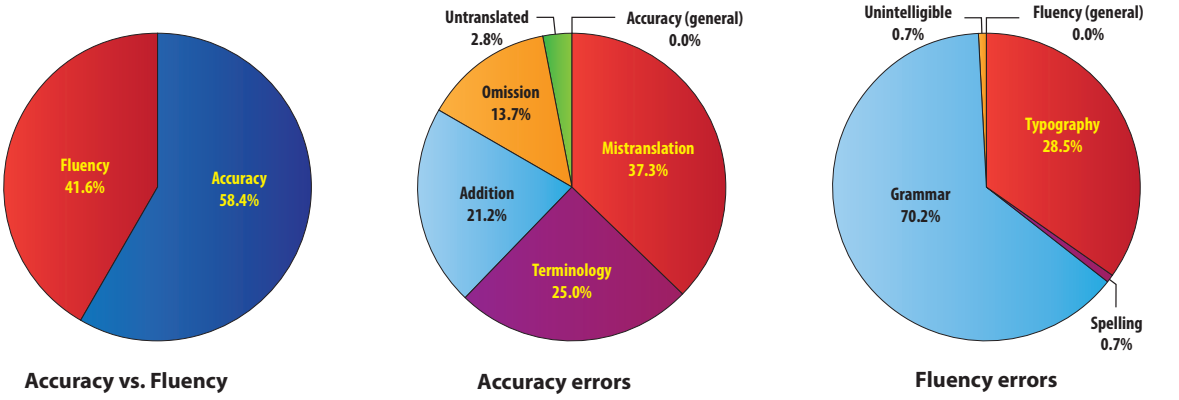


Figure 9: Error distribution for German.

4.3.6 Annotation Results for Portuguese

Unlike the other systems, the Portuguese Pilot 1 was evaluated by two individuals. They exhibited markedly different profiles in terms of the numbers of errors they marked. Annotator A marked an average of 4.8 errors/segment and Annotator B marked an average of 7.2 errors/segment. Although this section presents only the average of the two annotators, the generally low agreement between them is an issue that will complicate future comparisons. However, Annotator B was significantly more likely to mark "Minor" issues (500) than was Annotator A (168). Considering only "Major" issues, their performance was much closer (Annotator A: 310; Annotator B: 259). Note as well that work in QT-LaunchPad showed that there is considerable need for calibration of annotators, with inter-annotator agreement typically in the 0.4 to 0.5 (Cohen's kappa) range for first time annotators. While this number seems low, it compares favorably with agreement rates for simpler related tasks, such as quality ranking. See [Lommel et al., 2014].

For Portuguese (Figure 10), Fluency errors outnumbered Accuracy errors by almost two to one, with both annotators showing high agreement (within approximately half a percentage point) in the relative proportion of errors at this level. (In other words Annotator B systematically identified more errors in both categories.) There was considerable disagreement in the breakdown within Accuracy, with Annotator B showing more variety in the use of the various issue types. Overall, however, they both agreed that outright Mistranslation (or Terminology, which is a subtype of Mistranslation) comprised the bulk of the errors, with relatively little Addition or Omission, but a fairly large proportion of Untranslated content.

Fluency errors for Portuguese comprised largely Grammar and Typography, with Annotator B significantly more likely to identify Typography errors (117 vs. 29) and Word order (80 vs 43). Both annotators found the same number of Word form errors (73), but this figure represents a much lower percentage of errors found for Annotator B. Despite the disagreement about specific issue types, they both agreed that approximately 50% of all errors overall (78% of Fluency errors) were related to Grammar.

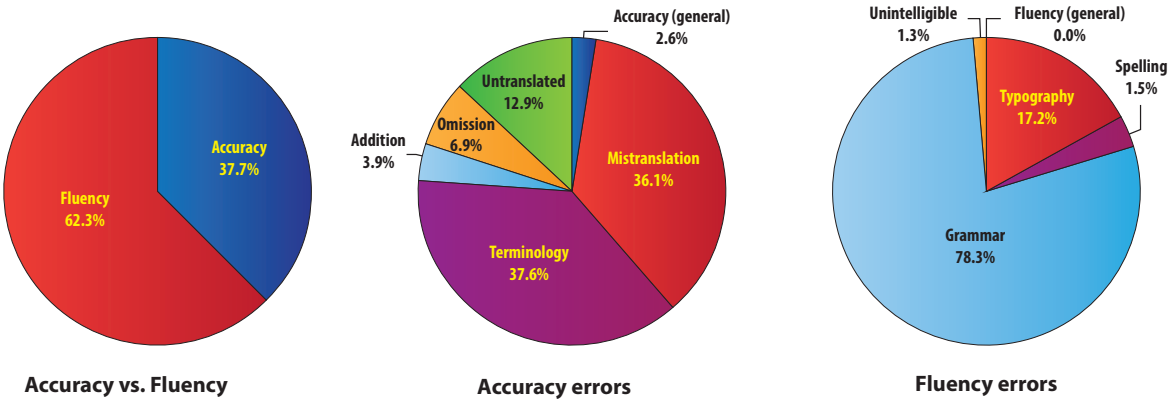


Figure 10: Error distribution for Portuguese.

Although errors per segment are not generally reported, a comparison of the two reviewers shows how sensitive the annotation is with regards to individual annotators. As Figure 11 shows, if we divide sentences into quality "bands" (1 to 2 errors, 3 to 5 errors, and more than 5 errors), we see that Annotator A provides a much more favorable view of the quality of the Portuguese system than does Annotator B. In addition, as noted above, Annotator B was much more likely to consider errors to be "Minor" (rather than "Major") than was Annotator A. As a result, if only "Major" errors are considered, the performance of the two Annotators is much closer than is apparent.

4.3.7 Annotation Results for Spanish

For Spanish (Figure 12) the number of Accuracy and Fluency errors are quite similar, with slightly more Fluency than Accuracy errors. The Spanish system was particularly likely to leave content untranslated and issues related to Mistranslation were particularly likely to be Terminology errors, rather than general Mistranslation errors. Terminology errors includes issues with specialized vocabulary as well as product names and user interface strings. Within Fluency, Grammar was by far the most common problem, with significant numbers of Spelling errors, and all other error types relatively negligible. A considerable amount of Spelling errors concern irregular word formations. Verbs whose lemma is slightly altered when conjugated are a good example. As Spelling is particularly

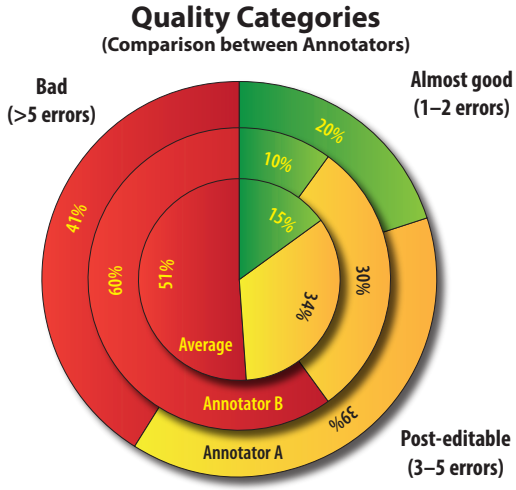


Figure 11: Errors bands for two Portuguese annotators.

tractable with existing MT methods, this result suggests one relatively straight-forward path to system improvement.

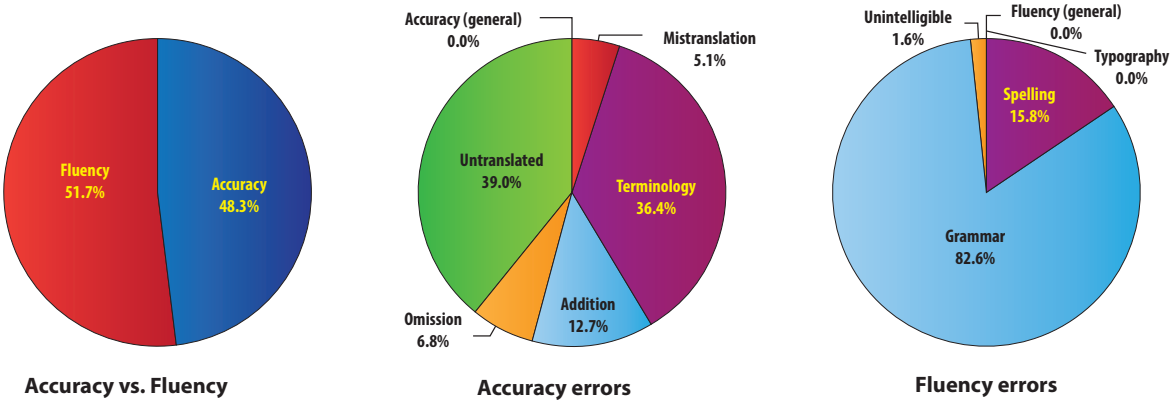


Figure 12: Error distribution for Spanish.

5 Final remarks

In our effort towards Pilot 1, we managed to achieve our main goal of developing a set of entry-level deep MT systems for all 14 language pairs, delivered in time and fully operational. The EN-CS, EN-DE, and BG-EN systems already existed prior to the QTLeap project, but were significantly improved and adapted to the QTLeap domain; the CS-EN and DE-EN systems also existed in part, but were lacking some important components. The remaining 9 systems were developed technically from scratch, by porting the existing systems to new language pairs, incorporating pre-existing language-specific analysis and synthesis tools into them, implementing all of the non-existent components, and further tuning and adapting the systems to the QTLeap domain.

All of the systems are now stable and running, a remarkable achievement made possible by tight and intense cooperation of the project partners, as very diverse tools, developed

by the partners as well as third parties, and often operating with significantly different lingusitic concepts and principles, had to be tightly bound together. Moreover, most of the systems achieve performance comparable to or better than their Pilot 0 counterparts, a clear success given the short time available for developing the Pilot 1 systems, as compared to the strong and long-existing baseline Pilot 0 systems. The already good performance of most of the Pilot 1 systems marks a clear potential of deep MT systems for all language pairs, including the currently weaker ones, to achieve high quality translation in future pilots.

Manual evaluation results have illustrated the systems' weaknesses that will be addressed for future pilots. The distribution of lexical errors (such as mistranslations or terminology), structural issues (such as agreement and word order), and issues like incorrect function words that might be located on different levels shows that there is room for system improvement. For some languages, the more mechanical fluency errors (typography and spelling) are quite frequent, too. Although these usually do not render translations unreadable, they have to be addressed.

References

Itzair Aduriz, María Jesús Aranzabe, Jose Mari Arriola, Aitziber Atutxa, Arantza Díaz de Ilarraza, Aitzpea Garmendia, and Maite Oronoz. Construction of a Basque dependency treebank. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories*, pages 201–204, 2003.

Rodrigo Agerri, Josu Bermudez, and German Rigau. IXA pipeline: Efficient and Ready to Use Multilingual NLP tools. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). ISBN 978-2-9517408-8-4.

Iñaki Alegria, Maria Jesus Aranzabe, Aitzol Ezeiza, Nerea Ezeiza, and Ruben Urizar. Robustness and customisation in an analyser/lemmatiser for Basque. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC). Customizing knowledge in NLP applications Workshop*, 2002.

Juan A Alonso and Gregor Thurmair. The Comprendium Translator system. In *Proceedings of the Ninth Machine Translation Summit*. International Association for Machine Translation (IAMT), 2003.

Eleftherios Avramidis. DFKI System Combination with Sentence Ranking at ML4HMT-2011. In *Proceedings of the International Workshop on Using Linguistic Information for Hybrid Machine Translation (LIHMT 2011) and of the Shared Task on Applying Machine Learning Techniques to Optimising the Division of Labour in Hybrid Machine Translation (ML4HMT-11)*, Barcelona, Spain, 2011. Center for Language and Speech Technologies and Applications (TALP), Technical University of Catalonia.

Eleftherios Avramidis. Sentence-level ranking with quality estimation. *Machine Translation (MT)*, 28(Special issue on Quality Estimation):1–20, 2013.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1): 39–71, 1996.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. A high-performance syntactic and semantic dependency parser. In *COLING (Demos)*, pages 33–36. Demonstrations Volume, 2010.

Bernd Bohnet. Top accuracy and fast dependency parsing is not a contradiction. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING*, pages 89–97. Tsinghua University Press, 2010.

Ondřej Bojar and Miroslav Týnovský. Evaluation of tree transfer system. Technical Report Deliverable 3.4, Charles University in Prague, ÚFAL, 2009. URL http://www.euromatrix.net/deliverables/em_deliv_34.pdf.

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The joy of parallelism with CzEng 1.0. In *Proceedings of the 8th International Conference on*

Language Resources and Evaluation (LREC 2012), pages 3921–3928, İstanbul, Turkey, 2012. European Language Resources Association. ISBN 978-2-9517408-7-7.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

Francis Bond, Stephan Oepen, Eric Nichols, Dan Flickinger, Erik Velldal, and Petter Haugereid. Deep open-source machine translation. *Machine Translation*, 25(2):87–105, 2011. ISSN 0922-6567.

António Branco and Tiago Henriques. Aspects of verbal inflection and lemmatization: Generalizations and algorithms. In Amália Mendes and Tiago Freitas, editors, *Atas do XVIII Encontro Nacional da Associaç ao Portuguesa de Linguística*, pages 201–210, Lisbon, Portugal, 2003. Associação Portuguesa de Linguística.

António Branco and João Silva. Contractions: breaking the tokenization-tagging circularity. In *Lecture Notes in Artificial Intelligence 2721*, pages 167–170. Spinger, 2003. ISSN 0302-9743.

António Branco and João Silva. Evaluating Solutions for the Rapid Development of State-of-the-Art POS Taggers for Portuguese. In Maria Teresa Lino, Maria Francisca Xavier, Fátima Ferreira, Rute Costa, and Raquel Silva, editors, *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, pages 507–510. ELRA, 2004.

António Branco and João Silva. Dedicated Nominal Featurization of Portuguese. In *Lecture Notes in Artificial Intelligence 3960*. Springer, 2006a.

António Branco and João Silva. A Suite of Shallow Processing Tools for Portuguese: LX-Suite. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, 2006b.

António Branco and João Silva. Very High Accuracy Rule-based Nominal Lemmatization with a Minimal Lexicon. In *Actas do XXI Encontro Anual da Associação Portuguesa de Linguística*, 2007.

António Branco, Sérgio Castro, João Silva, and Francisco Costa. CINTIL DepBank Handbook: Design options for the representation of grammatical dependencies. Technical Report di-fcul-tr-11-03, Department of Informatics, University of Lisbon, 2011. URL http://hdl.handle.net/10455/6747.

Michael Collins. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 1–8. Association for Computational Linguistics, 2002.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan Sag. Minimal Recursion Semantics: An Introduction. *Research on Language & Computation*, 3(4):281–332, 2005.

Francisco Costa. Verbal Conjugation in Portuguese. Online service, 2004. URL http://lxconjugator.di.fc.ul.pt.

Matthew S Crouse, Robert D Nowak, and Richard G Baraniuk. Wavelet-based statistical signal processing using Hidden Markov Models. *Signal Processing, IEEE Transactions on*, 46(4):886–902, 1998.

Daniël de Kok and Gertjan van Noord. A Sentence Generator for Dutch. *LOT Occasional Series*, 16:75–90, 2010.

Daniël de Kok, Jianqiang Ma, and Gertjan van Noord. A generalized method for iterative error mining in parsing results. In *Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks (GEAF 2009)*, pages 71–79, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W09/W09-2609.

Daniel de Kok, Barbara Plank, and Gertjan van Noord. Reversible stochastic attribute-value grammars. In *49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon, 2011.

Ondřej Dušek and Filip Jurčíček. Robust multilingual statistical morphological generation models. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 158–164, Sofija, Bulgaria, 2013. Association for Computational Linguistics.

Ondřej Dušek, Zdeněk Žabokrtský, Martin Popel, Martin Majliš, Michal Novák, and David Mareček. Formemes in English-Czech Deep Syntactic MT. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 267–274, Montréal, Canada, 2012. Association for Computational Linguistics.

Avramidis Eleftherios, Poustka Lukas, and Schmeier Sven. Qualitative: Open source python tool for quality estimation over multiple machine translation outputs. *The Prague Bulletin of Mathematical Linguistics*, 102(1):5–16, 2014.

R. E Fan, K. W Chang, C. J Hsieh, X. R Wang, and C. J Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Christian Federmann. Can Machine Learning Algorithms Improve Phrase Selection in Hybrid Machine Translation? In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics. Joint Workshop on Exploiting Synergies between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation*, pages 113–118, Avignon, France, April 2012. European Chapter of the Association for Computational Linguistics (EACL). URL http://www.dfki.de/web/forschung/publikationen/renameFileForDownload?filename=eacl12_cfedermann.pdf&file_id=uploads_1537.

Christian Federmann and Sabine Hunsicker. Stochastic Parse Tree Selection for an Existing RBMT System. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 351–357, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2141http://www.aclweb.org/anthology/W11-2141.pdf.

Georgi Georgiev, Valentin Zhikov, Petya Osenova, Kiril Simov, and Preslav Nakov. Feature-rich Part-of-speech Tagging for Morphologically Complex Languages: Application to Bulgarian. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 492–502, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-19-0. URL http://dl.acm.org/citation.cfm?id=2380816.2380876.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3153–3160, İstanbul, Turkey, 2012. ELRA, European Language Resources Association. ISBN 978-2-9517408-7-7.

Jan Hajič. *Disambiguation of Rich Inflection: Computational Morphology of Czech*. Karolinum, Praha, 2004.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.

Philipp Koehn and Hieu Hoang. Factored translation models. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 868–876, 2007. URL http://www.aclweb.org/anthology/D07-1091.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1557769.1557821.

Arle Richard Lommel, Maja Popovic, and Aljoscha Burchardt. Assessing inter-annotator agreement for translation error annotation. In *MTE: Workshop on Automatic and Manual Metrics for Operational Translation Evaluation*. LREC, 2014.

David Mareček, Martin Popel, and Zdeněk Žabokrtský. Maximum entropy translation model in dependency-based mt framework. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 201–206, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-1730.

Pedro Martins. Desambiguação Automática da Flexão Verbal em Contexto. Master's thesis, Universidade de Lisboa, Lisboa, Portugal, 2008. URL http://hdl.handle.net/10451/13862.

Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting of the Association*

for Computational Linguistics, pages 91–98. Association for Computational Linguistics, 2005a.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005b.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932, 2007. URL http://www.cs.brandeis.edu/~marc/misc/proceedings/acl-2007/EMNLP-CoNLL2007/pdf/EMNLP-CoNLL200796.pdf.

J. Nivre, J. Hall, and J. Nilsson. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219, 2006.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

Michal Novák, Anna Nedoluzhko, and Zdeněk Žabokrtský. Translation of "it" in a deep syntax framework. In Bonnie L. Webber, editor, *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Workshop on Discourse in Machine Translation*, pages 51–59, Sofija, Bulgaria, 2013a. Bălgarska akademija na naukite, Omnipress, Inc.

Michal Novák, Zdeněk Žabokrtský, and Anna Nedoluzhko. Two case studies on translating pronouns in a deep syntax framework. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 1037–1041, Nagoya, Japan, 2013b. Asian Federation of Natural Language Processing, Asian Federation of Natural Language Processing.

Václav Novák and Zdeněk Žabokrtský. Feature Engineering in Maximum Spanning Tree Dependency Parser. In Václav Matoušek and Pavel Mautner, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 92–98, Berlin / Heidelberg, 2007. Springer. ISBN 978-3-540-74627-0.

Filipe Nunes. Verbal Lemmatization and Featurization of Portuguese with Ambiguity Resolution in Context. Master's thesis, Universidade de Lisboa, Lisboa, Portugal, 2007. URL http://hdl.handle.net/10451/13851.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Stephan Oepen. The Transfer Formalism. General Purpose MRS Rewriting. Technical report, University of Oslo, 2008. URL http://www.emmtee.net/reports/11.pdf.

Barbara Plank and Gertjan van Noord. Dutch Dependency Parser Performance Across Domains. In Eline Westerhout, Thomas Markus, and Paola Monachesi, editors, *Computatuional Linguistics in the Netherlands 2010: selected papers from the 20th CLIN Meeting*. LOT, Netherlands Graduate School of Linguistics, 2010. URL https://odur.let.rug.nl/~vannoord/papers/bplank-vannoord-clin2010.pdf.

Martin Popel. *Ways to Improve the Quality of English-Czech Machine Translation*. Master's thesis, Charles University in Prague, 2009. URL http://ufal.mff.cuni.cz/~popel/papers/master_thesis.pdf.

Martin Popel, David Mareček, Nathan Green, and Zdeněk Žabokrtskỳ. Influence of parser choice on dependency-based MT. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 433–439. Association for Computational Linguistics, 2011. URL http://dl.acm.org/citation.cfm?id=2133019.

Robbert Prins and Gertjan van Noord. Reinforcing parser preferences through tagging. *Traitement Automatique des Langues*, 44(3):121–139, 2003.

Jan Ptáček and Zdeněk Žabokrtský. Synthesis of czech sentences from tectogrammatical trees. In Petr Sojka, Ivan Kopeček, and Karel Pala, editors, *Lecture Notes in Artificial Intelligence, Text, Speech and Dialogue. 9th International Conference, TSD 2006, Brno, Czech Republic, September 11–15, 2006, Proceedings*, volume 4188, pages 221–228, Berlin / Heidelberg, 2006. Masarykova univerzita, Springer. ISBN 978-3-540-39090-9. URL http://ufal.mff.cuni.cz/~zabokrtsky/papers/tsd06-synthesis.pdf.

J. Ptáček. Two Tectogrammatical Realizers Side by Side: Case of English and Czech. In *Fourth International Workshop on Human-Computer Conversation*, Bellagio, Italy, 2008.

Kashif Shah, Eleftherios Avramidis, Ergun Biçici, and Lucia Specia. QuEst: Design, Implementation and Extensions of a Framework for Machine Translation Quality Estimation. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 100:19–30, 2013. URL http://ufal.mff.cuni.cz/pbml/100/art-shah-avramidis-bicici-specia.pdf.

Michel Simard, Cyril Goutte, and Pierre Isabelle. Statistical phrase-based post-editing. In *Proceedings of The North American Chapter of the Association for Computational Linguistics Conference (NAACL-07)*, pages 508–515, Rochester, NY, April 2007.

Kiril Simov and Petya Osenova. A Hybrid System for MorphoSyntactic Disambiguation in Bulgarian. In *In: Proc. of the RANLP 2001 Conference, Tzigov chark*, pages 5–7, 2001.

Kiril Simov, Petya Osenova, and Milena Slavcheva. BTB:TR03: BulTreeBank morphosyntactic tagset BTB-TS version 2.0., 2004.

Kiril Simov, Iliana Simova, Ginka Ivanova, Maria Mateva, and Petya Osenova. A system for experiments with dependency parsers. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA). ISBN 978-2-9517408-8-4.

Drahomíra Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing 2007*, pages 67–74, Praha, Czechia, 2007. Univerzita Karlova v Praze, Association for Computational Linguistics. ISBN 978-1-932432-88-6.

Jana Straková, Milan Straka, and Jan Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-5003.

Gertjan van Noord. **A**t **L**ast **P**arsing **I**s **N**ow **O**perational. In *TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles*, pages 20–42, Leuven, 2006.

Gertjan van Noord. Learning efficient parsing. In *EACL 2009, The 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 817–825, Athens, Greece, 2009.

Zdeněk Žabokrtský. *From Treebanking to Machine Translation*. Habilitation thesis, Charles University in Prague, 2010. URL http://ufal.mff.cuni.cz/~zabokrtsky/publications/theses/hab-zz.pdf.

Rui Wang, Petya Osenova, and Kiril Simov. Linguistically-augmented Bulgarian-to-English Statistical Machine Translation Model. In *Proceedings of the Joint Workshop on Exploiting Synergies Between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra)*, EACL 2012, pages 119–128, Stroudsburg, PA, USA, 2012a. Association for Computational Linguistics. ISBN 978-1-937284-19-0. URL http://dl.acm.org/citation.cfm?id=2387956.2387972.

Rui Wang, Petya Osenova, and Kiril Simov. Linguistically-enriched Models for Bulgarian-to-English Machine Translation. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST-6 '12, pages 10–19, Stroudsburg, PA, USA, 2012b. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2392936.2392939.

Daniel Zeman. Reusable tagset conversion using tagset drivers. In *Proceedings of LREC*, pages 213–218, 2008.

Z. Žabokrtský, J. Ptáček, and P. Pajas. TectoMT: highly modular MT system with tectogrammatics used as transfer layer. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 167–170. Association for Computational Linguistics, 2008.

Zdeněk Žabokrtský and Martin Popel. Hidden Markov Tree Model in Dependency-based Machine Translation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 145–148, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1667583.1667628.

Appendix A QTLeap Manager

This Appendix aims at describing QTLeap Manager (`qtlm`), whichs was developed in the QTleap project and whose purpose is to make routine development work easier for developers working on TectoMT-based translation systems. It presents instructions on how to install `qtlm` (Section A.1), how to use it (Section A.2) and how to configure it (Section A.3).

Comments and suggestions for improvement are welcome (`luis.gomes@di.fc.ul.pt`).

A.1 Installation

Before starting, make sure that you have a working Treex installation. You can find instructions at the Treex web page.

The remainder of this installation guide assumes that you have checked-out the TectoMT repository into `$HOME/code/tectomt` as follows (adjust if necessary):

```
mkdir -p $HOME/code
url=https://svn.ms.mff.cuni.cz/svn/tectomt_devel/trunk
svn --username public co $url $HOME/code/tectomt
```

Pre-requisites/dependencies of `qtlm`:

- TectoMT (>= rev14386)
- Perl (>= v5.14.2)
- Python (>= 3.2.3)
- Bash (>= 4.2)
- Gawk (>= 3.1.8)
- GIZA++ (>= 1.0.7)

The `qtlm` package can be downloaded from `http://qtlm.sourceforge.net/` under a Creative Commons Attribution 3.0 Unported License. Extract the *QTLeap Manager* archive `qtlm_rev293.tgz` into `$HOME/code/qtlm`:

```
mkdir -p $HOME/code
tar xzf qtlm_rev273.tgz -C $HOME/code
```

Add the following ling to your `$HOME/.bashrc`:

```
source $HOME/code/qtlm/conf/env/default.sh
```

And run the previous command on your active terminal before proceeding.

A.2 Usage

If you type `qtlm help` on your terminal you shoud get the following usage summary:

```
Usage: qtlm <command> <args>...

List of commands:

  train
        Trains the transfer models for the current configuration.

  serve
        Starts two MTMonkey workers, one for each translation direction.

  evaluate <src> <trg> [testset]...
        Evaluates current pipeline using given testset or all configured
        testsets if a testset is not specified.

  list scores
        Lists BLEU scores from all evaluations in current directory.

  clean <src> <trg> [testset]...
        Cleans cache files from last evaluation. Use this if you changed
        the <src> languages analysis.

  save <testset> <description>
        Saves a snapshot of the current evaluation of <testset>.
        <description> should be a brief description of what changed since
        last save. Note that <testset> must have been evaluated in both
        translation directions before saving a snapshot.
        Snapshots are uploaded to the share server.

  list snapshots
        Lists all snapshots in reverse chronological order.

  compare [snapshot_id]
        Compare current evaluations with specified snapshot (or with
        latest snapshot if snapshot_id is not given).

  translate <src> <trg>
        Translates text from STDIN (expects one sentence per line) and
        writes translations to STDOUT.
        If environment variable $save_trees is set, then trees are saved
        into the directory specified by the variable.

  help
        Shows this help.

  version
        Shows qtleap script version.
```

Note: `save`, `list snapshots`, and `compare` are *experimental* features still under development.

Most of these commands require an environment variable `$QTLM_CONF` to be set. This variable should contain a string with three components separated by a forward slash (`/`):

1. the language pair (in the form of `L1-L2`);
2. the training dataset name;
3. the date when the transfer models were trained (formatted as `YYYY-MM-DD`)

Example: `QTLM_CONF=en-pt/ep/2015-02-12`

The two languages must be lexicographically ordered (`en-pt` is OK, `pt-en` is not). The same configuration identifier is used for both translation directions. According to the `$QTLM_CONF` variable defined above, the file `$QTLM_ROOT/conf/datasets/en-pt/ep.sh` must exist (see Dataset Configuration section below for further details). The date suffix (in this case `2015-02-12`) indicates when the transfer models were trained.

A.2.1 Training

Please see the relevant configuration for training Training transfer models (both translation directions are trained in parallel):

```
qtlm train
```

The training process will create a directory named

```
train_${DATASET}_${LANG1}-${LANG2}_${TRAIN_DATE}
```

which would be `train_ep_en-pt_2015-02-12` for the previous example. The training process will create several files and sub-directories within that directory. For example, when training models for English-Portuguese, we get the following files and directories:

```
.
`-- train_ep_en-pt_2015-02-12
    |-- [*] qtlm.info  # contains versioning information about qtlm
    |-- [*] qtlm.stat  # output of "hg stat" on $QTLM_ROOT repository
    |-- [*] qtlm.diff  # unified diff of the $QTLM_ROOT repository
    |-- [*] tectomt.info # contains versioning information about tectomt
    |-- [*] tectomt.stat # output of "svn stat" on the $TMT_ROOT repository
    |-- [*] tectomt.diff # unified diff of the $TMT_ROOT repository
    |-- dataset_files/   # downloaded from central share server
    |-- corpus/          # plain text split into chunks of 200 sentences
    |-- lemmas.gz        # GIZA input files
    |-- giza/            # GIZA itermediate files
    |-- alignments.gz    # GIZA final alignments
    |-- models/
    |   |-- en-pt/       # models for EN to PT transfer
    |   |   |-- formeme/
    |   |   |   |-- [*] maxent.model.gz
    |   |   |   `-- [*] static.model.gz
    |   |   |-- lemma/
    |   |   |   |-- [*] maxent.model.gz
    |   |   |   `-- [*] static.model.gz
```

```
|   |   `-- v/       # input vectors for machine learning
|   |-- pt-en/       # models for PT to EN transfer
|   |   |-- formeme/
|   |   |   |-- [*] maxent.model.gz
|   |   |   `-- [*] static.model.gz
|   |   |-- lemma/
|   |   |   |-- [*] maxent.model.gz
|   |   |   `-- [*] static.model.gz
|   |   `-- v/       # input vectors for machine learning
|-- logs/            # logs for all training stages/tools
|-- atrees/          # analytical-level trees
`-- ttrees/          # tectogrammatical-level trees
```

When training is finished, the files prefixed with `[*]` in the above tree are automatically uploaded to the share server into the directory `$upload_ssh_path/$QTLM_CONF`. See Sharing Configuration section for details about `$upload_ssh_path` and related variables.

A.2.2 Translation

Translating from English to Portuguese (reads one sentence per line from `STDIN` and writes one sentence per line on `STDOUT`):

```
qtlm translate en pt
```

If you want to save the trees of each translated sentence (for debugging purposes for example), then set the target directory in the environment variable `$save_trees`:

```
save_trees=somedir qtlm translate en pt
```

This will read from `STDIN` and write to `STDOUT` as previously, but it will also create a file named `somedir/######.treex.gz` for each input line (`######` is replaced by the number of the line, starting with `000001`).

A.2.3 MTMonkey XML-RPC workers

To start MTMonkey workers for the current configuration, just run:

```
qtlm serve
```

This will launch a pair of `treex-socket-servers` (one for each translation direction) and a pair of `treex-mtmworkers` which provide a XML-RPC interface to the (plain-text) socket servers. The ports of these 4 servers should be configured in the dataset configuration file. See Dataset Configuration section.

A.2.4 Evaluation

Evaluating the current pipeline on a specific evaluation set (in this example `qtleap_2a`):

```
qtlm evaluate en pt qtleap_2a
```

For this command to succeed the file `$QTLM_ROOT/conf/testsets/en-pt/qtleap_2a.sh` must exist and define a variable named `testset_files` as described below in Testset Configuration section.

A new directory `eval_qtleap_2a` will be created in the current directory with the following structure:

```
.
`-- eval_qtleap_2a
    |-- about.txt               # contains versioning information
    |-- qtleap_2a.en2pt.bleu    # output of `mteval-v13a.pl`
    |-- qtleap_2a.en2pt.cache.treex.gz # trees before synthesis stage
    |-- qtleap_2a.en2pt.final.treex.gz # final trees
    |-- qtleap_2a.en2pt.html    # original, reference and MT side by side
    |-- qtleap_2a.en2pt.ngrams  #
    |-- qtleap_2a.en2pt.resume  # output of Print::TranslationResume
    |-- qtleap_2a.en.txt        # original English text
    |-- qtleap_2a.pt_mt.txt     # machine translated (English to Portuguese)
    `-- qtleap_2a.pt.txt        # original Portuguese text
```

If you then evaluate on the other direction (Portuguese to English):

```
qtlm evaluate pt en qtleap_2a
```

The following files will be added to the directory:

```
.
`-- eval_qtleap_2a
    ...
    |-- qtleap_2a.en_mt.txt     # machine translated (Portuguese to English)
    |-- qtleap_2a.pt2en.bleu    # output of `mteval-v13a.pl`
    |-- qtleap_2a.pt2en.cache.treex.gz # trees before synthesis stage
    |-- qtleap_2a.pt2en.final.treex.gz # final trees
    |-- qtleap_2a.pt2en.html    # original, reference and MT side by side
    |-- qtleap_2a.pt2en.ngrams  #
    `-- qtleap_2a.pt2en.resume  # output of Print::TranslationResume
```

To evaluate the current pipeline on all evaluation sets listed in `$QTLM_ROOT/conf/testsets/en-pt` just omit the evalset name:

```
qtlm evaluate en pt
```

To list BLEU and NIST scores for all testsets evaluated under the current directory:

```
qtlm list scores
```

Which will output something like:

```
TESTSET          NIST    BLEU    SYSTEM
qtleap_2a.en2pt  5.4622  0.1942  qtleap:en-pt/ep/2015-01-19
qtleap_1a.en2pt  5.7766  0.2290  qtleap:en-pt/ep/2015-01-19
qtleap_2q.en2pt  4.8243  0.1419  qtleap:en-pt/ep/2015-01-19
qtleap_1q.en2pt  4.5370  0.1224  qtleap:en-pt/ep/2015-01-19
```

Cleaning cached intermediate trees If you are developing the synthesis and you want to re-evaluate the pipeline, you just repeat the above commands to re-synthesize the translations.

The re-runs will be much faster than the first evaluation because `qtlm evaluate` will reuse the previously created `*.cache.treex.gz` files (which contain the trees after analysis and transfer), and only the synthesis step is done.

However, if you have changed the analysis or transfer steps, then you should remove the cached trees by running:

```
qtlm clean
```

This will clean the cached trees for all configured testsets that have been already evaluated in the current directory.

A.2.5 Snapshots

Note: snapshots are under development.

A snapshot is a bundle of current evaluations together with all information needed to recover the exact state of the current pipeline.

Creating a snapshot To create a snapshot first you must ensure that all configured testsets have been evaluated using the current `$QTLM_CONF` for both translation directions. Then you may run:

```
qtlm save "brief description of what changed since last snapshot"
```

This command will create a new directory `snapshots/YYYY-MM-DDL` (year, month, day, and a letter) within the current directory and it will copy all current evaluations into it.

The value of the `$QTLM_CONF` variable is saved into `about.txt` within the snapshot directory, as well as the current mercurial and SVN revision numbers of `$QTLM_ROOT` and `$TMT_ROOT` respectively, and the current revision of the remote lxsuite service.

Furthermore, uncommited changes to the `$QTLM_ROOT` and `$TMT_ROOT` repositories are also saved in the form of a unified diff (`qtlm.diff` and `tectomt.diff`), allowing us to recover the current source code in full extent.

WARNING: only files already tracked by mercurial and SVN will be included in the unified diff of every snapshot, i.e. all files appearing with a question mark when you issue the commands `hg status` or `svn status` *WILL NOT* be included in the diff.

The snapshot is also uploaded to the configured share server, making it readily available for comparison and analysis to other users. The URL of a snapshot is

`$download_http_base_url/snapshots/LANGPAIR/DATASET/YYYY-MM-DDL`

where `$download_http_base_url` is a configuration variable described in Sharing Configuration, and `LANGPAIR` and `DATASET` are the first two components of `$QTLM_CONF`.

Listing snapshots Listing all saved snapshots, from the most recent to the oldest:

```
qtlm list snapshots
```

This will fetch an updated list of snapshots from the share server for the current `$QTLM_CONF`. The list is presented as follows:

```
------------------------------------------------------------------------
 Snapshot      | en2pt | pt2en | Description
---------------|-------|-------|---------------------------------------
 * 2015-02-09a | 12.81 |  6.27 | added some exceptions to the rules
   2015-02-02a |  9.56 |  4.69 | some reordering rules for noun phrases
------------------------------------------------------------------------
```

Columns `en2pt` and `pt2en` show the average BLEU scores over all configured evalsets for both translation directions. Snapshots marked with an asterisk (`*`) exist both locally and on the server. Unmarked snapshots exist only on the server.

Comparing snapshots To compare current translations/evaluations with the ones from last snapshot:

```
qtlm compare
```

To compare current translations/evaluations with a specific snapshot (in this case 2015-01-20):

```
qtlm compare 2015-01-20
```

Note: if the specified snapshot does not exist locally (i.e. it does not appear marked with an asterisk in the list of snapshots), then the comparison will take longer because the snapshot will be automatically downloaded from the server.

A.3 Configuration

All configuration files are kept in directory `$QTLM_ROOT/conf`.

A.3.1 Environment Configuration

The shell environment is configured by adding the following line to your `$HOME/.bashrc`:

```
source $HOME/code/qtlm/conf/env/default.sh
```

This file defines and exports the following variables: `QTLM_ROOT`, `TMT_ROOT`, `TREEX_CONFIG`, `PATH`, and `PERL5LIB`. If you installed the qtlm and tectomt repositories into the recommended place (`$HOME/code/qtlm` and `$HOME/code/tectomt`) then you don't have to change this file. Else, you should create a file with your username and source it from your `$HOME/.bashrc` like this:

```
source $QTLM_ROOT/conf/env/$USER.sh
```

A.3.2 Host Configuration

The file `$QTLM_ROOT/conf/hosts/$(hostname).sh` will be used if it exists, else the file `$QTLM_ROOT/conf/hosts/default.sh` is used instead. Either of these files must define the following variables:

$num_procs The maximum number of concurrent processes that should be executed. Specify a number lower than the number of available processors in your machine.
(default: `2`)

$sort_mem How much memory can we use for sorting?
(default: 50%)

$big_machine Set this to `true` only if your machine has enough memory to run several concurrent analysis pipelines (for example a machine with 32 cores and 256 GB RAM).
(default: `false`)

$giza_dir Where GIZA++ has been installed.
(default: `"$TMT_ROOT/share/installed_tools/giza"`)

A.3.3 Sharing Configuration

Corpora and transfer models are downloaded/uploaded automatically, without user intervention. All data is stored in a central server, configured in `$QTLM_ROOT/conf/sharing.sh`:

$upload_ssh_* These variables configure SSH access for automatic uploading of transfer models after training. Example:

```
upload_ssh_user="lgomes"
upload_ssh_host="nlx-server.di.fc.ul.pt"
upload_ssh_port=22
upload_ssh_path="public_html/qtleap/share"
```

$download_http_* These variables configure HTTP access for automatic downloading of datasets, testsets, and transfer models as needed. Example:

```
download_http_base_url="http://nlx-server.di.fc.ul.pt/~lgomes/qtleap/share"
download_http_user="qtleap"
download_http_password="paeltqtleap"
```

A.3.4 Dataset Configuration

A dataset is a combination of parallel corpora that is used to train the transfer models. For each `DATASET` we must create a respective file `$QTLM_ROOT/conf/datasets/L1-L2/DATASET.sh` and it must define the following variables:

$dataset_files A space-separated list of files (may be gzipped), each containing tab-separated pairs of human translated sentences. The file paths specfied here must be relative to `$download_base_url` configured in `$QTLM_ROOT/conf/sharing.sh`.

Example: `dataset_files="corpora/europarl/ep.enpt.gz"`

$train_hostname The hostname of the machine where the transfer models are to be trained. This must be the exact string returned by the hostname command. It is used as a safety guard to prevent training on an under-resourced machine. You may use an `*` to allow training of this dataset on any machine.

$*_train_opts Four variables set the options affecting the behaviour of the machine learning algorithms for training each transfer model:

- $lemma_static_train_opts
- $lemma_maxent_train_opts
- $formeme_static_train_opts
- $formeme_maxent_train_opts

Refer to `$TMT_ROOT/treex/training/mt/transl_models/train.pl` for further details. Example:

```
static_train_opts="--instances 10000 \
    --min_instances 2 \
    --min_per_class 1 \
    --class_coverage 1"

maxent_train_opts="--instances 10000 \
    --min_instances 10 \
    --min_per_class 2 \
    --class_coverage 1 \
    --feature_column 2 \
    --feature_cut 2 \
    --learner_params 'smooth_sigma 0.99'"

lemma_static_train_opts="$static_train_opts"
formeme_static_train_opts="$static_train_opts"

lemma_maxent_train_opts="$maxent_train_opts"
formeme_maxent_train_opts="$maxent_train_opts"
```

$rm_giza_files If `true` then GIZA models are removed after the aligment is produced.

$treex_socket_server_ports This variable defines the two ports of Treex socket servers (one for each translation direction).

Example:

```
treex_socket_server_ports="7001 7002"
```

$treex_mtmworker_ports This variable defines the two ports of Treex MTMonkey XML-RPC servers (one for each translation direction).

Example:

```
treex_mtmworker_ports="8001 8002"
```

A.3.5 Testset Configuration

A testset is a combination of parallel corpora that is used to test the whole pipeline. For each `TESTSET` we must create a respective file `$QTLM_ROOT/conf/testsets/L1-L2/TESTSET.sh` and it must define the following variables:

$testset_files A space-separated list of files (may be gzipped), each containing tab-separated pairs of human translated sentences. The file paths specified here must be relative to `$download_base_url` configured in `$QTLM_ROOT/conf/sharing.sh`.

Example: `testset_files="corpora/qtleap/qtleap_1a.gz"`

A.3.6 Treex Configuration

Treex configuration for each user is kept in `$QTLM_ROOT/conf/treex/$USER/config.yaml`. If you wonder why we don't simply use `$QTLM_ROOT/conf/treex/$USER.yaml`, it is because Treex expects its configuration file to be named exactly `config.yaml`.

Here's a Treex configuration (`$QTLM_ROOT/conf/treex/luis/config.yaml`) for guidance:

```
---
resource_path:
  - /home/luis/code/tectomt/share
share_dir: /home/luis/code/tectomt/share
share_url: http://ufallab.ms.mff.cuni.cz/tectomt/share
tmp_dir: /tmp
pml_schema_dir: /.../tectomt/treex/lib/Treex/Core/share/tred_extension/treex/resources
tred_dir: /home/luis/tred
tred_extension_dir: /.../tectomt/treex/lib/Treex/Core/share/tred_extension
```