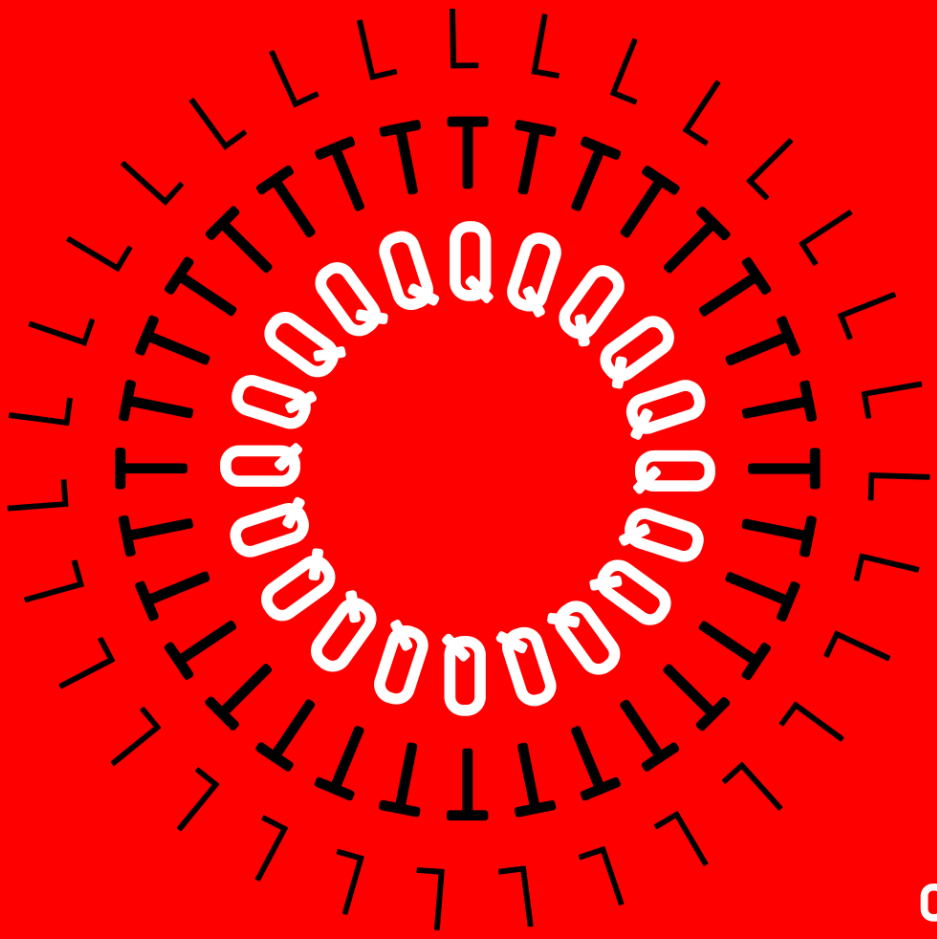


qtleap
quality translation by deep language engineering approaches

Report on MT improved with robust deep processing

DELIVERABLE D4.13

VERSION 2.0 | 2016-12-15

QTLeap

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

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

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

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

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

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

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

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

www.qtleap.eu

Funded by

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

Supported by

And supported by the participating institutions:

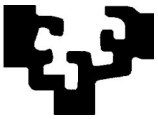
Faculty of Sciences, University of Lisbon

German Research Centre for Artificial Intelligence

Charles University in Prague

Bulgarian Academy of Sciences

Humboldt University of Berlin



University of Basque Country

University of Groningen

Higher Functions, Lda

Revision history

Version	Date	Authors	Organisation	Description
1.0	Sep 30, 2016	Petya Osenova	IICT-BAS	First draft
1.1	Oct 26, 2016	Gertjan van Noord, Dieke Oele	UG	Contribution
1.2	Oct 24, 2016	Will Roberts	UBER	Contribution
1.3	Oct 28, 2016	João Silva	FCUL	Contribution
1.4	Dec 9, 2016	Kiril Simov	IICT-BAS	Contribution
2.0	Dec 14, 2016	Markus Egg	UBER	Review comments incorporated

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 MT improved with robust deep processing

DOCUMENT QTLEAP-2016-D4.13
EC FP7 PROJECT #610516

DELIVERABLE D4.13

completition
FINAL
status
SUBMITTED
dissemination level
PUBLIC

responsible
Kiril Simov (WP 4 Coordinator)
reviewer
Markus Egg
contributing partners
FCUL, DFKI, IICT-BAS, UBER, UPV-EHU, UG

authors
Petya Osenova, Kiril Simov, João Silva, Dieke Oele, Gertjan van Noord, Will Roberts

Contents

List of Abbreviations

CoNLL	Conference on Natural Language Learning
DPC	Dutch Parallel Corpus
ERG	English Resource Grammar
HPSG	Head-driven Phrase Structure Grammar
LRTs	Language Resources and Tools
OOV	Out-Of-Vocabulary item
MT	Machine Translation
MRS	Minimal Recursion Semantics
MWE	Multiword Expressions
NLP	Natural Language Processing
POS	Part-Of-Speech
PDT	Prague Dependency Treebank
RMRS	Robust Minimal Recursion Semantics
SRL	Semantic Role Labeling

1 Introduction

Deliverable D4.12 reports on Machine Translation (MT) improved with robust deep processing::

- Out-of-Vocabulary Item (OOVs) components;
- Multiword Expressions(MWEs) components;
- Deep Semantic Transfer Module, and
- Hybrid Generation Module.

The various components and modules cover different language pairs. The OOVs components are applied in the analysis of the source language text, during the generation of the target sentences, and after the translation as post-processing. The post-processing refers to the transliteration of names between Bulgarian and English.

The Deep Semantic Transfer Module implements a hybrid architecture in which the backbone is statistical machine translation and deep linguistic knowledge is transferred from the source language sentences to the target language ones via the produced word alignments. The transferred deep knowledge is used for post-processing of the output from the statistical machine translation.

The Hybrid Generation Module exploits a model of lexical choice to improve the deep MT. The model is concerned with the selection of an appropriate lemma from a WordNet synset.

The discussed results are from Pilot 3 settings, described in D2.10 and D2.11.

The structure of the Deliverable is as follows. Section 2 presents the treatment of OOVs for deep MT, Section 3 describes experiments with MWEs analysis and generation for MT. Section 4 defines a hybrid architecture for deep semantic transfer. Section 5 reports on model for lexical choice in generation for MT. The last section concludes the deliverable.

2 MT enhanced with OOV handling

Out of vocabulary items (OOVs) are words or terms, which are not recognised by a MT system. OOVs typically manifest as untranslated words that appear in the output of TectoMT. We distinguish two kinds of OOVs:

1. Words or word forms which are unknown to a particular language analysis pipeline (e.g. parser). These are more of an issue in rule-based systems; in statistical systems, they can be handled by robust statistical parsers, or, in deep grammar analysis pipelines, by supertaggers.

2. Lexemes unknown to the translation model.

Other OOV-like categories include forms which should not be translated (named entities and "fixed" entities). For some languages like Bulgarian many foreign names (Barak Obama, Berlin, Shengen, etc.) have to be transliterated using Cyrillic letters. Similarly, Bulgarian names needs to be transliterated using Latin alphabet when translating to English. The problem is especially important for the direction from Bulgarian to English where names in Cyrillic alphabet are unreadable.

We address these types of OOVs in this section.

2.1 Robust deep processing

It is important to ensure that the analysis stage in a MT pipeline produces a high quality result since errors in the analysis will propagate to the later stages. Deep processing grammars can provide this, since they produce complex and highly precise analyses of their input sentences, including a formal logic semantic representation of their meaning.

Central to most such grammars is a very rich lexicon, where each word or expression is associated with a variety of linguistic information, ranging from simple morphosyntactic category to subcategorization frames and beyond. As a consequence, if an expression is missing from the lexicon, a full analysis may not be produced.

The work of Silva [2014] investigated ways of improving the robustness of deep processing to OOV expressions, culminating in the development of a novel approach where a classifier is run prior to the deep grammar to assign rich annotation to words, thus enabling the grammar to carry on with the analysis even when faced with OOV expressions.

The approach that was finally developed places a machine-learning classifier as a bridge between the input pre-processing stage and the grammar itself. This classifier takes features from the input pre-processing stage and assigns a tag—known as a deep lexical type—to words. The classifier makes use of kernel methods to enable it to seamlessly use linguistic structures as features.

A large part of the linguistic information encoded in a deep lexical type of an expression pertains to its subcategorization frame. Existing supertaggers make use only of shallow features that are based on a limited window of context, but these features are not suited for capturing such dependencies, since they often occur over long distances. By making use of structured linguistic information in the form of grammatical dependency graphs, the classifier of Silva [2014] can better handle these harder cases.

The classifier was tested on the HPSG framework, but it can be applied to any computational grammar framework where the lexical information can be encoded as a tag to be assigned. When tested over English, with the ERG grammar [Flickinger, 2000] and the larger Redwoods dataset [Oepen et al., 2004], showed that the classifier was able to consistently perform better than the supertagger [Silva, 2014].

2.2 OOVs in the translation model

Generally, lexemes unknown to the translation model can be dealt with in the limit by increasing the quantity of parallel training data available to the TectoMT system.

Morphological analysis and generation, supported by the TectoMT system, can allow the interpretation and production of forms of lexemes unseen in training data. Similarly, the incorporation of word sense disambiguation (see Deliverable D5.7, Section 2; and the description of Experiment 5.4.2 in Deliverable D5.11) is able to recognise unknown words, as long as they are synonyms of other words seen in the training data, and group these into conceptually coherent categories (WordNet synsets); using Hidden Markov Tree Models [Oele and van Noord, 2015] (see Section 5 below), TectoMT can also translate source language into target concepts, and then produce words unseen in the training corpus. Therefore, the problem of unknown vocabulary items should ideally be addressed at the level of lexemes or concepts.

Work on the TectoMT system has focussed on addressing the problem of OOVs using bilingual terminologies relevant to the application domain (see description of gazetteers in Deliverable D5.7, Section 4.2). Bilingual terminologies can also be incorporated as

interpolated translation models (D5.7, Section 4.3). See also the description of Experiment 5.4.4 in Deliverable D5.11.

2.3 Transliteration of Names from Bulgarian to English

The transliteration of proper names presents many and quite challenging difficulties not only to automatic systems, but also to human translators. A lot of information is needed to perform the task: the language of origin to determine the pronunciation; some real world facts like how this name was/is actually pronounced by the person it belongs or belonged to (in case of personal names) or by the locals (in case of toponyms) and so on; and also the tradition in transliterating names from this particular language into the given target language. In the case of Statistical Machine Translation names are problematic in two frequent cases: when they coincide with common words, and when they do not appear in the training corpus.

We have performed several experiments reported in Todorova and Simov [2015]. The idea is to train a statistical machine translation model on parallel data of names and their corresponding transliterations.

The parallel lists of names on which we have trained our models have been extracted from DBpedia. We have used the instance type feature to select the URLs of all people, places and organizations. For the module used in the experiment reported here we have extended this initial list with new pairs collected from other sources. In the end, for Bulgarian-English we had at our disposal more than 65 000 pairs. We have trained several machine translation Moses[1] models. The language models have been trained on the Bulgarian part of the English – Bulgarian parallel list of names. The names were represented as unigram models and they look like this:

$$E\ l\ v\ i\ s$$

and each individual name was separated; for example, *Elvis Presley* was represented as *Elvis* and *Presley*.

Language pair	Pilot3 without Transliteration	Pilot3 with Transliteration
bg→en	15.06	15.44

Table 1: BLEU on translation from Bulgarian into English on the QTLeap News Corpus.

The trained models were applied to the Named Entities that in the translation are left as they appear in the source text. In Table 1 BLEU scores for Pilot3 on translation of the QTLeap News Corpus with and without transliteration module are presented.

3 MT enhanced with MWE

Deliverable D5.7, Section 6.3 reported on a series of experiments on multiword expressions conducted on the English-Spanish language pair. That work analysed non-compositional multiword expressions in English (the so-called *analysis-only* condition). Section 3.1 reports work to replicate these experiments using a different QTLeap language, Basque

[1] http://www.statmt.org/moses/

(section 3.1.1), and also on the same language pair, in the reverse direction (Spanish-English, section 3.1.2). Furthermore, section 3.2 reports on an extension of this experimental paradigm, whereby multiword analysis is conducted in both source and target languages (the so-called *generation* condition).

3.1 Replication of analysis-only MWE experiments

3.1.1 English-Basque

We have conducted work to replicate the MWE experiments done on the English-Spanish language pair, using a different QTLeap language, Basque. The Basque models are trained using the Elhuyar training corpus (1.3M sentences, 15.8M EN words).

We have compared the effect of our MWE analysis machinery to the pre-existing QTLeap gazetteer module (which we have disabled to date).

In an effort to improve the quality of our trained models, we also built a larger training corpus (here named Elhuyar+), on the basis of the Elhuyar training corpus, by adding the GNOME, KDE4, and Ubuntu parallel corpora from the OPUS Project[2] (Tiedemann [2012]; these give a total of 832K sentences, 4.5M EN words).

Results are summarised in table 2.

			Training MWEs		Test MWEs	
Condition	BLEU	NIST	Tokens	Types	Tokens	Types
Pilot 2	17.68	4.9187				
Elhuyar Baseline, no MWEs	16.46	4.8048				
Elhuyar MWEs, $\theta = 0.1$	16.52 ***	4.8073	5153	977	0	0
Elhuyar MWEs, $\theta = 0.2$	**16.58** **	4.8153	23933	4073	4	3
Elhuyar MWEs, $\theta = 0.3$	16.47	**4.8159**	94429	14457	27	11
Pilot 2 (+ gazetteers)	19.78	**5.3203**				
Elhuyar Baseline (+ gazetteers)	18.30	5.1805				
Elhuyar, $\theta = 0.1$ (+ gazetteers)	18.35 **	5.1822	5153	977	0	0
Elhuyar, $\theta = 0.2$ (+ gazetteers)	**18.42** ***	5.1871	23933	4073	4	3
Elhuyar+ Baseline	16.37	4.8166				
Elhuyar+, $\theta = 0.1$	**16.49** ***	**4.8223**	6251	1083	0	0
Elhuyar+, $\theta = 0.2$	16.42	4.8175	32487	4350	4	3

Table 2: Summary of English-Basque results on QTLeap test set (`batch2a`). Statistical significance with respect to the relevant baseline: ** $p < 0.01$, *** $p < 0.001$.

On `batch2a`, we replicate the effect seen in the English-Spanish results, showing a statistically significant improvement over the baseline (+0.12 BLEU) with a small value for the MWE compositionality threshold ($\theta = 0.2$), although the effect is comparatively small.

Gazetteers seem to be complementary to the MWE analysis code. The gazetteer module delivers improvements across the board (about +2.0 BLEU on `batch2a`). Regarding the MWE experiments, we observe the same pattern of results with gazetteers turned on as with them turned off; further, the +0.12 BLEU improvement seen with MWE analysis

[2] http://opus.lingfil.uu.se/

on batch2a is preserved even when gazetteers are turned on. This suggests that gazetteers do not interfere (or even interact) with the MWE analysis, as we had previously assumed they would.

The larger Elhuyar+ corpus did not deliver the quality improvements we had hoped for. Despite increasing the corpus size by ca. 33% of in-domain text, the Elhuyar+ models do not train a significantly larger number of MWE types (although the number of MWE tokens seen during training does increase). At test time, the numbers of MWEs found are identical to the numbers from the models trained with just the Elhuyar corpus. Further, the Elhuyar+ models seem to be of lower quality (cf. almost -0.1 BLEU on batch2a).

As in the Spanish-English experiments, improvements are often seen even in cases where the trained translation model does not find any MWEs in the test set (e.g., here where $\theta = 0.1$). This suggests that analysing MWEs in the training corpus improves the quality of the trained translation model, independently of whether any MWEs are actually encountered when the model is used.

3.1.2 Spanish-English

We have replicated the multiword expression experiments done on the English-Spanish language pair in the other direction, Spanish-English. Results of evaluation on the QTLeap test set are summarised in table 3. As in the English-Spanish experiments, we obtained higher performance by training on the smaller in-domain corpus than by using Europarl. All of our results perform more poorly than the baseline system; however, in most cases, the decrease in performance is not statistically significant. These experiments are the first to attempt analysis of multiword expressions in a language other than English (note that, in these experiments, the source language is Spanish). This could indicate that the Spanish list of multiword expressions is of lower quality than the English list. We note that the number of multiword expression types is lower in both training and testing in Spanish than in English. The test corpus for these Spanish-English experiments differs considerably from the corpus used to test the English-Spanish models. In particular, the Spanish-English corpus consists of questions, whereas the English-Spanish corpus consists of answers. Questions are significantly shorter than answers, and make use of less detailed language. This may be a reason for the poor performance of the multiword expression analysis code on this task. For the QTLeap application, the quality of question translation is of lesser importance than the quality of answer translation. For this reason, it may be more useful to perform multiword expression analysis on the English$\rightarrow X$ translation pipelines only.

These experiments were also used to test the marginal effect of manually filtering the list of Spanish MWEs (work conducted at San Sebastian). Under this scheme, MWEs were removed from the list if they contained numbers, or contained certain tokens (*and, but, if, that, when, while, because, also, other*, and various other determiners, prepositions and verbs). This filtering removed some 36,565 MWEs, leaving the filtered list containing 119,066 expressions. The filtered lists are represented in the Table with the suffix *v2*. The results obtained here suggest that such filtering is ineffective at increasing the quality of the learned translation models.

3.2 Experiments on generation of multiword expressions

In this section, we describe the implementation of infrastructure to reversibly collapse treelets representing multiword expressions in a parallel training corpus, and evaluate

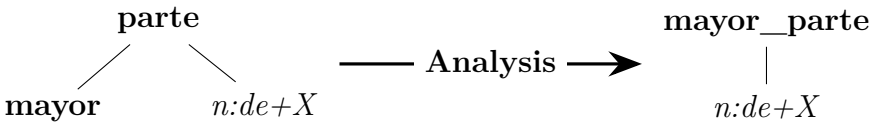
			Training MWEs		Test MWEs	
	BLEU	NIST	Tokens	Types	Tokens	Types
Pilot 2	24.86	6.1873				
Europarl Baseline, no MWEs	23.00	5.9624				
Europarl MWEs v1, $\theta = 0.1$	23.02	5.9584	146278	766	0	0
Europarl MWEs v1, $\theta = 0.2$	**23.02**	**5.9607**	444553	3053	0	0
Europarl MWEs v2, $\theta = 0.1$	23.00	5.9597	124596	549	0	0
Europarl MWEs v2, $\theta = 0.2$	**23.02**	5.9613	369096	2282	0	0
Europarl MWEs v2, $\theta = 0.3$	23.00	**5.9626**	1115019	7921	1	1
`indomain` Baseline, no MWEs	25.47	6.3516				
`indomain` MWEs v1, $\theta = 0.1$	**25.47**	**6.3515**	17517	672	0	0
`indomain` MWEs v1, $\theta = 0.2$	25.45	6.3482	44273	2603	2	1
`indomain` MWEs v1, $\theta = 0.3$	25.41	6.3470	117404	8523	2	1
`indomain` MWEs v2, $\theta = 0.1$	25.42	6.3466	12478	518	0	0
`indomain` MWEs v2, $\theta = 0.2$	**25.44**	**6.3482**	34229	2044	2	1
`indomain` MWEs v2, $\theta = 0.3$	25.41 **	6.3454	91577	6942	2	1

Table 3: Summary of Spanish-English results on QTLeap test set (`batch2q`). Statistical significance with respect to the relevant baseline: ** $p < 0.01$.

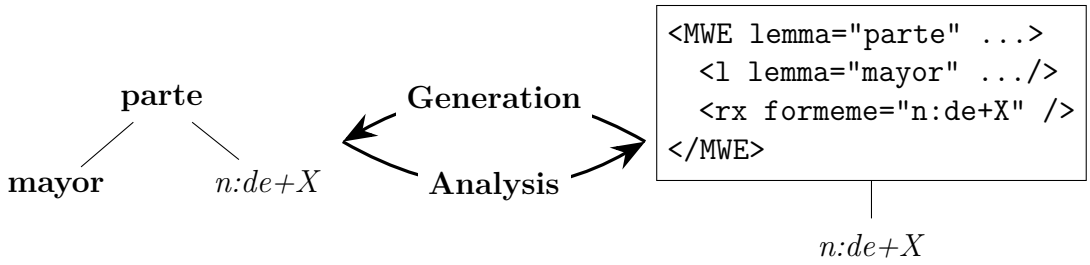
this on the English-Spanish language pair. As previously discussed in Deliverable D5.7, the analysis of multiword expressions in the source language only reflects the strong assumption that a non-compositional English multiword expression can be translated by a single Spanish lexeme. In cases where this assumption does not hold, multiword expression analysis will hurt the performance of the TectoMT system. Further, the analysis-only paradigm cannot help with cases where an English lexeme should be translated by a Spanish multiword expression, or cases where an English multiword expression is best translated by a Spanish multiword expression.

A straightforward extension of the analysis-only paradigm performs analysis of multiword expressions in both the source and target languages. This allows the translation model to learn to translate not only multiword expressions to single lexemes, but also single lexemes to multiword expressions, as well as multiword expressions to multiword expressions. Furthermore, such a system should exhibit less dependence on the value of the *compositionality threshold* θ: in cases where a compositional expression is accidentally collapsed in English, the system has the capacity to translate this into a compositional expression in Spanish, and then re-expand this into a treelet. Thus, a multiword analysis module with the ability to generate treelets permits some degree of *non-isomorphic transfer*, a feature which was not previously available in the TectoMT architecture.

A prerequisite for this model is the capability to represent a multiword expression as a complex structure with morphosyntactic and semantics attributes. At test time, these representations of multiword expressions must be expanded back into t-tree nodes. For this reason, we describe the functionality introduced in this section as *reversible collapsing*. The code used to implement this follows the design of the mechanism used in the analysis-only experiments, except that, instead of representing a multiword expression as a word-with-spaces, it builds a complex structural description.



(a) Analysis-only collapsing of multiword expression treelets.



(b) Reversible collapsing of multiword expression treelets.

Figure 1: Comparison of multiword expression collapsing methods on the phrase *mayor parte de* (*the majority of*).

The treelet representation is encoded in an XML format. This format records all relevant *t*-node properties (e.g., lemma, formeme, sempos (semantic part-of speech), etc.). The XML also encodes treelet structure: child nodes in the *t*-tree are represented as child tags of the XML representation. The encoding also represents children that are not part of the multiword expression. This provision is made to account for optional arguments to a multiword expression. Figure 1b diagrams how treelets can be collapsed to, and expanded from, this representation. The diagram includes an argument (here indicated only with its formeme, *n:de+X*).

The generation-enabled multiword expression experiments perform multiword collapsing on the source language side, exactly as in the analysis-only experiments. The system additionally performs reversible collapsing on the target language side, using the described XML format. This results in a trained translation model that is able to translate both single lexemes as well as multiword expressions from the source language into the target language. The translation model is also able to learn to produce treelet representations of multiword expressions in the target language. At test time, these compressed representations are detected and expanded. The expansion mechanism keeps track of arguments and adjuncts that are children of the collapsed multiword expression node, and moves these into their correct positions inside the the expanded treelet, insofar as possible, according to the configuration recorded in the XML format. Unexpected *t*-nodes are left in place, to allow for unforeseen modification. As before, our experiments are parameterised by an independent variable, θ, which represents an upper threshold on the estimated compositionality of the multiword expression. The experiments we report here use the same value of θ for both source and target language analysis.

We present experiments here using the in-domain training corpus, consisting of about 25 million words and 1.25 million sentences from the sentence-aligned parallel text in the "KDE" and "OpenOffice" files, and half of the "commoncrawl" file. This material is predominantly technical in nature, and prior experiments have demonstrated it to have a better domain overlap with the QTLeap test set than Europarl.

BLEU score results are shown in table 4, along with the counts of multiwords found during training and testing. Results from the analysis-only experiments are also included

			Training MWEs				Testing MWEs			
			EN		ES		EN		ES	
Description	BLEU	NIST	Toks	Types	Toks	Types	Toks	Types	Toks	Types
Baseline	26.00	6.8665								
MWEs analysis-only										
$\theta = 0.1$	**26.46** ***	**6.8992**	4593	837			0	0		
$\theta = 0.2$	26.43 **	6.8985	19586	3576			2	1		
$\theta = 0.3$	26.30 **	6.8918	67709	12333			18	7		
$\theta = 0.4$	26.10	6.8684	160828	32126			234	138		
$\theta = 0.5$	25.84	6.8363	303724	61657			480	293		
MWEs generation										
$\theta = 0.1$	26.09	6.8512	4593	837	4304	468	0	0	0	0
$\theta = 0.2$	26.47 ***	**6.9119**	19586	3576	20468	1866	4	3	0	0
$\theta = 0.3$	26.47 ***	6.9041	67709	12333	61257	6235	27	11	2	2
$\theta = 0.4$	**26.48** **	6.9031	160828	32126	142281	16213	39	21	23	10
$\theta = 0.5$	25.81	6.8182	303724	61657	260883	30722	107	41	17	10

Table 4: Results of testing generation models on the QTLeap test set (`batch2a`), trained on the `indomain` corpus. Statistical significance with respect to the baseline: ** $p < 0.01$, *** $p < 0.001$.

for comparison. The results demonstrate that the generation system gives slightly better performance than the analysis-only system. Furthermore, using the generation model, we observe more robust performance with increasing values of θ, supporting our hypothesis that a generation model should be tolerant to a greater range of values for θ than an analysis-only system.

4 MT enhanced with Deep Semantic Transfer

In Deliverable D4.1 we have reported on a list of phenomena considered as deep semantics. From this list in the implementation of Deep Semantic Transfer we considered

- Logical Form (LF), and
- Lexical Semantics (WSD).

As it was presented in D4.1, one of the deep semantic transfer methods for Machine Translation is using Minimal Recursion Semantics (Copestake et al. [2005]). The main idea was that an underspecified semantic representation is appropriate for machine translation because it provides an abstract level to semantic transfer while at the same time postponing difficult decisions. These difficulties are assumed to be less important in the area of machine translation.

Here we use Robust Minimal Recursion Semantics (RMRS). RMRS is introduced as a modification of MRS which captures the semantics resulting from a shallow analysis — see Copestake [2003] and Copestake [2007]. The main motivation for this development is the observation that currently no single system can do everything: both deep and shallow processing have inherent strengths and weaknesses. Consequently, the domain-dependent and domain-independent processing must be linked. Therefore, Copestake [2003, 2007] proposes a semantic representation which allows to build a uniform semantic representation for both deep and shallow processing. The justification behind RMRS is to

allow more underspecification in MRS. This is done by e.g. the separation of arguments from the predicates. Thus each predicate is represented via its name (constructed on the basis of the lemma of the word form in the text) and its main argument which depends on the part of speech — *referential index* for nouns and some pronouns or *event index* in other cases. In this way it is possible that the predicates and their arguments are added to the structure separately from each other. Here we present a formal definition of RMRS as defined in Jakob et al. [2010]. An RMRS structure is a quadruple

$$< hook, EPbag, argumentset, handleconstraints >$$

where a hook consists of three elements $l : a : i$, l is a label, a is an anchor and i is an index. Each elementary predication (members of $EPbag$) is additionally marked with an anchor[3] — $l : a : r(i)$, where l is a label, a is an anchor and $r(i)$ is a relation with one argument of appropriate kind — referential index or event index. The argument set contains argument statements of the following kind $a : ARG(x)$, where a is anchor which determines for which relation the argument is defined, ARG is the name of the argument, and x is an index or a hole variable or handle (h) for scopal predicates. The handle constraints are of the form $h =_q l$, where h is a handle, l is a label and $=_q$ is the relation expressing the constraint similarly to MRS. $=_q$ sometimes is written as qeq. This representation is appropriate for our goal because the important linguistic knowledge is connected to the tokens in the sentences. We could think about this linguistic knowledge as represented with the help of unary and binary relations. The unary relations represent the grammatical features, elementary predicates, lemmas, the main arguments of the elementary predicates, etc. of tokens in the sentence. The binary relations relate the anchors of arguments of the elementary predicates and the anchor of the elementary predicate — in our case both represented as tokens in the sentence.

MRS was applied in the past in at least two ways to support machine translation:

- rule-based semantic transfer, and
- factor-based statistical machine translation.

The rule-based semantic transfer uses transfer rules working on the MRS representation of the source language MRS structures and constructing the target language MRS structure. Thus, the transfer rules in this framework are rewriting rules over MRS (Minimal Recursion Semantics) structures. The basic format of the transfer rules is:

$$[\mathcal{C} :]\mathcal{I}[!\mathcal{F}] \rightarrow \mathcal{O}$$

where $\mathcal{I}$ is the *input* of the rule, $\mathcal{O}$ is the *output*. $\mathcal{C}$ determines the *context* and $\mathcal{F}$ is the *filter* of the rule. $\mathcal{C}$ selects the positive and $\mathcal{F}$ the negative context for the application of a rule. For more details on the transfer rules, see Oepen [2008]. This type of rules allows an extremely flexible transfer of factual and linguistic knowledge between the source and the target languages. The rules have access to the MRS structure for the source language, but can also access the (partially) constructed target language MRS structure. Thus elements in each rule could include parts from both MRS structures.

This approach requires a good deep grammar for the source language which produces complete MRS structures, then a complete set of rules for transferring of the source

[3]The anchors determine the tokens which generate the corresponding elementary predicates and related arguments. This information facilitates the transfer of information from the source text to target one.

language MRSes to the target language MRSes and a generation grammar for the target language. There are not many languages equipped with such grammars and rules.

The factor-based translation model is built on top of the factored SMT model proposed by Koehn and Hoang [2007], as an extension of the traditional phrase-based SMT framework. Instead of using only the word form of the text, it allows the system to take a vector of factors to represent each token, both for the source and target languages. The vector of factors can be used for different levels of linguistic annotations, like lemma, part-of-speech, or other linguistic features, if they can be (somehow) represented as annotations to each token.

In the project we have used the factored-based transfer of elementary predicates from MRSes in our Bulgarian to English and English to Bulgarian Pilot 1 system (see Deliverable D2.4). The results were only slightly better or slightly worse than the baseline system — Pilot 0. Thus we decided to implement a hybrid system combining automatic translation with transfer of deep information from the analysis of the source language to the automatic translation in the target language where this information, together with the language resources for the target language is used for a post processing over the automatic translation.

Thus, for our Pilot 3 system we have implemented a hybrid system which combines statistical machine translation and deep semantic processing. The main idea is to use the statistical machine translation as a translation model, then to exploit word alignments constructed by the statistical translation model in order to transfer deep semantic information from the source to the target text. The transferred linguistic knowledge is used for postprocessing of the target text, produced by the statistical MT model.

4.1 A Hybrid MT Architecture for Deep Transfer

Bulgarian ↔ English Pilot 3 systems build on Pilot 1 (described in D2.4) and Pilot 2 (described in D2.8). Pilot 3 presents a hybrid machine translation system consisting of three main steps (depicted in Figure 2 and further described in Sections 4.2.1–4.2.3). The source-language text is linguistically annotated, then translated with the Moses system to the target language and post-processed using the linguistic annotation projected from the source side.

During the translation with the Moses system the word alignment is stored in order to be used for the projection of the linguistic analyses from the source text to the target text.

It is important to mention that the number of the tokens in the source and the target language might differ. Also, the alignments can include many-to-many correspondences, not just one-to-one. Nevertheless, in practice about 80 % of the alignments are one-to-one or two-to-two tokens.

Here is an example of aligned texts annotated with morphosyntactic information of the English[4] sentence “Place them in the midst of a pile of dirty, soccer kit.” and its translation into Bulgarian[5]:

```
(place/VB them/PRP in/IN) = (postavyaneto/Ncnsd im/Ppetdp3;Ppetsp3;Pszt--3 v/R)
(the/DT midst/NN of/IN)   = (razgar/Ncmsi na/R)
```

[4]For English, the tagset of Pen treebank is used: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[5]For Bulgarian, the tagset of BulTreeBank is used: http://www.bultreebank.org/TechRep/BTB-TR03.pdf.

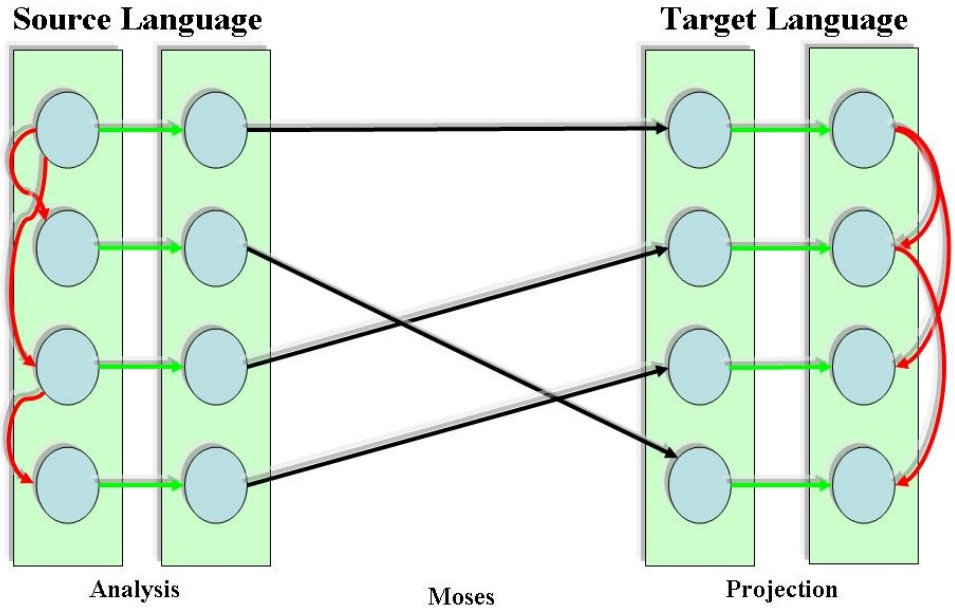


Figure 2: A hybrid architecture of bg↔en Pilot 3 for transferring linguistic information from the source to the target language. The linguistic analyses for the source language (Analysis - column 1) are projected to a tokenized source text (Analysis - column 2); then the Moses models (Moses) are applied for producing a target language translation. The translation alignment (Projection - column 1) is used for transferring the information to the corresponding tokens in the target language (Projection - column 2). The projected linguistic information interacts with the linguistic features of the tokens in the target text (for example the morphosyntactic features). Finally, the resulting annotation of the target text is used for post-processing.

```
(a/DT pile/NN of/IN)      = (kup/Ncmsi)
(dirty/JJ)                = (izmyrsyavam/Vpitf-r1s)
(,/,)                     = (,/Punct)
(soccer/NN)               = (futbolni/A-pi)
(kit/NN)                  = (komplekt/Ncmsi)
(./.)                     = (./Punct)
```

From the alignment and rules for mappings between the two tagsets we could establish the following alignments on token level:

```
(them/PRP)    = (im/Ppetdp3;Ppetsp3;Pszt--3)
(in/IN)       = (v/R)
(midst/NN)    = (razgar/Ncmsi)
(of/IN)       = (na/R)
(pile/NN)     = (kup/Ncmsi)
(,/,)         = (,/Punct)
(soccer/NN)   = (futbolni/A-pi)
(kit/NN)      = (komplekt/Ncmsi)
(./.)         = (./Punct)
```

Additionally, the alignment `(place/VB) = (postavyaneto/Ncnsd)` would be possible because the noun `(postavyaneto/Ncnsd)` is a deverbal noun, derived from a verb

(postavyam/Vpitf-r1s). To establish such an alignment we would need a derivational lexicon which however is not available to us. Thus, we do not consider this type of alignment. Likewise, the alignment between the English adjective (dirty/JJ) and the Bulgarian verb (izmyrsyavam/Vpitf-r1s) would be also possible as much as "something to be dirty" could be a result from the action denoted by the verb. We consider such rules also quite unreliable and thus we do not have such alignment rules.

On the basis of these alignments, we were able to transfer additional information like dependency links, word senses and elementary predicates. It is clear from the example that the transfer is only partial. The alignment (soccer/NN) = (futbolni/A-pi) is allowed because of the fact that they form a compound (see below).

After the transfer of linguistic information, a set of rules for post processing are applied. For example, here a rule for agreement between the adjective futbolni and the noun komplekt has been applied. Our work on the projection of linguistic analyses from the source to the target text is similar to Ramasamy et al. [2014] and Mareček et al. [2011].

4.1.1 Linguistic Processing of Source Languages

A linguistic pipeline is used which consists of the following processing tools over the source text:

- **Tokenizer.** This tool segments the input text in tokens and sentences. The confirmed tokens are common for each of the next processing steps.
- **POS tagger and lemmatizer.** It determines the part-of-speech and the grammatical features of the tokens in the input text. The lemma for each token is identified after the POS tagging.
- **Dependency parser.** We use dependency parsers that produce Universal Dependency annotations of the sentences.
- **Word Sense Disambiguation module.** A knowledge-based approach to WSD is exploited which uses a WordNet for the construction of the knowledge graph.
- **Minimal Recursion Semantics module.** Robust Minimal Recursion Semantic (RMRS) structures are constructed over the results from the previous processing steps. The rules are described in the Deliverable D4.1.

The important requirement here is that each linguistic piece of information added by the processing tools is assigned to tokens produced by the tokenizer. On the basis of this we define two ways of representation of the rest of the linguistic knowledge: unary and binary relations. Unary relations represent characteristics of the tokens. They are parts-of-speech and grammatical features, lemmas, word senses, elementary predicates and the main variable in RMRS. These are assigned to a single token. The binary relations connect two tokens. They represent dependency relations and predicate-to-argument relations in RMRS. This representation is used by different language resources including Universal Dependency Treebanks and MRS representations in DeepBank.

4.1.2 Alignment Enrichment

In this section we describe the approach we use for enrichment of the alignment generated during the statistical machine translation. For each sentence we have an alignment in the form of phrase alignments where each phrase alignment is as follows:

$$(st_{i_1}\ st_{i_2}\ st_{i_3}\ \ldots\ st_{i_k}) \Leftrightarrow (tt_{j_1}\ tt_{j_2}\ tt_{j_3}\ \ldots\ tt_{j_m})$$

where st_{i_n} is a source language token and tt_{j_o} is a target language token. Our goal in the hybrid architecture for deep semantic transfer is to use this alignment for transfer of linguistic knowledge from the source to the target language. In order to do this we need a more fine-grained alignment on the basis of tokens. Thus, our goal is from the alignment:

$$(st_{i_1}\ st_{i_2}\ st_{i_3}\ \ldots\ st_{i_k}) \Leftrightarrow (tt_{j_1}\ tt_{j_2}\ tt_{j_3}\ \ldots\ tt_{j_m})$$

to produce alignments of the following kind:

$$(st_{i_n}) \Leftrightarrow (tt_{j_o})$$

assuming that the source language token st_{i_n} is translated into the target language token tt_{j_o}. In order to do this kind of alignments we exploit the source language morphosyntactic annotation and the potential target language annotation:

$$(st_{i_n}/stag_{i_n}) \Leftrightarrow (tt_{j_o}/ttaglist_{j_o})$$

where $stag_{i_n}$ is the morphosyntactic annotation of the token st_{i_n} and $ttaglist_{j_o}$ is a list of the potential morphosyntactic tags for the token tt_{j_o}.

Such alignment is constructed by examination of all combinations of token correspondences from the phrase alignments. Then we formulated rules to expel from candidate token alignments the impossible ones. The rules have the following form:

$$stagT \Rightarrow ttagT$$

where $stagT$ is a template for source language morphosyntactic tags and $ttagT$ is template for target language morphosyntactic tags. For a candidate token level alignment $(st_{i_n}/stag_{i_n}) \Leftrightarrow (tt_{j_o}/ttaglist_{j_o})$ such a rule is applicable if the source language tag template matches the tag $stag_{i_n}$, then the target language tag template is evaluated with respect to each tag in the list $ttaglist_{j_o}$. If a target language tag can not match the template it is excluded from the list. If after application of the rule the list is empty, then the candidate token level alignment $(st_{i_n}/stag_{i_n}) \Leftrightarrow (tt_{j_o}/ttaglist_{j_o})$ is deleted from the candidate set. If the list is not empty then we keep the resulting token alignment as a candidate: $(st_{i_n}/stag_{i_n}) \Leftrightarrow (tt_{j_o}/ttaglist^R_{j_o})$, where $ttaglist^R_{j_o}$ is the reduced list of tags. After the application of the rules there are several possibilities:

- **1 to 0.** For a given source language token there is no alignment to a target language token;
- **0 to 1.** For a given target language token there is no alignment from any source language token;
- **1 to n.** For a given source language token there are more than one alignments to target language tokens;
- **n to 1.** For a given target language token there are more than one alignments from source language tokens;
- **1 to 1.** For a given source language token and a given target language token there is a unique token level alignment;
- **n to m.** For **n** given source language tokens there are **m** target language tokens between which there are token level alignments.

We assume that for the task of the projection of the linguistic analyses from the source text to the target text only unique token level alignments (**1 to 1**) could be used.

Thus after performing the application of the rules to all phrase alignments for a given sentence we have a new alignment between the sources and the target sentences in which there are a number of token level alignments of the kind:

$$(st_{i_n}/stag_{i_n}) \Leftrightarrow (tt_{j_o}/ttaglist_{j_o})$$

where the tokens st_{i_n} and tt_{j_o} participate only in one such alignment. This new sentence level alignment is called an **enriched sentence alignment**.

4.1.3 Linguistic Information Projection and Post-processing

The linguistic information to be projected is in the form of unary and binary relations over source tokens. The unary relations represent at least the following linguistic information:

- Elementary predicates and main variables. The elementary predicates originate from tokens in the source text. Each elementary predicate is related to a main variable.
- Grammatical features. Grammatical features are transferred on the basis of a mapping between the tagsets of the source and the target languages. The mapping is different from the rules described above for removing the impossible correspondences at token level alignments.

Binary relations correspond to relations between the main variable of an elementary predicate and one of the variables for the rest of the arguments of the elementary predicate.

For each token in the source language analysis for which there is a token level alignment we could transfer the unary relations assigned to the token by the source language analysis.

For each two tokens in the source language analysis for which there are unique token level alignments we could transfer the binary relations between the two tokens in the source sentence. For two token pairs with unique token level alignment there are no more than two possible binary relations — one from the dependency analysis and one from the RMRS analysis. A pair of source–target tokens with a unique token level alignment could participate in the transfer of more than one binary relation because the RMRS analysis forms a graph instead of a tree.[6]

The transferred linguistic information is used in the form of rules for modification of target language sentences. This is done in the following steps:

1. First, the transferred linguistic information forms a partial linguistic analysis of the target sentence.
2. Second, on the basis of the source sentence analysis and the (partial) target analysis we construct rules of the form discussed above:

$$[\mathcal{C} :]\mathcal{I}[!\mathcal{F}] \rightarrow \mathcal{O}$$

but, the rules here are more like templates in the sense that many elements in them are left underspecified. These rules determine the possible translations on different types of MRS structures from the source language to the target language. In this way, the main variable type from the source language could be changed to a different type of main variable and the type of the relation between two tokens. When the rules are applied each target token involved in the target RMRS needs to have at

[6]The same is true for the enhanced representation for Universal Dependency analyses.

least one appropriate morphosyntactic tag in $ttaglist_{j_o}$ list assigned to it that agrees with its elementary predicate type.

3. Third, the rules are instantiated with the corresponding tokens from token level alignments. In this way the template RMRS structures involved in the rules from step 2 become more concrete. This allows to check some grammatical and syntactic characteristics of the involved tokens and eventually to perform some post processing.

On the basis of the constructed (partial) RMRS structure for the target sentence we define post processing rules if some conditions are not met by the RMRS structure. We consider here two actions: change of word form and change of word order. In future we could extend the set of actions with insertion of tokens like obligatory elements (clitics, subject, for instance), deletion of tokens, etc. The rules of the following kinds:

- For each binary relation in the target language RMRS structure we define rules for their manipulations. These rules have the following form:

$$\langle tt_{i_n}, tt_{j_o}, rel, cond \rangle \rightarrow action$$

where for a given pair of target tokens $\langle tt_{i_n}, tt_{j_o} \rangle$ and relation rel between them. If the condition $cond$ is not fulfilled a certain action $action$ is performed. The actions could be one of the following: $form(tt)$ — this action performs change of the word form for the token and it is used to ensure agreement, and $reorder(tt_{i_n}, tt_{j_o})$ used for reordering of the two tokens. Possible conditions could be $agreement(tt_{i_n}, tt_{j_o})$ — it is true if the agreement between grammatical features are met, $prec(tt_{i_n}, tt_{j_o})$ — it is true if the token tt_{i_n} precedes tt_{j_o}, and $prec(tt_{i_n}, tt_{j_o})$ — it is true if the token tt_{i_n} immediately precedes token tt_{j_o}. The conditions have access to all information available to target tokens.

- For each target token for which there is a token level alignment and which is not related to any other target token via a binary relation in the (partial) RMSR structure of the target sentence we define rules of the following kind:

$$cond(tt) \rightarrow form(tt)$$

where tt is the target token, $cond(tt)$ checks whether the transferred source language grammatical features are in correspondence to at least one appropriate morphosyntactic tag in $ttaglist_{j_o}$ list assigned to it. If this is not the case the action $form(tt)$ is performed and it generates an appropriate word form for the token.

Using an enriched sentence alignment, the projected linguistic information and a set of rules of the above types one could manipulate the target language sentence in order to produce a new one which satisfies some of the constraints transferred from the source to the target sentences.

The rules described here could clash over the actions. For instance, there might be the case where two rules need to change the word order for two tokens. Here we do not consider this problem assuming an order of rule application. Thus, only one action could be applied for a given case — word form of a token, or word order of two tokens.

The described hybrid architecture provides utilities to exploit deep semantic knowledge for the transfer modules in machine translation. It is appropriate in cases when there is no a good deep grammar for a given language or a domain as in our case. It is an alternative to the LOGON architecture including the hybrid architecture presented in Oepen et al. [2007]. It presents a hybrid architecture which still exploits the deep grammar approach of LOGON, but with statistical components for selecting the best solutions at each translation step.

In the next sections the implementation of the hybrid architecture within QTLeap project is presented. We have also implemented some rules that are more specific to the domain of the project.

4.2 Implementation

4.2.1 Analysis

For the en→bg direction the source-language linguistic annotation consists of tokenization, POS tagging, lemmatization, dependency parsing, Minimal Recursion Semantics annotation and word sense disambiguation (see below).

The analysis of English (tokenization, lemmatization, POS tagging and dependency parsing) as a source language was done with the CoreNLP tools[7] of Stanford University. The word sense disambiguation was done by the UKB tool.[8] The MRS structures and the post-processing rules were implemented in the CLaRK System.[9]

For the analysis of Bulgarian as a source language, we trained Mate tools[10] on the Bulgarian treebank.

In order to adapt the processing to the domain, we have annotated Batch1 and Batch2 with morphosyntactic information.

4.2.2 Transfer

In the en→bg system, we use a two-step translation strategy depicted on Fig. 3.

The first step (Moses1) is done using a phrase-based Moses model. We have used the following parallel data: SETimes parallel corpus, LibreOffice parallel corpus, Bulgarian English Dictionary aligned on wordform level, Microsoft product descriptions and Microsoft Terms.

The texts were tokenized with the tokenizers for the corresponding languages.

We trained a phrase-based Moses model using the following options `-alignment grow-diag-final-and` and `-reordering msd-bidirectional-fe`.

The language model used is a 5-gram language model trained with SRILM on the data from SEtimes corpus, LibreOffice corpus, domain articles from Wikipedia, Microsoft data, and the Bulgarian National Reference Corpus (mainly news data and fiction).

The tuning was done on Batch1a of the QTLeap corpus.

[7] http://stanfordnlp.github.io/CoreNLP/
[8] http://ixa2.si.ehu.es/ukb/
[9] http://www.bultreebank.org/clark/index.html
[10] http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/matetools.en.html

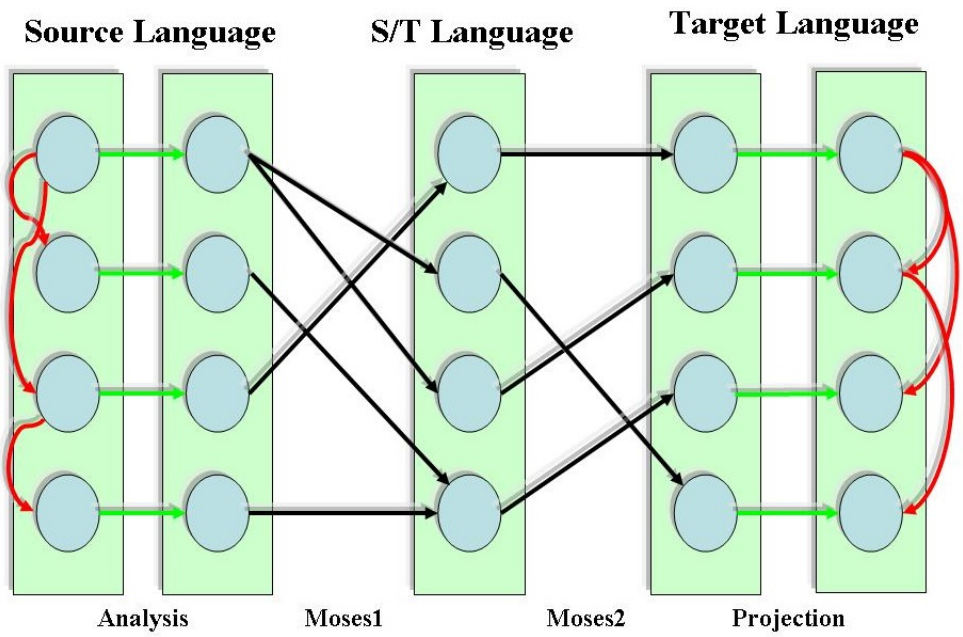


Figure 3: A hybrid architecture of en→bg Pilot 3 for transferring linguistic information from the source to the target language. The linguistic analyses for the source language are projected to a tokenized source text; then Moses models (Moses1 and Moses2) are applied for producing a target language translation. The translation alignment is used for transferring the information to the corresponding tokens in the target language. Finally, the target linguistic annotation is used for post-processing.

The second step (Moses2) includes a factor-based Moses model (similar to the en→bg Pilot 2 system) which starts from a partially translated source language (S/T language).

In S/T language, some of the source tokens were replaced with target-language lemmas using the results from the WSD of the source language. The result from step one was used to extend the S/T-language text with lemmas for words that are not translated via WSD.

The training was done on the same parallel data but processed as in the example given on page 26. Both the source (**English sentence with factors**) and the target (**Bulgarian sentence with factors**) texts were processed with the corresponding language pipelines. The options used in training the factor-based model are `-translation-factors 0,2-0,2+1,2-0,2`, `-decoding-steps t0:t1`, `-alignment grow-diag-final`, and `-reordering distance`.

The factors are `SWF-TL|SL|STag` for the input text and `TWF|TL|TTag` for the output. `SWF-TL` denotes source language word form or target language lemma, if the pipeline established correspondence for the input source word form. `SL` denotes the lemma for source language word form, `STag` denotes the POS tag for source language word form. `TWF`, `TL`, and `TTag` denote target language word form, lemma, and POS tag.

The language model used is a 5-gram language model trained with SRILM on the data from SEtimes corpus, LibreOffice corpus, and the Bulgarian National Reference Corpus (mainly news data and fiction).

The tuning was done on Batch1a of QTLeap corpus.

For some functional words (where we are sure about the alignment), the source language word form was replaced with the target language lemma from the translation produced by the Moses1 model. The idea is to maximise the number of substituted source

type of change English→Bulgarian	frequency	example
noun1 noun2→noun1 noun2	rare	business meeting→biznes sresta
noun1 noun2→noun2 noun1	frequent	Rila mountain→planina Rila
noun1 noun2→adj1 noun2	frequent	antivirus software→antivirusni programi
noun1 noun2→noun2 prep noun1	frequent	email settings→nastrojki za posta

Table 5: Examples of structural changes in translation of English noun-noun phrases into Bulgarian.

language word forms in S/T language text.

For the bg→en system, we are using the same parallel corpora and options as for the phrase-based model for en→bg, but the whole transfer is done in one step.

For the language model we are using the SEtimes corpus, the LibreOffice corpus, domain articles from Wikipedia, Microsoft data, and the Europarl corpus.

The tuning was done on Batch1q of the QTLeap corpus.

4.2.3 Post-processing

The post-processing is a rule-based system that includes linguistically-enhanced information. This information is projected from the source side with the help of the word alignments produced by Moses1 and Moses2. The projected information is the linguistic knowledge in the form of RMRS-based elementary predicates, labeled dependencies, word senses (synset ids from WordNet), and POS tags in the source language.

It should be noted that the alignment between the source language and the S/T language and between the S/T language and the target language is not one-to-one. It generally maps sets of tokens from the source language to sets of tokens in the target language. Thus, as presented above in the example, the transfer of the linguistic information from the analysis of the source language to the target one is not straightforward. Here we apply heuristic rules. Thus, the transferred linguistic information is only partial. For the rules definition we also exploited the language resources and tools for the target language — a morphological lexicon, a lemmatizer, and a morphological generator.

Once the linguistic annotation is projected via the alignment, the post-processing rules can be applied.

An example of a word order rule is the transformation of the English noun compounds into the appropriate syntactic structures in Bulgarian. The different templates are represented in Table 5. The direct transfer is rare, since the NN compounds are not so frequent in Bulgarian. The combination in which the first noun is a Named Entity is the most frequent one in the domain data. In the case of a phrase with an adjective and a noun in Bulgarian, a rule for word form change reflecting the agreement is applied. Other rules are concerned with agreement between the anchors of arguments in different elementary predicates: the anchor of an adjective elementary predicate and the anchor of the argument for the noun modified by the adjective, agreement by the anchor for a verb elementary predicate and the anchor of the argument corresponding to its subject. Rules applicable to not related tokens modify the word form of the token when the source

language grammatical features require a different form. For example, when a plural noun is translated into a singular one. The results for Pilot3 are presented in Deliverable D2.11.

4.3 Using WSD for Factor-based MT

For the en→bg translation, we extended the above architecture by adding another Moses model. Our goal was to reuse the Pilot 2 setup (see Deliverable 5.7).

The motivation for using the representative lemma in the target language is our expectation that we can unify the various synset IDs with the similar translations in the target language. For example, in the en→bg direction, the two concepts referred by *donor*: `wn30-10025730-n` ("person who makes a gift of property") and `wn30-10026058-n` ("a medical term denoting someone who gives blood or tissue or an organ to be used in another person") are very close to each other. They have the same translation in Bulgarian in both corresponding synsets: *donor*. The representative word is selected on the basis of a frequency list of Bulgarian lemmas constructed over large corpora (70 million words).

As an example, the procedure we performed with respect to the training, testing, and tuning of the Moses system is as follows:

English sentence:
This is real progress .

English sentence with factors:
this|this|dt is|be|vbz realen|real|jj napredyk|progress|nn .|.|.

Bulgarian sentence with factors:
tova|tova|pd e|sym|vx realen|realen|a napredyk|napredyk|nc .|.|pu

Bulgarian sentence:
Tova e realen napredyk.

We selected this Moses model for en→bg because in the earlier versions of Pilot 3 it performed slightly better than the phrase-based model.

The architecture for en→bg is depicted in Figure 3. Here the source language is analyzed linguistically, then the tokenized text is processed in two ways in order to produce the text for the Source/Target text (in the example above it corresponds to **English sentence with factors**). First, the replacements with the Bulgarian lemmas were done on the basis of Word Sense annotation of the source text. Additionally we translated the source text with phrase-based Moses model (Moses1).

From this translation we selected some words to be used as factors. The idea was to enrich S/T text with more target-language factors. Here it is important to keep in mind that the number of tokens in the S/T text is the same as in the source text. Thus, the analyses produced for the source text are easy to transfer to the S/T text.

Then the actual translation was done with factor-based Moses model (Moses2) where the alignment is used for the projection of the linguistic analyses over the source text. The result for the factor-based Moses model is presented in Deliverable D5.11.

5 MT enhanced with Hybrid Generation Module

Our experiments regarding generation for deep MT primarily focused on lexical choice. Lexical choice is a subtask of Natural Language Generation (NLG), where an ideal model produces varied, natural-sounding, linguistic utterances.

The choice of a correct lemma is a difficult task which depends heavily on the quality of the dictionaries used. One such dictionary is WordNet [Fellbaum, 1998], a lexical semantic database containing lemmas corresponding to their word meanings. Querying this database for a word returns a group of one or more synonyms called a synset containing a set of words of the same class. Being roughly synonymous in one of their meanings, makes them well suited for lexical choice.

Unfortunately, not every lemma in a synset is a full synonym of its original word which could cause errors when selecting the most probable variant without considering the context. Consider for instance the English WordNet synset: {"employment", "work"}. Both lemmas in this synset have the meaning of "The occupation for which you are paid". In the sentences of example 1 they are perfectly exchangeable. In example 2, however, both sentences require a different lemma in this particular context. This indicates that a more sophisticated system is required that selects a correct lemma given a synset while considering its context.

(1) 'He is looking for **employment**'
'He is looking for **work**'

(2) 'He is out of **employment***'
'He would like to terminate his **work*** agreement.

To this end, we mapped dependency trees over synsets to dependency trees over lemmas while taking into account both context information and the frequency of the lemma and synset combination. A dependency tree is a labeled tree in which nodes correspond to the words of a sentence. It contains edges that represent the grammatical relations between those words. The latter makes them a good source for contextual information.

Our model takes as input directed labeled dependency trees with nodes corresponding to Wordnet synsets and its edges corresponding to syntactic relations. We use a Hidden Markov Tree Model (HMTM) and a Tree-Viterbi algorithm (Crouse et al. [1996], Durand et al. [2004], Žabokrtský and Popel [2009]) to label the nodes of our dependency trees with a correct lemma by revealing the hidden states in the tree nodes, given another, observable, labeling of the nodes of the same tree.

The independence assumptions that are made by HMTMs can be useful for modeling dependency trees. They fit dependency trees well, since they assume conditional dependence only along the tree edges, which corresponds to intuition behind the linguistic dependency relations in dependency trees. Moreover, HMTMs can be used for the labeling of nodes in a dependency tree. In our model, this can be interpreted as the revealing of the hidden states (the lemmas) in the tree nodes, given another observable labeling of the nodes of the same tree (the synsets) by use of a tree-modified Viterbi algorithm. The resulting labeled dependency trees should then comprise the correct lemmas given their context and can be used as input to a generation system.

5.1 Method

HMTMs are similar to the well known Hidden Markov Models (HMM) as they both contain a sequence of observed states with corresponding hidden states [Diligenti et al., 2003, Durand et al., 2004]. However, instead of a linear chain of observations, they map over a tree of observations. Furthermore, analogously to regular HMMs, HMTMs rely on emission probabilities and transition probabilities. In the Markov process for lexical choice, we assume that we are given a directed labeled dependency tree. Its

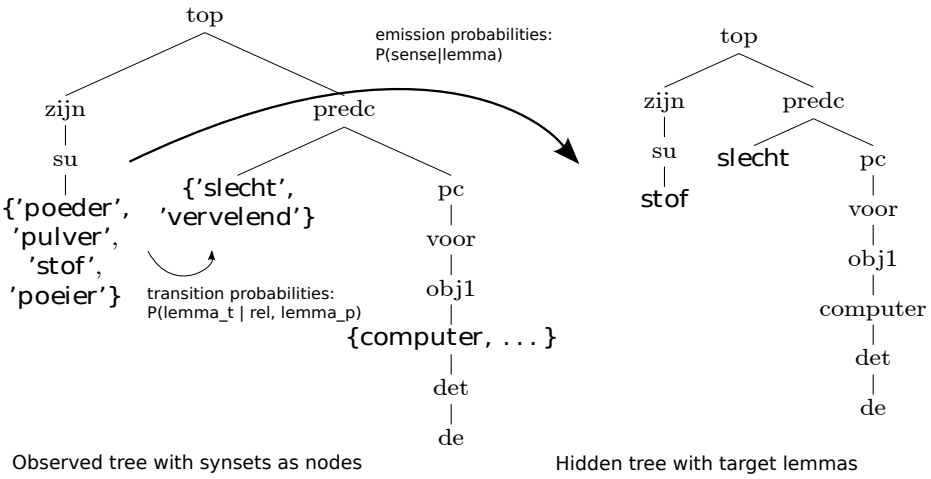


Figure 4: HMTM for the Dutch sentence: *"'**Stof** is slecht voor de computer"* ("Dust is bad for your computer")

nodes correspond to synsets, used for emission probabilities, and its edges correspond to syntactic-semantic relations, used for transition probabilities.

The hidden states in the tree nodes are revealed on the basis of an observed labeling of the nodes of the same tree. When using HMTMs for lexical choice, the hidden states consist of actual lemmas, whereas the observations are word senses (synsets). Figure 4 contains an example of an HMTM containing synsets for a Dutch sentence. In this particular context from the synset $\{'poeder', 'pulver', 'stof', 'poeier'\}$, the best choice of lemma would be "stof", and it is up to the tree-viterbi algorithm to choose this option over the other lemmas, given the synset and its context.

The tree is defined by an observed dependency tree containing synsets as nodes, $S = \{S(n_1), ..., S(n_m)\}$, and a hidden tree with target lemmas, $T = \{T(n_1), ..., T(n_m)\}$, isomorphic to the observed tree where m is the size of the tree. The function $\pi(n)$ returns a pair (n', l) where n' is the unique parent of node n (with r corresponding to the root of the tree), and l is a label indicating the nature of the dependency. Such labels include *subject, object, modifier, ...* Each node, except the root node, refers to a word in the sentence. Like HMMs, HMTMs make two independence assumptions: given $T(\pi(n))$, $T(n)$ is conditionally independent of other nodes and given $T(n)$, $S(n)$ is conditionally independent of other nodes.

The required frequency counts for the emission probabilities of a sense given a target lemma can be obtained from sense annotated corpora or from the output of WSD-systems. We need to estimate the probability of an observed state (the sense), given the hidden state (the lemma):

$$P(sense|lemma) \approx \frac{freq(sense, lemma)}{freq(lemma)} \quad (1)$$

Consider for instance the probability of the synset $\{lager, beer, ale, ...\}$ given the lemma "beer". If the lemma "beer" is associated in the corpus with the $\{lager, beer, ale, ...\}$ sense in 89 out of a 100 cases, then the emission probability will be estimated as 0.89.

The transition probabilities of a target lemma $lemma_t$ given a dependency relation rel (such as subject, object or modifier) and its parent $lemma_p$, can be collected from

large parsed corpora. For this we can use the following equation:

$$P\left(lemma_t \mid rel, lemma_p\right) \approx \frac{freq(lemma_p, rel, lemma_t)}{freq(lemma_p, rel)} \tag{2}$$

If we want the probability of the lemma "beer" given a parent "drink" in the dependency relation "obj" we compute:

$$P(t_i|t_{i-1}) = p\left(beer \mid obj1, drink\right) = \frac{freq(drink, obj1, beer)}{freq(drink, obj)} \tag{3}$$

The frequency of a lemma given its parent is the count of how often its parent appears in relation rel and N is the total number of p as arguments of rel. For example, if "drink" occurs 40 times with an object, and in 20 cases that object is the lemma "beer", then we estimate the probability as 0.5.

The most probable hidden tree labeling given the observed tree labeling can then be found by use of a modification of the traditional Viterbi algorithm for HMMs to a Tree-Viterbi algorithm for HMTMs. Details on this modification can be found in [Durand et al., 2004] and [Diligenti et al., 2003]. The tree-viterbi algorithm, starts at its leaf nodes and continues upwards. In every node of each state and each of its children, a downward pointer to the optimal hidden state of the child is stored. Downward recursion is then used along the pointers from the optimal root state in order to retrieve the most probable hidden tree.

5.2 Experiments

Our model takes abstract dependency structures over senses as input, which can easily be obtained by applying WSD on the source side of the MT pipeline, store them in the nodes of the dependency tree, and retain them during transfer. For this, we used the UKB-WSD module [Agirre and Soroa, 2009] in the English analysis phase of the MT pipeline resulting in dependency trees containing synsets in their nodes. The English synsets are then converted to Dutch synsets from the Cornetto database [Vossen et al., 2013]. For each node in the dependency tree that contains a synset, our model is used to find an optimal lemma given its synset and its context. This lemma is then used to substitute the one that was originally translated by the translation model in the transfer phase. Ultimately, the trees are used to generate full sentences.

We used the answers-part of Batch 1 of the QTLeap corpus [Osenova et al., 2015]. In addition, the model was tested on a broader domain test set. For this, we took 1000 sentences from the English-Dutch News Commentary data set [Tiedemann, 2012].

The data is analyzed and translated from English to Dutch with Treex, a tree-to-tree machine translation system whose translation process follows the analysis-transfer-synthesis pipeline [Žabokrtský et al., 2008, Popel and Žabokrtský, 2010]. The sentences are analyzed and translated from English to Dutch with Treex, a modular framework for natural language processing [Popel and Žabokrtský, 2010]. It contains a tree-to-tree machine translation system whose translation process follows the analysis-transfer-synthesis pipeline [Žabokrtský and Popel, 2009]. In the analysis phase, a source sentence is transformed into a deep syntax dependency representation. Isomorphism of the tree representation is mostly assumed in both languages, translating the tree node-by-node. In the English to Dutch pipeline, the resulting dependency trees are transferred to Dutch abstract representations that are the input for the generation of Dutch sentences. In the

Table 6: BLEU scores for English-Dutch translation using the tree-viterbi algorithm

	QTLeap	News
Most frequent	20.16	07.00
Tree-viterbi	21.93	07.45

analysis phase, a source sentence is transformed into a deep syntax dependency representation. The resulting dependency trees are transferred to Dutch abstract representations that are the input for the generation of Dutch sentences.

For generation we use the the Alpino Generator [van Noord, 2006, De Kok, 2013] that generates Dutch sentences on the basis of an abstraction of dependency structures. The Alpino system for Dutch [van Noord, 2006] is a collection of tools and programs for parsing Dutch sentences into dependency structures, and for generating Dutch sentences on the basis of an abstraction of dependency structures. Since dependency structures for generation contain less information (such as word order and word inflection) than dependency trees, we refer to them as Abstract Dependency Trees (ADTs) [De Kok, 2013].

ADTs model the grammatical relations between lexical items and categories built from lexical items. Similar to a normal dependency tree, they contain a syntactical representation of a sentence in the form of a tree. In the Alpino Generator [De Kok, 2013], the grammar is used in the reverse direction as for parsing. The process starts with an abstract dependency structure and then uses the grammar to construct one or more sentences. The generation process starts with an ADT and then uses the grammar to construct one or more sentences. Ultimately, a statistical model is used for choosing the most fluent one.

The emission probabilities are taken from DutchSemCor Vossen et al. [2012]. An important issue, however, is that the sense-annotated corpus only contains counts for a limited number of lemmas, which can be problematic when estimating emission probabilities. If the target lemma does not appear in the corpus, it is not considered, possibly causing a less suitable lemma to be chosen. In our data, for example, for a synset containing the following senses: *{'bladzijde', 'pagina', 'zijde'}* the lemma "pagina" will not be chosen as it does not appear in our sense-tagged corpus. We therefore applied a simple heuristic on these "missing" lemmas to estimate the probability of the sense by dividing 1 by the number of synsets the lemma appears in. Transition probabilities are obtained from a large set of parsed corpora. For this we take the SONAR part of Lassy Large [van Noord et al., 2013], containing approximately 500M words. From these parses, dependency relations and their counts are retrieved resulting in a transition probability matrix that can be queried for each lemma given its parent lemma and their relation.

5.3 Results

In WSD, a typical baseline consists of taking the most frequent sense of the target word. We adopt the idea behind this baseline for the lexical choice problem by looking at the frequency distribution of a lemma/synset combination in DutchSemCor. As can be seen in table 6, the Tree-Viterbi algorithm performs better compared to the most frequent baseline. This is confirmed by a manual comparison of the lemmas chosen by our model

with the ones picked by the baseline system. For this evaluation we took a random subset of 100 lemmas, chosen by our model, and assessed whether they where either correct or incorrect choices. If the choice of lemma was incorrect it was determined whether this was either caused by the model itself or by the fact that a wrong WordNet synset was chosen as input. From these 100 choices, 56 where correct lexical choices. Of the remaining 44 erroneous choices, 33 are caused by a wrong input sense causing the output to be worse compared to the original MT output.

Our manual evaluations shows that the amount of wrong input senses is fairly large. However, when our model does get a correct input sense, it makes a better choice in most cases. In the output, examples can be found where the tree-viterbi algorithm outperforms the baseline. In example 3, for instance, the baseline system chooses the incorrect adjective "bezig" (busy) instead of "actief" (active). Example 4 demonstrates that the inclusion of dependency information of the tree-viterbi algorithm works well. The most frequent lemmas that replace "harde schijf" (hard disc) with "sterke bedrag" (strong amount) are avoided by our model, even though the wrong synset was chosen as input. For the replacement of 'schijf' the system can choose between "bedrag" (amount), "schijf" (disc) and "som" (sum), with the latter having the highest emission probability. For the lemma "hard" it can choose between multiple adjectives, including "sterk". The transition probabilities of the combination of both lemmas ensures that the right lemmas are chosen by the algorithm.

(3) *English original:* Access your friend's profile to see if the account is still **active**.
Most frequent: Open je vriend profiel om te kijken of het account nog **bezig** ("busy") is.
Tree-viterbi: Open je vriend profiel om te kijken of het account nog **actief** ("active") is.

(4) *English original:* Your **hard drive** has two USB configurations.
Most frequent: Je **sterke bedrag** ("strong amount") heeft twee USB uitvoeringen.
Tree-Viterbi: Je **harde schijf** ("hard drive)" heeft twee USB uitvoeringen.

5.4 Tree-Viterbi for out-of-vocabulary items

Our model appears to make an improvement in particular in cases where the original language model was not able to find a translation and therefore yields the original, non-translated, word. Using the algorithm for lexical choice only on these out-of-vocabulary items (OOV's) could thus improve the output.

Table 7: BLEU scores for English-Dutch translation using the tree-viterbi algorithm only on OOV's

	QTLeap	News
Original MT-output	23.02	08.52
Tree-viterbi for OOV	23.03	08.63

In table 7, the results can be found of a second experiment on OOV's. The BLEU scores for both test sets are similar to the scores of the system without lexical choice. However, when looking at the output manually, the algorithm does improve the original sentences in almost all cases, yielding more fluent and natural sounding sentences. Examples of these improvements can be seen in sentence 5 and 6. Since OOV's are not very common in the test data, our model finds a Dutch word for 132 of them, which could explain the small difference in BLEU score. From a random subset of these lexical choices, it was manually assessed that 7 of them are wrong choices made by our model, 33 of them are errors caused by a wrong input sense, while 60 are good lexical choices.

(5) Zet je wachtwoord in [**rectangle => rechthoek**] en klik op log In.
Insert your password in the rectangle and click Log In

(6) Probeer de belangrijke combinatie van [**brightness => helderheid**] te controleren.
Try to check the important combination of brightness

5.5 Discussion

From our evaluation it becomes clear that the model is able to select the correct lemma if the correct input synset is available. However, it also becomes apparent that the algorithm for lexical choice is not very suitable in an MT-Setup. Although the HMTM outperforms the most frequent baseline, the MT-system does not improve when the algorithm is used on OOVs. More importantly, since the systems are trained on domain-specific data which probably decreases the chance of translating a lemma incorrectly in the transfer-phase, they are rarely incorrect. This leaves little room for improving the output by way of lexical choice, while still having the same chance of causing errors.

The main cause of errors, therefore, are wrong input senses. In most of the cases where the tree-viterbi model chooses a wrong lemma, an incorrect synset was used as input. A target lemma cannot be retrieved from a source synset in some context if the input synset appears in a different sense than the one in which it is synonymous with the target. When a wrong synset is chosen as input, the system has a high chance of selecting a wrong lemma. Consider for example the lemma "menu", that appears in two Dutch synsets:

(7) a. {menukaart:noun:1', 'menu:noun:1', 'spijskaart:noun:1', 'kaart:noun:4'}
b. **{'menu:noun:3', 'keuzemenu:noun:1'}**

Since the data used for the experiments belongs to the IT domain, the second synset, in bold, is the preferred one. When first synset, with the meaning of *restaurant menu* would be used as input to the algorithm the lemma "kaart" (map) could be chosen in stead of "menu" which is usually not preferred in this domain.

In this MT-setup, where the transfer model is trained on domain specific data, our system is, not able to significantly improve the translation output. Since the systems are trained on domain-specific data which probably decreases the chance of translating a lemma incorrectly in the transfer-phase, they are seldom wrong. This leaves little room for improving the output by way of lexical choice, while still having the same chance of causing errors. However, the fact remains that our model is able to yield a similar score without the use of domain specific parallel data and could therefore be useful in settings where this data is not available.

Another problem that is highly likely to cause errors in this setup are mistakes in the analysis and/or the transfer phase. For example, errors in the assignment of part-of-speech tags or dependency relations could have negative effects on the outcome since it would not be possible to find correct transition probabilities in the transition matrix.

6 Conclusions

The deliverable presents different approaches and experiments for using deep processing to enhance machine translation including processing of OOVs, MWEs, transfer of semantic information and generation.

The OOVs and MWEs approaches are active during the analysis of the source language as well as during the generation into the target language. The improvements demonstrated in the experiments show their usefulness. In some cases the improvement is modest (for example, the transliteration of Bulgarian names into English), which in our view is because of two reasons: (1) in some cases the underlining MT model succeeded with the problem (for example, learn many names from the training corpus), and (2) the cases of OOVs and MWEs do not have that much impact on the automatic evaluation metrics.

The hybrid architecture for deep semantic transfer demonstrates the utility of exploiting deep semantics for MT even in case of languages with not enough language resources or in case of new domains. The architecture allows inclusion of new processing modules and usage of additional language resources like bilingual lexicons, including aligned valency lexicons.

References

Eneko Agirre and Aitor Soroa. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 33–41, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1609067.1609070.

António Branco and Francisco Costa. A deep linguistic processing grammar for Portuguese. In *Proceedings of the 9th Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR)*, number 6001 in Lecture Notes in Artificial Intelligence (LNAI), pages 86–89. Springer, 2010.

António Branco, Francisco Costa, João Silva, Sara Silveira, Sérgio Castro, Mariana Avelãs, Clara Pinto, and João Graça. Developing a deep linguistic databank supporting a collection of treebanks: the CINTIL DeepGramBank. In *Proceedings of the 7th Language Resources and Evaluation Conference (LREC)*, pages 1810–1815, 2010.

Ann Copestake. Robust minimal recursion semantics (working paper)., 2003.

Ann Copestake. Applying robust semantics. In *Proceedings of the 10th Conference of the Pacific Assocation for Computational Linguistics (PACLING)*, pages 1–12, 2007.

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

Matthew S Crouse, Richard G Baraniuk, and Robert D Nowak. Hidden Markov Models for Wavelet-based Signal Processing. In *Signals, Systems and Computers, 1996. Conference Record of the Thirtieth Asilomar Conference on*, pages 1029–1035. IEEE, 1996.

D.J.A. De Kok. *Reversible Stochastic Attribute-value Grammars*. Groningen dissertations in linguistics. 2013. ISBN 9789036761123. URL http://books.google.nl/books?id=uNlcmwEACAAJ.

Michelangelo Diligenti, Paolo Frasconi, and Marco Gori. Hidden tree Markov Models for Document Image Classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(4):519–523, 2003.

Jean-Baptiste Durand, Paulo Goncalves, and Yann Guédon. Computational Methods for Hidden Markov Tree Models-An Application to Wavelet Trees. *Signal Processing, IEEE Transactions on*, 52(9):2551–2560, 2004.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May 1998. ISBN 978-0-262-06197-1.

Dan Flickinger. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28, 2000.

Max Jakob, Markéta Lopatková, and Valia Kordoni. Mapping between dependency structures and compositional semantic representations. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 2491–2497, 2010.

Philipp Koehn and Hieu Hoang. Factored translation models. In *Proceedings of EMNLP*, 2007.

David Mareček, Rudolf Rosa, Petra Galuščáková, and Ondřej Bojar. Two-step translation with grammatical post-processing. In Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan, editors, *Proceedings of the Sixth WMT*, pages 426–432, Edinburgh, UK, 2011. University of Edinburgh, ACL. ISBN 978-1-937284-12-1.

Dieke Oele and Gertjan van Noord. Lexical choice in abstract dependency trees. In *1st Deep Machine Translation Workshop*, pages 73–80, Prague, Czech Republic, 2015.

Stephan Oepen. The Transfer Formalism. General Purpose MRS Rewriting. Technical Report LOGON Project. Technical report, University of Oslo, 2008.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher Manning. LinGO Redwoods: A rich and dynamic treebank for HPSG. *Research on Language and Computation*, 2:575–596, 2004.

Stephan Oepen, Erik Velldal, Jan Tore Lønning, Paul Meurer, Victoria Rosén, and Dan Flickinger. Towards hybrid quality-oriented machine translation. on linguistics and probabilities in mt. In *In Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation*, pages 144–153, 2007.

Petya Osenova, Rosa Del Gaudio, João Silva, Aljoscha Burchardt, Martin Popel, Gertjan van Noord, Dieke Oele, and Gorka Labaka. Interim Report on the Curation of Language Resources and Tools for Deep MT. Technical Report Deliverable D2.5, Version 2.0, QTLeap Project, 2015.

Martin Popel and Zdeněk Žabokrtský. Tectomt: Modular nlp framework. In *Proceedings of the 7th International Conference on Advances in Natural Language Processing*, IceTAL'10, pages 293–304, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-14769-0, 978-3-642-14769-2. URL http://dl.acm.org/citation.cfm?id=1884371.1884406.

Loganathan Ramasamy, David Mareček, and Zdeněk Žabokrtský. Multilingual dependency parsing: Using machine translated texts instead of parallel corpora. *The Prague Bulletin of Mathematical Linguistics*, 102:93–104, 2014. ISSN 0032-6585.

João Silva. *Robust Handling of Out-of-Vocabulary Words in Deep Language Processing*. PhD thesis, University of Lisbon, 2014.

Jörg Tiedemann. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Velislava Todorova and Kiril Simov. Training automatic transliteration models on dbpedia data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 654–662, Hissar, Bulgaria, September 2015. INCOMA Ltd. Shoumen, BULGARIA. URL http://www.aclweb.org/anthology/R15-1084.

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, Gosse Bouma, Frank Van Eynde, Daniël de Kok, Jelmer van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste. *Large Scale Syntactic Annotation of Written Dutch: Lassy*, pages 147–164. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-30910-6. doi: 10.1007/978-3-642-30910-6_9. URL http://dx.doi.org/10.1007/978-3-642-30910-6_9.

Piek Vossen, Attila Görög, Rubén Izquierdo, and Antal Van den Bosch. DutchSemCor: Targeting the Ideal Sense-tagged Corpus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

P.T.J.M. Vossen, H. van der Vliet, I. Maks, R. Segers, M.-F. Moens, K. Hofmann, E.F. Tjong Kim Sang, and Maarten de Rijke, editors. *Cornetto: A Combinatorial Lexical Semantic Database for Dutch*. Springer, 03/2013 2013.

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.

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. TectoMT: Highly Modular MT System with Tectogrammatics Used As Transfer Layer. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 167–170, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-932432-09-1. URL http://dl.acm.org/citation.cfm?id=1626394.1626419.