Constrained Recombination in an Example-based Machine Translation System

Monica Gavrila
University of Hamburg,
Faculty of Mathematics, Informatics and Natural Sciences
Vogt-Koelln Str. 30, 22527, Hamburg, Germany
gavrila@informatik.uni-hamburg.de

Abstract

Constraints in natural language processing play an important role. In this paper we show which impact (word-order) constraints have on the translation results, when they are applied in the recombination step of a linear EBMT system. Both the baseline EBMT system and the constrained one are implemented during this research. In the experiments we use two language-pairs (Romanian-English and Romanian-German), in both directions of translations. In these language constellations, Romanian, an inflected language with Latin root, is considered under-resourced. This aspect makes the process of translation even more challenging.

1 Introduction

Machine translation (**MT**), one of the most challenging domains in Natural Language Processing (**NLP**), plays an important role in ensuring global communication. Documents in various domains need to be translated in a large combination of language-pairs. As quite often it is hard to find the right human translators, with the right domain- and language-knowledge, MT can be considered, at least for these cases, a solution.

Less spoken languages have to overcome a major gap in language resources and tools, which all ensure the development of a good MT-system. Even more, some of these under-resourced languages are highly inflected, with a more complicated grammar and often having linguistic phenomena which have been not encountered in previous language combinations. On the other side, exactly for these languages, human translators are few or missing, so MT-systems are highly required.

Based mainly on the existence of a parallel corpus, which does not necessary have to include a large number of examples[1], example-based machine translation (**EBMT**) seems to be a solution for under-resourced languages. This MT approach, which has its start in Nagao's work (Nagao, 1984), is essentially translation by analogy. The basic premise is that, if a previously translated sentence occurs again, the same translation is likely to be correct again.

Constraints in natural language processing play an important role, such as in constraint-based grammars. Constraints usually restrict the possible values that a variable (or a feature) may take with respect to certain rules. In MT, they have been used for example in the SMT approach: (Canisius and van den Bosch, 2009), (Cao and Sumita, 2010).

In this paper we explore how (word-order) constraints can be used in a linear EBMT system. As we employ an under-resourced language (i.e. Romanian), we keep the systems as resource-free as possible. The algorithms are mainly based on surface forms and corpus statistics. That is why our EBMT systems borrow ideas only from the linear and template-based EBMT approaches.

We investigate two language pairs: Romanian-English and Romanian-German, in both directions of translation. The under-resourced language we consider in this work is Romanian, as when starting this work not sufficient linguistic resources were publicly available, or, when available, comparing with the other two languages, they were

[1] In contrast to statistical MT (**SMT**).

Mikel L. Forcada, Heidi Depraetere, Vincent Vandeghinste (eds.)
Proceedings of the 15th Conference of the European Association for Machine Translation, p. 193–200
Leuven, Belgium, May 2011

under-developed or not sufficiently tested. Furthermore, there was also no real possibility of choosing among several resources, as, when available, only one resource was at hand. The use of the German - Romanian language-pair raises interesting questions, as most of the example-based translation systems consider English as a source/target language (**SL/TL**), which has a simpler syntax and morphology. Romanian and German, both inflected languages, present language specific characteristics (morphological and syntactical), that makes the process of translation even more challenging.

After this short introduction, the following two sections will present both EBMT systems we implemented: the baseline EBMT system $Lin-EBMT$, and the constrained system $Lin-EBMT^{REC+}$. In section 4 we will describe the data used and the translation results. The results obtained by $Lin-EBMT$ are compared with the ones provided by different constraint settings applied in $Lin-EBMT^{REC+}$. In all the experiments the same training and test data are used. The paper will end with conclusions and further work.

2 $Lin-EBMT$, the Baseline EBMT System

In this section we describe $Lin-EBMT$, the baseline EBMT system implemented during the research. $Lin-EBMT$ is a linear EBMT system, in the sense of system classification found in (McTait, 2001). It is based on surface-forms and uses no linguistic resources, with the exception of the parallel corpus. It contains all the three steps of an EBMT system[2]: matching, alignment and recombination.

Before starting the translation, training and test data are tokenized and lowercased. In order to reduce the search space in the matching process, we use a word index. This approach has already been encountered in the literature, for example in (Sumita and Iida, 1991). The matching procedure is run only after the search space size is decreased. If during the matching procedure the test sentence is found in the corpus, its translation represents the output. Otherwise, the translation steps described in the following subsections are performed.

Matching the Input

The matching procedure is a linguistic light approach, focusing in finding common substrings. As, the longer the common subsequence, the lower the probability of boundary friction problems, the longest common subsequence (**LCS**) is considered. The procedure is based on the Longest Common Subsequence Similarity (**LCSS**) measure we implemented using a dynamic programming algorithm similar to the one found in (Bergroth et al., 2000). The initial LCS character-based algorithm is transformed into a token[3]-based one. A penalty $P = 0.01$ is introduced for each token-gap between the input and the matched sentence. This way it is chosen the sentence that covers the input with less token-gaps. Therefore, the chance to have a minimum number of sequences that should be recombined to form the output increases. This approach could decrease the appearance of boundary friction and word-order problems.

The matching score is calculated as follows: given the input I and a sentence S from the example database, LCSS is calculated as:

$$LCSS(I,S) = LCSS_T(I,S) - P * noTG, \quad (1)$$

where

$$LCSS_T(I,S) = \frac{Length(LCS(I,S))}{Length(I)}, \quad (2)$$

$LCS(I,S)$ is the LCS between I and S, $Length(x)$ is the number of tokens of a string x and $noTG$ is the number of token-gaps of the $LCSS(I,S)$ when compared with I.

For example, considering the sentences
Input s1 = "*Saving names and phone numbers (Add name)*"
Sentence in the corpus s2 = "*Erasing names and numbers*"
The longest common subsequence $LCS(s1,s2)$ is "*names and numbers*"

Given the input and the example database, the matching procedure gives as output the sentences that best cover the input. The algorithm tries to match the input with an entry in the corpus, and in case this is not possible, to match parts of the input with (parts of) the sentences in the corpus. The matching algorithm is recursive and follows the steps enumerated below:

1. Find the sentence in the corpus that best match the input, by using the similarity measure previously described. Keep it as part of

[2]The steps of an EBMT system – matching, alignment and recombination – are firstly described in (Nagao, 1984) and specifically presented under these names in (Somers, 1999).

[3]A token can be a word-form, a number, a punctuation sign, etc.

the solution. In this step only one maximum value for $LCSS(I, S)$ is chosen.

2. If the input is not fully covered, eliminate what has been already found from the input and for the rest return to step 1. Else, stop the matching procedure: the result is found.

Output Generation - Alignment and Recombination

After matching the input on examples, two further steps are required: alignment and recombination.

The alignment information is extracted from the GIZA++[4] output, when considering the information for target-source language direction. The choice of only one language direction is motivated by the need to avoid conflicts between the matching step and the alignments extracted.

The alignment procedure considers the sentences given as output by the matching procedure and their translations. From these sentences all the GIZA++ alignments are extracted[5], but only the longest target language aligned subsequences are used further in the recombination step. For example, given the extracted LCS "*technical regulations standards*" and the following alignments: "*technical - tehnice*" (position 8 in TL), "*regulations - reglementările*" (position 7 in TL) and "*standards - standarde*" (position 23 in TL), the sequences "*reglementările tehnice*" and "*standarde*" are used in the recombination step (language-pair English-Romanian).

The recombination step has as input the "*the bag of TL sequences*" given as output by the alignment and as result the translation. It is based on the monolingual distribution of bi-grams and on the **recombination matrix** $A = (a_{i,j})_{1 \leq i,j \leq n}$, which we define as follows: If the outcome of the alignment is n word-sequences $\{sequence_1, sequence_2, ..., sequence_n\}$ which form the output and are not necessarily different, with $sequence_i = w_{i_1} w_{i_2} ... w_{i_{last}}$, then A is a square matrix of order n that is defined as follows:

$$A = \begin{cases} -3, & \text{if } i = j; \\ -2, & \text{if } i <> j, \\ & w_{i_{last}} w_{j_1} \text{ is} \\ & \text{not in the} \\ & \text{corpus;} \\ \frac{2 * count(w_{i_{last}} w_{j_1})}{count(w_{i_{last}}) + count(w_{j_1})}, & \text{else.} \end{cases} \quad (3)$$

where $count(s)$, when s is a token, represents the number of the appearances of s in the corpus. The bi-grams considered are formed from the last word of the sequence $sequence_i$ - $w_{i_{last}}$ and the first word of $sequence_j$ - w_{j_1}. The value for the case "$i <> j$ and $count(w_{i_{last}} w_{j_1}) <> 0$" is computed using the Dice coefficient.

The idea of representing the information in a matrix is motivated by the "*similarity matrix*" found in (Kit et al., 2002). However, the way in which we define the recombination matrix and obtaine the output, differentiate our work from the previous approach.

The recombination algorithm is based on finding the maximum value $a_{i,j}$, '*combining*' $sequence_i$ and $sequence_j$, and deleting all the values from the matrix corresponding to $sequence_j$ (line and column j). When $sequence_i$ and $sequence_j$ are combined, they are concatenated and the matrix values for the new element $sequence_i sequence_j$ are the ones which previously corresponded to $sequence_j$.

Given a certain corpus, the maximum value for $a_{i,j}$ means that the probability that $sequence_j$ follows $sequence_i$ is the highest. This happens as the probability that w_{j_1} follows $w_{i_{last}}$ is the highest. Data sparseness has a direct influence on the results. The whole recombination process starts with the first maximum value found in the initial matrix and it continues until the order of the matrix becomes one and the output is obtained.

3 Imposing Constraints on Recombination

In the previous section we showed how $Lin - EBMT$ is implemented. In the recombination step it makes no use of the information directly extracted by the matching step, as it employs only the '*bag of TL word sequences*' – the output of the alignment. From these word sequences, the output is formed by considering only 2-gram information

[4]GIZA++ is a toolkit that is used to train the 1-5 IBM models and an HMM word alignment model. More on http://code.google.com/p/giza-pp/.

[5]All alignments need to be extracted, as they are further used in the template-generation in $Lin - EBMT^{REC+}$.

and the recombination matrix. This way the information provided by the matching (SL sentences and their translations) is lost, although helpful data for deciding the word order in the recombination step is still present.

In the implementation of the extended version of $Lin - EBMT$ (i.e. $Lin - EBMT^{REC+}$) ideas from the template-based EBMT approach are incorporated in the recombination step. The previous two steps[6] remain unchanged.

The idea of mixing the linear approach with the template-based one is also found in previous published works. In (Sumita, 2001), when combining these two EBMT approaches, in contrast to our implementation, a pure template-based approach is used for the recombination. In our approach the values in the recombination matrix are constrained by information extracted from templates. A template follows the definition from (McTait, 2001), with the difference that the alignments of the text fragments and variables are contained in only one set of alignments (A_{all}) and not in two. Furthermore, there is no connection between the number of text fragments and the one of variables. A template contains an SL and a TL side.

During the translation process the template extraction algorithm is applied for each test sentence in the test data set after the alignment. It is a runtime procedure in the translation process. It has as input the sentence to be translated, the matched sentences and their translations, the correspondent longest common subsequences and GIZA++ alignments. A template is extracted for each matched sentence.

The template extraction algorithm has two phases: a monolingual and a bilingual one. The monolingual phase is considered only for the source language, in contrast to other template extraction algorithms presented in the literature, for example in (McTait, 2001). Similar ideas appear also in (Caseli et al., 2006). Before starting the monolingual phase, which is based on the information provided by the $LCS(I, S_i)$, the alignment information $A_{all\,i}$ is extended, so that each aligned sequence is marked either as text fragment ($TEXT$) or as variable (VAR).

The Monolingual Phase:
The monolingual phase of the algorithm has as output the SL side of the template. The common tokens between I and the SL matched sentence S_i (i.e. $LCS(I, S_i)$) are considered as text fragments in the SL side of the template. All the other tokens from S_i represent variables.

The Bilingual Phase:
Given the SL side of the template, the translation T_i, and the alignment information $A_{all\,i}$, the TL side of the template can be obtained. The TL sequences aligned to the SL text fragments represent the TL text fragments. The rest of the TL tokens are considered variables. The alignments between the SL/TL variables and text fragments are given by the information provided by GIZA++.

All aligned SL/TL text fragments and SL/TL variables are attached the same identification number. In case no alignment information for some variables is found, these variables are of the generic type $NOALIGNnumber$. The variables from TL, which are not aligned, are of the type $NOALIGN0$. When SL text fragments are not aligned, no correspondent alignment number is found in the TL side.

After the template is obtained, it is reduced, i.e. if on both SL and TL sides there is the same variable sequence $VAR_iVAR_{i+1}...VAR_{j-1}VAR_j$, with $i \leq j$ and with variables one after another in the same order, this sequence is reduced to a one variable VAR_{ij} on both SL and TL sides.

The output of the whole algorithm for template extraction is the set of all reduced templates. This is used later in generating constraints in the recombination step.

The recombination step in $Lin-EBMT^{REC+}$ builds the output almost in the same way as in $Lin-EBMT$. Differences appear in the values of the recombination matrix and in the way the maximum value is searched.

From the extracted templates, word-order rules are determined and a set $C = \{(w_i, w_j)\}$ of constraints is built. C contains no duplications. A constraint (w_i, w_j) imposes that the words w_i and w_j can not appear one after another in the output as the sequence w_iw_j. Therefore, the value in the recombination matrix corresponding to the entry w_iw_j is set to **-2**. This way the possibility of choosing this combination as a maximum is reduced.

We considered three types of constraints:

1. **The First-Word-Constraint (C.1)**, which refers to the first word of the output:
   If a word $w_{TL_{first}}$
   - *is found as a first word in the TL side of a template, and*

[6]The matching and alignment algorithms.

- *is aligned to the first word $w_{SL_{first}}$ on the SL side*[7]*,*

then it is considered as the first word of the output and no other words or word-sequences can precede it. This means that for all TL words w_i provided by the alignment, the constraint $(w_i, w_{TL_{first}})$ is added to the set of constraints C.[8]

2. **TLSide-Template-Constraint (C.2)**, which is deduced only from the TL side of each of the templates extracted:
If in a TL side of a template the words w_i and w_j appear in the sequence $w_i[...]w_j$, then the sequence w_jw_i is not allowed in the output formation. Therefore the constraint (w_j, w_i) is added to the set of constraints C.

3. **Whole-Template-Constraint (C.3)**, which is extracted considering each of the templates, together with the input sentence, and the alignment information[9]:
Before defining this type of constraints, some observations need to be made. Given the input sentence $I = \{t_{SL_1}...t_{SL_n}\}$ and the alignment information $t_{SL_i} \leftrightarrow t_{TL_i}$, where $1 \leq i \leq n$, it is considered that:
   (a) In case t_{SL_i} is not aligned on the TL side, t_{TL_i} has a generic value "*NOALIGNMENT*", value which will be ignored in further steps;
   (b) If t_{SL_i} is an out-of-vocabulary word (**OOV-word**), then it is aligned to itself. That is $t_{TL_i} = t_{SL_i}$.

   If on the SL-side of a template, before a text fragment t_{SL_k}, the 'same'[10] *variables/text fragments appear as on the TL-side of the same template before the aligned TL text fragment t_{TL_k} , then the TL aligned sequences $t_{TL_p}...t_{TL_q}$ corresponding to the sequences $t_{SL_p}...t_{SL_q}$ in the input, which are before t_{SL_k}, appear in the output also before the t_{TL_k}. That means that constraints of the form (t_{TL_k}, t_{TL_j}), $p \leq j \leq q$, are added to the set of constraints C.*

We chose only constraints which can be easily extracted from the templates, without using extra linguistic resources. In a broader context more types of constraints can be included.

For the recombination step we define the **constrained recombination matrix**, which can be seen as an extended version of the previous recombination matrix (formula 3). Given the outcome of the alignment – n word-sequences $\{sequence_1$, $sequence_2$, ..., $sequence_n\}$ that form the output and which are not necessarily different, with $sequence_i = w_{i_1}w_{i_2}...w_{i_{last}}$– and a set of constraints $C = \{(w_{i_q}, w_{i_r})\}$, with $1 \leq i_q, i_r \leq n$, then $A = (a_{i,j})_{1 \leq i,j \leq n}$ is a square matrix of order n that is defined as:

$$A = \begin{cases} -3, & \text{if } i = j; \\ -2, & \text{if } i <> j, \\ & w_{i_{last}}w_{j_1} \text{ is not in} \\ & \text{the corpus or} \\ & (w_{i_{last}}w_{j_1}) \in C; \\ \frac{2*count(w_{i_{last}}w_{j_1})}{count(w_{i_{last}})+count(w_{j_1})}, & \text{else.} \end{cases} \quad (4)$$

where $count(s)$, when s is a token, represents the number of the appearances of s in the corpus. The bi-grams considered are formed as in the previous definition (see formula 3). Also the value for the case "$i <> j$ and $count(w_{i_{last}}w_{j_1}) <> 0$ and $(w_{i_{last}}w_{j_1}) \notin C$" is computed using the Dice coefficient.

As in $Lin - EBMT$, the recombination algorithm of $Lin - EBMT^{REC+}$ is based on finding the maximum value $a_{i,j}$ in the constrained recombination matrix

The algorithm follows the same steps as in $Lin - EBMT$[11], when no **C.1** constraints can be applied. When **C.1** constrains can be applied the maximum value is not searched in the whole matrix, but on a specific row: given that the first word is w_{FIRST} in $sequence_p$, the first maximum value in the matrix is searched as $a_{p,i}$. The algorithm continues considering the previous word found and incorporated in the output.

For some of our experiments we made the difference between the case '$w_{i_{last}}w_{j_1}$ *is not in the corpus*' – i.e. no information found in the LM –

[7] This means it is the translation of the first word of the input $w_{SL_{first}}$.

[8] When extracting these types of constraints, information might be derived, which is not used later. An improvement of the algorithm could be made by considering only the TL sequences which form the output and not all possible words.

[9] The alignment refers to the corresponding TL tokens (token-sequences) of the SL tokens (token-sequences) in the input.

[10] In this context, the '*same*' means that the variables and /or text fragments have the same alignment number.

[11] See the previous section.

and the case '$(w_{i_{last}}w_{j_1}) \in C$' – i.e. there is a constraint for these specific words. Therefore, we changed the definition of the constrained recombination matrix:

$$A = \begin{cases} -3, & \text{if } i = j; \\ -1, & \text{if } i <> j, \\ & w_{i_{last}}w_{j_1} \text{ is not in} \\ & \text{the corpus;} \\ -2, & (w_{i_{last}}w_{j_1}) \in C; \\ \frac{2*count(w_{i_{last}}w_{j_1})}{count(w_{i_{last}})+count(w_{j_1})}, & \text{else.} \end{cases} \quad (5)$$

In further work the definition of the matrix could be more refined, for example by using weights on the constraints.

4 Evaluation

In this section, before the evaluation results will be presented, the training and test data used for the experiments are described.

4.1 Data Description

We chose for our experiments a parallel corpus, considering four languages (Romanian, German, English and Russian), called **RoGER**. The corpus was manually aligned at sentence level. Moreover, its translations were manually verified. It is a domain-restricted corpus, as the text represents a users' manual of an electronic device. The text is preprocessed, by replacing numbers, web pages, etc. with '*meta-notions*', for example numbers with *NUM*. More information about RoGER can be found in (Gavrila and Elita, 2006).

The small size, i.e. 2333 sentences, is compensated by the correctness of the translations and of the alignments. 133 sentences were randomly extracted as the test data set; the rest of 2200 sentences remain as the training data. Some statistical information about the corpus is presented in Table 1.

4.2 Results

The obtained translations were evaluated using three (3) automatic evaluation metrics: BLEU (Papineni et al., 2002), NIST (Doddington, 2002) and TER (Snover et al., 2006). The choice of the metrics is motivated by the available resources and, for comparison reason, by the results reported in the literature. Due to lack of data and further translation possibilities, we consider the comparison with only one reference translation.

Data SL	No. of words	Vocabulary	Average sentence length
English-Romanian			
Training	27889	2367	12.68
Test	1613	522	12.13
Romanian-English, Romanian-German			
Training	28946	3349	13.16
Test	1649	659	12.40
German-Romanian			
Training	28361	3230	12.89
Test	1657	604	12.46

Table 1: RoGER Statistics.

The evaluation of $Lin - EBMT^{REC+}$, when changing the combination of constraints, is presented in Table 2 for Romanian-English and in Table 3 for German-Romanian, for both directions of translation. All possible combinations of constraints and definitions of the constrained recombination matrix are tested.

System	BLEU	NIST	TER
English – Romanian			
$Lin - EBMT$	0.2997	5.4093	0.6046
C. 1	0.3067	5.5768	0.5930
C. 2	0.3042	5.4187	0.5991
C. 3	0.3083	**5.5836**	0.5906
C. 1+2	0.3062	5.5353	0.5930
C. 1+3	0.3083	**5.5836**	0.5906
C. 2+3	0.3073	5.5638	0.5882
C. 1+2+3	0.3073	5.5638	0.5882
C. 1+2+3 1:2	**0.3085**	5.5322	**0.5864**
Romanian – English			
$Lin - EBMT$	0.3597	6.0586	0.5065
C. 1	0.3695	6.2694	0.5034
C. 2	0.3711	6.1625	**0.4984**
C. 3	0.3633	6.2415	0.5108
C. 1+2	**0.3712**	6.2879	0.5009
C. 1+3	0.3632	6.2355	0.5114
C. 2+3	0.3656	6.2620	0.5083
C. 1+2+3	0.3656	6.2620	0.5083
C. 1+2+3 1:2	0.3668	**6.2991**	0.5077

Table 2: Evaluation Results for $Lin - EBMT^{REC+}$ – English-Romanian, when changing the constraints (C=constraint).

In both tables 2 and 3, for the case **C. 1+2+3 1:2** the definition of the constrained recombination matrix presented in formula 5 is used. For the rest of the experiments, we employ the definition shown in formula 4. The notation '*C. number*' means that only the constraint of type '*number*' is used. In '$C.number_1 + number_2$' two constraints are used, and they are of type $number_1$ and $number_2$, respectively.

The evaluation results show small improvements for (almost all) the cases when constraints are used. The differences between the $Lin -$

System	BLEU	NIST	TER
German – Romanian			
$Lin - EBMT$	0.2643	4.5589	0.6428
C. 1	0.2658	**4.6935**	0.6422
C. 2	**0.2682**	4.6074	0.6409
C. 3	0.2627	4.6757	0.6422
C. 1+2	0.2654	4.6745	0.6428
C. 1+3	0.2627	4.6757	0.6422
C. 2+3	0.2633	4.6807	0.6422
C. 1+2+3	0.2633	4.6807	0.6422
C. 1+2+3 1:2	0.2646	4.6559	**0.6361**
Romanian – German			
$Lin - EBMT$	0.2867	4.9792	0.6795
C. 1	0.2842	5.0664	0.6716
C. 2	0.2857	5.0253	0.6789
C. 3	0.2891	5.0622	0.6716
C. 1+2	0.2836	5.0591	**0.6698**
C. 1+3	0.2891	5.0622	0.6716
C. 2+3	0.2875	5.0593	0.6722
C. 1+2+3	0.2875	5.0593	0.6722
C. 1+2+3 1:2	**0.2894**	**5.0770**	0.6722

Table 3: Evaluation Results for $Lin - EBMT^{REC+}$ – German-Romanian, when changing the constraints (C=constraint).

$EBMT$ scores and the scores obtained by $Lin - EBMT^{REC+}$ are higher for English-Romanian (both directions of translations), than for German-Romanian (both directions). Analyzing the results it can be seen that best results (bold-face values in the tables 2 and 3) are encountered for different combinations of constraints. However, the combination **C. 1+2+3 1:2** gives best results in 50% of the cases, when all three evaluation scores and all combinations of languages are considered. Therefore, it can be considered the "*winner*".

A visual representation of all results is shown in Figure 1.

Our results of the EBMT systems for Romanian English, in both directions of translations, are comparable[12] with the ones presented in (Irimia, 2009). The system in (Irimia, 2009) is using extra linguistic resources and as corpus the JRC-Acquis corpus[13]. The maximum BLEU scores reported here were 0.3088 and 0.3689, for English-Romanian and Romanian-English, respectively. To our knowledge no results were reported for EBMT systems, for German-Romanian (in both directions).

Concerning the time of translation for $Lin - EBMT^{REC+}$, on the whole, it required less time than $Lin - EBMT$. although extra-time is needed for the extraction of the constraints. This happens due to the changes in the recombination matrix.

[12] A 1:1 comparison is excluded, as different type of data is used.

[13] http://wt.jrc.it/It/Acquis/

5 Conclusions

In this paper we presented $Lin - EBMT^{REC+}$, an extension of the pure linear EBMT system $Lin - EBMT$. $Lin - EBMT^{REC+}$ combines ideas from linear EBMT systems and template-based ones. When compared with $Lin - EBMT$, changes appear only in the recombination step by adding word-order constraints. The other two EBMT steps – matching and alignment – remain unchanged. Adding extra word-order information in the recombination, as expected, led to an improvement in the translation results. As the changes in the recombination matrix might have a seldom impact on the results – due to the corpus, due to the fact that only one solution is taken, etc. – this improvement was not very big. As further work we plan testing how further constraints could influence the translation results and how the systems react to a different type of data, e.g. the JRC-Acquis corpus. A manual analysis of the results would show how exactly the results are influenced by the constraints and how the systems react to the degree of inflection of the languages involved. Additionally, testing n-grams of several lengths could be of interest.

References

Bergroth, Lasse, Harri Hakonen and Timo Raita. 2000 A survey of longest common subsequence algorithms *Proceedings of the Seventh International Symposium on String Processing and Information Retrieval - SPIRE 2000*, 39–48 A Curuna, Spain, September, ISBN: 0-7695-0746-8.

Canisius, Sander and Antal van den Bosch. 2009 A Constraint Satisfaction Approach to Machine Translation *Proceedings of the 13th Annual Conference of the EAMT*. 182–189 Barcelona, May.

Cao, Hailong and Eiichiro Sumita. 2010 Filtering Syntactic Constraints for Statistical Machine Translation *Proceedings of the ACL 2010 Conference Short Papers*, 17–21 Uppsala, Sweden, July, 11-16.

Caseli, Helena and Maria das Graças V. Nunes and Mikel L. Forcada. 2006 Automatic induction of bilingual resources from aligned parallel corpora: application to shallow-transfer machine translation *Machine Translation*, Volume 20, Number 4, 227–245 December, ISSN: 0922-6567, Kluwer Academic Publishers, Hingham, MA, USA.

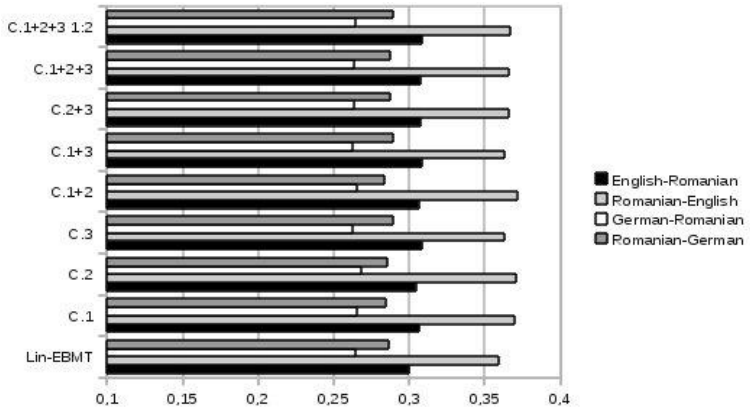

Figure 1: The Influence of Constraints and Constraint Settings on $Lin - EBMT^{REC+}$ (BLEU scores).

Doddington, George. 2002 Automatic evaluation of machine translation quality using n-gram co-occurrence statistics *Proceedings of the second international conference on Human Language Technology Research*, 138–145, San Francisco, CA, USA Morgan Kaufmann Publishers Inc., San Diego, California

Gavrila, Monica and Natalia Elita. 2006 Roger - un corpus paralel aliniat *In Resurse Lingvistice şi Instrumente pentru Prelucrarea Limbii Române Workshop Proceedings*, 63–67 December, Publisher: Ed. Univ. Alexandru Ioan Cuza, ISBN: 978-973-703-208-9.

Irimia, Elena. 2009 EBMT Experiments for the English-Romanian Language Pair *Proceedings of the Recent Advances in Intelligent Information Systems*, 91–102 ISBN 978-83-60434-59-8.

Kit, Chunyu Kit, Haihua Pan and Jonathan J. Webster 2002 Example-Based Machine Translation: A New Paradigm *Translation and Information Technology*, 57–78 Chinese U of HK Press.

McTait, Kevin. 2002 *Translation Pattern Extraction and Recombination for Example-Based Machine Translation* PhD Thesis, Centre for Computational Linguistics, Department of Language Engineering, PhD Thesis, UMIST.

Nagao, Makoto. 1984 A Framework of a Mechanical Translation between Japanese and English by Analogy Principle *Proceedings of the international NATO symposium on Artificial and human intelligence*, 173–180 New York, NY, USA, Elsevier North-Holland, Inc., ISBN 0-444-86545-4, Lyon, France.

Papineni, Kishore, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002 BLEU: a method for automatic evaluation of machine translation *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Session: Machine translation and evaluation*, 311–318 Philadelphia, Pennsylvania, Publisher: Association for Computational Linguistics Morristown, NJ, USA.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006 A Study of Translation Edit Rate with Targeted Human Annotation *Proceedings of Association for Machine Translation in the Americas*, 223–231 August

Somers, Harold. 1999 Review Article: Example-based Machine Translation *Machine Translation*, Volume 14, Number 2, 113–157, Publi. Springer Netherlands

Sumita, Eiichiro and Hitoshi Iida. 1991 Experiments and prospects of Example-Based Machine Translation *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, 185–192 Morristown, NJ, USA, Association for Computational Linguistics, Berkeley, California.

Sumita, Eiichiro. 2001 Example-based machine translation using DP-matching between word sequences *Proceedings of the workshop on Data-driven methods in machine translation*, 1–8 Morristown, NJ, USA, Publisher: Association for Computational Linguistics.