Recent Advances in Intelligent Information Systems
ISBN 978-83-60434-59-8, pages 91–102

EBMT Experiments for the English-Romanian Language Pair

Elena Irimia

Institute of Artificial Intelligence, Romanian Academy, Bucharest, Romania

Abstract

The paper describes an attempt of constructing an example-based machine translation application for the English-Romanian language pair by making the best possible use of the resources and the pre-processing tools available in our working group. Therefore, the application is not necessarily innovative in the general field of EBMT, but an experiment of what can be done with specific resources and tools, for a specific pair of languages, in a research direction of machine translation which is not very much explored.

Keywords: example-based machine translation

1 Introduction

The purpose of this project was experimenting with one of the most important assumptions in the empirical corpus-based paradigm in Machine Translation that states that generalization (of different types and levels) of the data is beneficial as it provides more information in less memory space. Additionally, by these generalizations, data sparseness is diminished and the linguistic coverage is increased. One of the handiest generalization, specifically appropriate for highly inflected languages, is the lemmatization of the data. Obviously, there is a price to be paid and this is represented by the morpho-lexical properties transfer between source and target translation equivalents, followed by a generation of the inflected form in the target language: those processes are, of course, subject to error. We wanted to evaluate how this gain and loss balance in the case of an EBMT application, more precisely how (if this is the case) the translation results are improved if the whole process (the creation of the database, the matching and the recombination) is centered on the lemmas, and not on the word forms.

2 Linguistic resources and pre-processing applications

2.1 JRC-Acquis

As fundamental resource for our EBMT application (since we are in the corpus-based paradigm) we chose the multilingual parallel corpus JRC-Acquis (Steinberger *et al.*, 2006). We considered this specific corpus to be very appropriate for it is:

- relatively homogeneous: dedicated to a specific domain - legislation of the European Union - but covering a variety of fields that fall under this legislation;
- consistent: at least in theory, any juridical expression in the source language must always have the same translation in the target language and this translation must be validated by the law community;
- JRC-Acquis is a dynamical collection of juridical documents extracted from Acquis Communautaire (AC), which represents the European Union's total body of laws, applicable to all the member countries. AC, and implicitly JRQ-Acquis, is constantly augmented with new documents as the European Union is expanding and the member states align their legislation to the community law.

JRC-Acquis is available in 22 of the 23 official languages of the EU (as the Irish translations are not ready yet) and represents the biggest parallel corpus existent at this moment, taking into account both its size and the number of the covered languages. The number of documents is around 8000, with variants for every language (each document has a unique numerical identifier called CELEX, which helps at finding the same document in the various languages.) The documents were converted in the XML format and uniformly encoded to UTF-8. In the form used by our application, the corpus contains only the Romanian-English language pair and it is the result of consecutive pre-processing actions: sentence splitting and alignment, word-level splitting, part-of-speech tagging, lemmatisation, lexical alignment, chunking and dependency linking analysis. The tools used to perform these operations are briefly described in the following sections.

2.2 The pre-processing stage

In the form used by our application, the corpus contains only the Romanian-English language pair and it is the result of consecutive pre-processing actions:

- **sentence splitting:** done with TTL, (Ion);it uses a set of regular expressions for identifying the markers for sentence ends; to disambiguate the period marker, the application uses a list of usual abbreviations for English and Romanian (the period can indicate the end of a sentence, an abbreviation or both);
- **word-level segmentation:** done with TTL; the usual word end marker in English and Romanian is the space character; but there are situation when the space does not mark an word end (in both of the languages there are idiomatic expressions which meaning is not decomposable and that represent a single lexical unit) and some other where the marker is absent (for example in contractions like "can't" and "nu's"- Romanian for "aren't"); to cope with this particular cases, TTL uses lists of idiomatic expressions and lists of suffixes and prefixes;
- **part-of-speech tagging:** the TTL tagger is a reimplementation of the statistical HMM tagger described in (Brants, 2000) with some improvements:
  – for the words unknown to the tagger (that were not found in the training corpus), TTL introduces some heuristics to propose a tag;

  - uses a suffix analysis to guess the tag, but only for the open class category of words (nouns, verbs, adjective, adverbs);
  - gives a proper noun tag to all the words that start with majuscule letter and are not positioned at the beginning of the sentence;

- **lemmatisation** is the algorithmic process of determining the lemma (the canonical or dictionary form) for a given word. The TTL strategy uses a lexicon[1] of inflected forms to automatically extract a lemmatisation model for MSDs. The lemmatisation algorithm receives an entry of the structure $(form\, w, MSD\, t)$ and follows the steps: 1) it looks for the pair in the lexicon and extracts the lemma associated to the identified entry; 2) if the pair is not to be found in the lexicon, TTL applies a set of lemmatisation rules for t (automatically extracted from the lexicon) and produces a list of candidate lemmas that is ordered in accordance with the probabilities of the lemmatisation model for the same t; the lemma with the biggest probability is selected;
- **lexical alignment**[2] done with YAWA (Yet Another (simple) Word Aligner), (Tufiş *et al.*, 2006); YAWA's alignment process is based on the previous preprocessing steps (segmentation, POS-tagging, lemmatisation) and it also uses an additional phase of meta-category annotation: a meta-category is a POS tag class identified by a number, that allow the inter-category alignment. For example, if the Vmg and Ncfs-n tags are in the same meta-category, a gerund verb like *"thinking"* and a noun like *"gânduri"* (*"thoughts"*) can be aligned. The alignment is made from Romanian to English and has four stages, with every stage producing an alignment skeleton to which the alignments in the next stage are added. The process is directed to an increased recall of the global alignment with as less diminishing of the precision as possible. After his final stage, the recall of YAWA reaches 74.83% and the precision is 88.8%.
- **dependency linking analysis:** it is done by LexPar (see (Ion)), an application based on Yuret's Lexical Attraction Model (LAM, (Yuret)). In his vision, the lexical attraction is a measure of the combining affinity between two words in the same sentence. If two words are "lexically attracted" to one another in a sentence, the probability for them to combine in other sentences is significant. Therefore, two or more words that manifest lexical attraction, together with their translations in other languages, represent good translation examples. The formalization for the lexical attraction between two words is given by a probabilistic model in which every word of a sentence is probabilistically dependent only of his regent. The mutual information between two words is the measure for the syntactic dependency of those words. LexPar is an extension of the Yuret's algorithm running on tagged and lemmatised texts, which assures a better linguistic coverage and better estimation for the model's parameters. It also uses a language dependant syntactic filter to reject some of the links,

[1]A very large word-form lexicon containing more than 1,217,296 manually validated lexical entries for Romanian (tbl.wordform.rov60) and 135,500 manually validated lexical entries for English (tbl.wordform.en.v27), each containing a word form occurrence, its lemma and the appropriate MSD tag.

[2]For a language pair (s, t), the lexical alignment of two sentences s_s and s_t is a an array A of correspondences $w_s^i \leftrightarrow w_t^j$, where i and j are the positions of the words in the sentences. For processing needs, a correspondence is represented only like a list of pairs (i, j).

thus accelerating the convergence of the training process to the LAM that approximates the dependency structure of that language. LexPar's recall gets to 60.70% (better than the Yuret's analyzer) and the precision attains 53.69%.

3 The EBMT application's architecture

The application is structured in two modules:

1. ExTract: the module for the translation examples extraction;
2. MatchRec: the matching and recombining module, which implements the mechanism of translating a new sentence from a language to another

3.1 ExTract and the example database

Traditionally, the EBMT systems use different levels of granularity for their example database. Thus, the translation units can be:

- sentences: experimental systems or systems dedicated to specific domain can find this solution appealing, since the sentence boundaries are easy to identify and the sentences are quite simple; for real corpora systems, the sentence complexity brings new problems, compensated by complicated mechanisms of fuzzy matching and recombination.
- chunk-bounded n-grams(): (Kupiec), (Kumano and Hirakawa, 1994), (Smadja *et al.*, 1996); this kind of translation examples are not able to catch the structural dependencies in a sentence (e.g., the accord between a verb and a noun phrase in the subject position);
- dependency-linked n-grams:in a comparative study of a bounded-length n-grams model, a chunk-based model and a dependency-based model, (Yamamoto and Matsumoto, 2003) concludes that the last two models have clearly superior results in translation, but they are complementary, since the precision of the chunk-based model is very good while the dependency-based model offers the possibility of extracting long and sometimes non-successive examples.

We opted to implement the assumption in (Cranias *et al.*, 1994) that the EBMT potential should rely on exploiting text fragments shorter than the sentence and we looked for "the best covering" of an input sentence, in other words we tried to compute a decomposition of the sentence in coherent segments which are to be found in the database.

For the construction of our example database, we wanted to start from Yuret's lexical attraction described earlier in section 2.2. We already introduced the idea that this attraction increases the association probability between two or more words in other contexts than that in which the links between those words were originally identified. For that reason, the word sequence is a translation example better than a simple bounded-length n-gram. The lexical model introduced by Yuret is also able to catch the connections between words in the same syntactic phrase (or chunk) and the dependencies between words across the chunk boundary. We will use the same lexical attraction concept in the translation mechanism: before the matching stage, we need to decompose the sentence to be translated in subsequences that will be matched against the database and the decomposing process is guided by the links that express the lexical attraction between words.

For the extraction of translation examples, it is not enough to design a strategy for dividing the sentences in word sequences (phrases), but we also need to set correspondences between the phrases of a sentence in the source language and their translations in the equivalent sentence in the other language (phrasal alignment). For this purpose, we will use the lexical alignments computed by YAWA (section 2).

As resulted from all the processing steps described in the previous section, every translation unit in the corpus is annotated at the word level with the following attributes:

- *lemma:* the value of this attribute is a string that represents the word's lemma as identified by TTL;
- *ana:* the value of this attribute is a string that contains the meta-category and the POS-tag of the word, separated by comma;
- *head:* this is an optional attribute and its value is a number representing the position in the sentence of a word that is lexically attracted by the word with the head attribute.

Additionally, YAWA produces a file encoding the lexical alignments for every translation unit (see footnote 2 in this paper).

In Figure 1, one can observe that links identified by LexPar have the tendency to group by imbrications (see the link structure corresponding to "made in the national currency") or to decompose the sentence by chaining. These properties suggest more possible decompositions for the same sentence, and implicitly the extraction of substrings of different length that satisfy the condition of lexical attraction between the component words.

Example 1: in Figure 1, from the word sequence "made in the national currency" can be extracted the subsequences: "national currency", "the national currency", "in the national currency", "made in the national currency". The irrelevant (incoherent) sequences and those susceptible of generating errors (like "the national", "in the", "made in", "made in the national") are ignored.

We call *superlink* an array $S = (pos_1, \ldots, pos_s)$ - where $pos_i, (i \in [1, s])$ represents the position of a word in a given sentence P - that satisfies the condition: there is an array of links of the form $[(pos_1, pos_2), (pos_1, pos_3), \ldots, (pos_1, pos_s)]$ or $[(pos_1, pos_s), (pos_2, pos_s), \ldots, (pos_{s-1}, pos_s)]$ that describes the sequence of words specified by the positions in S (the eventually intersecting links are filtered by LexPar).[3]

We call *chain* an array $L = (pos_1, \ldots, pos_l)$ - where pos_i, $(i \in [1, s])$ represents the position of a word in a given sentence P - that satisfies the condition: there is an array of links $[(pos_1, pos_2), (pos_2, pos_3), \ldots, (pos_{s-1}, pos_s)]$ that describes the sequence of words specified by the positions in L.

Observation: a pair of positions (pos_i, pos_{i+1}) is not necessarily composed by consecutive positions in the sentence P.

[3]The relation that brings together two positions in the same pair is different than the one we saw in the alignment arrays. In this situation, the positions are referring to words in the same sentence that are connected through lexical attraction.

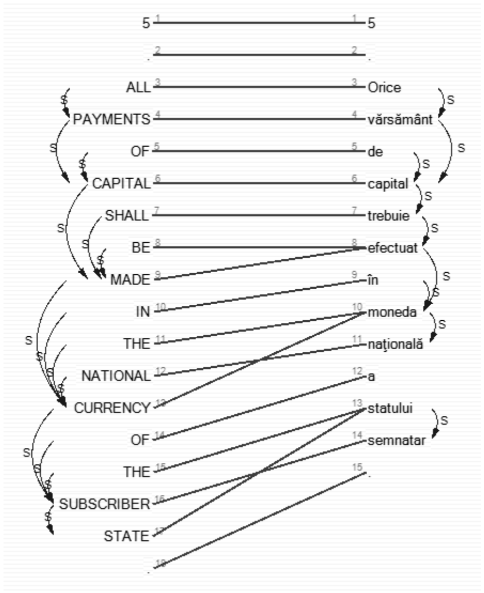

FIGURE 1: Visualisation for the alignments and links of one translation unit in the corpus. An arrow marks the existence of a syntactic dependency link between the words it unites, but the direction of the arrow is irrelevant (LexPar is not able to provide information about the head and the adjunct in the relation it identifies).

As input data, ExTract receives the processed corpus and the file containing the YAWA alignments. We will describe the extracting procedure for a single translation unit U in the corpus, containing P_{en} (a sentence in English) and its translation P_{ro} (a sentence in Romanian). ExTract works in two stages:

Stage 1. The superlinks and chains for P_{en} and P_{ro} are constructed. For simplification, we will describe the procedure for a generic sentence P. We collect the information concerning the lexical attraction from the "head" attributes of the words in P and put it in an array of pair positions : *Links.* The link array is constructed by traversing the sentence word by word and, for every word that has a "head" attribute, introducing in *Links* the pair *(position of the word, value of the attribute "head").*

Example 2: for the English sentence in Figure 1, Links contains the pairs: (3,4), (4,6), (5,6), (6,9), (7,9), (8,9), (9,13), (10,13), etc.

Based on the *Links* array, all the superlinks in the sentence are computed and inserted in an *Lfinal* array which will contain relevant link structures of variable size.

Example 3: at this step, Lfinal contains all the links in Links together with the following superlinks: (4, 5, 6), (7, 8, 9), (6, 7, 8, 9), (11, 12, 13), (10, 11, 12, 13), (9, 10, 11, 12, 13), (14, 15, 15), (13, 14, 15, 16).

The *Lfinal* array is consequently traversed and chains are computed from combination of adjacent simple links and superlinks but the number of chain loops is limited at 3. This decision is due to the necessity to avoid overloading the database with very long structures that can be easy constructed by the recombination mechanism. Empirical evidence suggested that the probability of two words separated by more than 2 chain loops to be lexically attracted (and thus morphologically dependent and conducting to boundary friction phenomena) is very small. The computed chains are added to the *Lfinal* array.

Stage 2. Assuming we computed (in the Stage 1) the two arrays $Lfinal_{en}$ and $Lfinal_{ro}$, we now have to set correspondences between the elements of those arrays, for constructing the translation examples. We remind that an array A containing word position pairs of the form (pos_i, pos_j) - with pos_i being the position of the word w_i in P_{en}, pos_j the position of the word w_j in P_{ro} and w_j the translation of the w_i - is available for helping set the correspondences.

The structure formalizing these correspondences is a set TE of triplets:

$$((p_1, p_2, \ldots, p_k), (p'_1, p'_2, \ldots, p'_s), trust_score),$$

so that for every p_i in $(p_1, p_2, \ldots, p_k)$ there is a p'_j in $(p'_1, p'_2, \ldots, p'_s)$ so that $(p_i, p'_j) \in A$ and one of the following conditions is satisfied:

1. $(p_1, p_2, \ldots, p_k) \in Lfinal_{en}$ and $(p'_1, p'_2, \ldots, p'_s) \in Lfinal_{ro}$;
2. $(p_1, p_2, \ldots, p_k) \in Lfinal_{en}$ and $(p'_1, p'_2, \ldots, p'_s) \notin Lfinal_{ro}$;
3. $(p_1, p_2, \ldots, p_k) \notin Lfinal_{en}$ and $(p'_1, p'_2, \ldots, p'_s) \in Lfinal_{ro}$.

In ideal work conditions, when both the linking and the lexical alignment are made without error, the lexical attraction properties in a fragment of the source sentence are supposed to be transfered in its tranlastion equivalent fragment of the target sentence. Thus, a pair of link structures which satisfies the condition 1 is more probably to represent a correct translation example than a pair satisfying condition 2 or 3. We decided to rank the examples according to that distinction, assigning a *trust score=2* to the examples falling under the condition 1 and a *trust score=1* to the other examples.

The algorithm for the construction of TE:

Step 1. The alignment 1:n and n:1 from A are grouped. Those multiple alignment, together with the simple ones from A, are introduced in TE. We execute this procedure because we want our database to contain lexical translation equivalences, which will compensate the absence of a dictionary.

Step 2. Correspondences between link structures (simple links, superlinks, chains) in the two sentences are set using the lexical alignment from A:

```
for all lists l in Lfinalen do
  list l'=null;
  for all positions p from l do
    extract from A a list C of pairs (x,x') for which $p = x$;
    for all (p,x') in C do
      introduce x' in l'
    end for
  end for
  if l' is in Lfinalro then
    trustscore=2;
  else
```

```
      trustscore=1;
    end if
    if (l,l',trustscore) is not in TE then
      introduce (l,l',trustscore) in TE;
    end if
  end for
  for all lists l' in Lfinalro do
    list l=null;
    for all positions p' from l' do
      extract from A a list C of pairs (x,x') for which p' = x';
      for all (x,p') in C do
        introduce x in l
      end for
    end for
    if l is in Lfinalen then
      trustscore=2;
    else
      trustscore=1;
    end if
    if (l,l',trustscore) is not in TE then
      introduce (l,l',trustscore) in TE;
    end if
  end for
```

Step 3. The final step is a simple recovery of the word sequences indexed by the lists of positions in TE and produces an output file whose entries have the form:

$enl_1(enf_1, enm_1)...enl_n(enf_n, enm_n)|rol_1(rof_1, rom_1)...rol_m(rof_n, rom_m)|trustscore$

where enl = English word lemma, enf= English word form, enm = English word MSD, 1..n = the positions of the words in the English sequence; similar, rol = Romanian word lemma, rof = Romanian word form, rom = Romanian word MSD, 1..m = the positions of the words in the Romanian sequence. The information about the lemma and the MSD tag of the word is extracted from the "lemma" and "ana" attributes in the corpus.

The corpus was divided in the part dedicated to the extraction of the translation examples (99% from the total amount of data) and the part for adjusting the recombining parameters and testing (1% from the total amount of data). After running ExTract on the extraction data, the results were counted and a file containing 2,650,000 different translation examples associated to their frequencies in the corpus and their trust scores was produced.

ExTract is not a part of the translation flow: the construction of the database is made previously and is also followed by an information reorganizing procedure that splits the data in 5 different files, connected by an efficient indexing system designed to improve the matching efficiency and speed. By dividing information in 4 different tables according to the language and to the level of morphological analyze (lemma or surface form), we can modularize the application so that the translation could be done in both directions (English-to-Romanian, Romanian-to-

English) and whether form-centered or lemma-centered.

Every entry in the output file of *ExTract* is associated with a translation example index (*tei*) and the information in the entry are distributed to the following files:

- en-forms: $tei \quad enf_1\ enf_2 \ldots enf_n \quad MD5(``enf_1\ enf_2 \ldots enf''_n)$
- en-lemmas: $tei \quad enl_1\ enl_2 \ldots enl_n \quad MD5(``enl_1\ enl_2 \ldots enl''_n)$
- ro-forms: $tei \quad rof_1\ rof_2 \ldots rof_n \quad MD5(``rof_1\ rof_2 \ldots rof''_n)$
- ro-lemmas: $tei \quad rol_1\ rol_2 \ldots rol_n \quad MD5(``rol_1\ rol_2 \ldots rol''_n)$
- infos: $tei \quad enm_1\ enm_2 \ldots enm_n \quad rom_1\ rom_2 \ldots rom_3 \quad frequency \quad trustscore$

where MD5 is a hash function often used in cryptography which associates to a string of characters a natural number on 16 bytes, usually represented as a sequence of 32 hexadecimal digits.

3.2 MatchRec: the matching and recombination stages

Because of the structure of the database, the matching process is reduced to a search in a list of natural numbers, so becoming very efficient. We will describe the matching and combination process for the direction of translation English-to-Romanian. The separation of the information about the lemma of a word and that about its word form becomes useful in the matching stage, when the application can follow one of the directions:

- word-form matching: **Step 1.** decomposes the English sentence to be translated (S) in smaller fragments, using the algorithms for identification of the superlinks and chains introduced in section 3.1 (in fact, the decomposing stage is similar to the Stage 1 from 3.1. applied to S and produces an array *LfinalS* similar to *Lfinalen* computed in that section); **Step 2.** for each element of the array link produced at Step 1:
  - recovers the form sequence indexed in the sentence - *form-string* - and computes *MD5(form-string)*;
  - search for *MD5(form-string)* in the MD5 list from the *en-forms* file and extracts the *tei* index for all the entries identified in this manner; a single MD5 identifier can have more associated *tei* because a form string in English can have more than one Romanian translation in the example database;
  - for each *tei*, recovers the information from *ro-forms* and *infos* files.
- lemma matching: the same steps are executed, but substituting en-forms with *en-lemmas*, ro-forms with *ro-lemmas* and form-string with *lemma-string.*

The MatchRec application organizes the information associated to a candidate translation example in an object from a class *trans-ex* that has the properties:

- *tei, en-form, en-lemma, ro-form, ro-lemma, en-msd, ro-msd, frequency, trust-score, md5, position:* these attributes receive values in the *matching* stage; the value for the *position* attribute is the list of positions from *LfinalS* corresponding to the word-form list en-forms (or en-lemmas for the lemma matching) and is necessary for distinguishing between identical words or word sequences found in different position in the sentence;

- *aligning-score, translation-score, overlapp-length-en, overlapp-length-ro, the-best-translation-score:* those attributes are initialised in the recombination stage.

The matching stage produces a list *Frg* of objects from trans-ex that represents the set of all the decomposing fragments of S found in the example database. The best possible translation for S must be constructed by combining the elements of *Frg*. For this stage, we choose the *Maximal Overlap Method* (Hutchinson *et al.*, 2003), that combines "overlapping fragments which translations are consistent". The authors exploit the intuition that when two translation example overlap at the source fragment level and at the target fragment level, the probability that a combination of these examples produces a correct translation is increased.

In (Hutchinson *et al.*, 2003), the overlap of two string is in fact left overlap: two strings $s = w_1, w_2, \ldots, w_n$ and $s' = w'_1, w'_2, \ldots, w'_m$ overlap if there is an integer $p < m, n$, so that $w'_1 = w_{n-p}, w'_2 = w_{n-p-1}, \ldots, w'_p = w_n$. The combination of the translation examples is guided by an *evaluation function s(E)* (see equations (1) and (2)) where E is a translation example represented as an object from the *trans-ex class. s(E)* is computed taking into account only one other translation example, considered to be the E's predecessor in the final solution and depends on the following parameters:

- *ovlengthen*: the length of the overlap between the English fragment of E and the English fragment of E's predecessor;
- *ovlengthro*: the length of the overlap between the Romanian fragment of E and the Romanian fragment of E's predecessor;
- *lengthen*: the length of the English fragment of E (a longer fragment is favored);
- *gap*: the distance from the first word in the English fragment in E and the last word of the English fragment in E' predecessor as measured in the context of the sentence S;
- *alignment*: the alignment score, computed from the combination of the frequency and trust score of E with an MSD score given by an MSD translation model (the score is computed for the pair (en-msd, ro-msd) associated to E and the model is extracted from the info file).

$$s(E) = g * gap + s'(E) \tag{1}$$

$$s'(E) = \frac{1}{a * alingment + o * (ovlengthro + ovlengthen) + l * lengthen + 1} \tag{2}$$

The g, a, o and l coefficients are optimized experimentally. The evaluation function for the entire sentence, *s(S)*, is additive on the set of the translation examples that decompose S. To minimize the number of computations (which tend to be very large) (Hutchinson *et al.*, 2003) designed a beam best-first technique, expanding the first best unfinished candidate at a certain moment and keeping in memory only the first best n unfinished candidates. The algorithm produces an array of translation examples (called *Solution*) which combination should produce the best possible translation for the sentence S in the conditions of a specific example database. The guidelines in (Hutchinson *et al.*, 2003) were implemented quite precisely, with few modification:

- taking into account the overlap length of the Romanian fragment;
- setting a coefficient $l = \sqrt{lengthen}$ which can better differentiate the evaluation function and favor a long example against the sum of the translation score of more shorter examples;
- setting a beam n=120, optimized experimentally.

A final stage of processing is necessary for transforming the information contained in the array *Solution* into a sentence in the target language. For both the lemma matching and form matching options, the adaptation will imply elimination of the sequences doubled by overlapping, concatenation of the sequences and some rules of reordering based on the MSD sequences associated to the word sequences. For the lemma matching direction, it was necessary to integrate a mechanism of word form generation before the adaptation stage. This procedure consist of two steps:

- the prediction of the most probable sequence of MSDs for the target lemma sequence of a specific fragment, having the translation model on MSDs we mentioned before and the MSD sequence associated in the corpus to that specific fragment;
- the generation of the form sequence starting from the lemma sequence and the MDS sequence (identified at the previous step) using a generation routine for the romanian word form generation and an word-form lexicon developed for the english word-form generation (both the application and the lexicon were already developped at ICIA).

4 Performance Evaluation and Conclusions

We mentioned we kept 1% of the working corpus for optimising the parameters and testing the performance. The measure used for evaluation is the BLEU score and we used only one reference text. As the purpose of this research was to compare the performance of the lemma-centered and the form-centered translation process, we considered interesting to see how the BLEU scores differ if the evaluation is made at the lemma level and what is lost in the form generation process. In Table 1, we can notice that the final BLEU score is better when the whole translation process in centered on the form for the English-Romanian direction (0,3088 vs. 0,2911) and when the translation process in centered on the lemma for the Romanian-English direction (0,3689 vs. 0,3575). We consider these results are promising and we believe that we can improve them by:

- augmenting the translation examples database with chunk-based examples (the TTL application has a chunking module with good performance);
- increasing the database dimension by almost 10 times (using resources developped lately at ICIA).

TABLE 1: Results of the BLEU score evaluation on a test set of 600 sentences using only one reference.

	lemma matching		form matching	
	BLEU score on lemmas	BLEU score on forms	BLEU score on lemmas	BLEU score on forms
English-Romanian	0,3493	**0,2911**	0,3484	**0,3088**
Romanian-English	0,3733	**0,3689**	0,3744	**0,3575**

References

Thorsten BRANTS (2000), TnT: A Statistical Part-of-Speech Tagger, in *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, pp. 224–231, Seattle, WA.

Lambros CRANIAS, Harris PAPAGEORGIOU, and Stelios PIPERIDIS (1994), A Matching Technique in Example-Based Machine Translation, in *Proceedings of Coling 1994, The 15th International Conference on Computational Linguistics*, pp. 100–104, Kyoto, Japan.

R. HUTCHINSON, P.N. BENNETT, J.G. CARBONELL, P. JANSEN, and R. BROWN (2003), Maximal Lattice Overlap in Example-Based Machine Translation, Technical Reports CMU-CS-03-138/CMU-LTI-03-174, School of Computer Science, Carnegie Mellon University.

Radu ION (2007), *Word Sense Disambiguation Methods Applied to English and Romanian*, Ph.D. thesis, Romanian Academy, Bucharest.

A. KUMANO and H. HIRAKAWA (1994), Building an Empty Dictionary from Parallel Texts Based on Linguistic and Statistical Information, in *COLING-94: Proceedings of the 15th International COnference on Computational Linguistics*, pp. 76–81, Kyoto, Japan.

Julian KUPIEC (1993), An Algorithm for Finding Noun Phrase Correspondences in Bilingual Corpora, in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 17–22, Columbus, Ohio.

F. SMADJA, K.R. MCKEOWN, and V.H. HATZIVASSILOGLOU (1996), Translationg Collocations for Billingual Lexicons: A Statistical Approach, *Computational Linguistics*, 22(1):1–38.

Ralf STEINBERGER, Bruno POULIQUEN, Anna WIDIGER, Camelia IGNAT, Tomaz ERJAVEC, Dan TUFIŞ, and Daniel VARGA (2006), The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages, in *Proceedings of the 5th International Conference on Language Resources and Evaluation*, Genoa, Italy.

Dan TUFIŞ, Radu ION, Alexandru CEAUŞU, and Dan ŞTEFĂNESCU (2006), Improved Lexical Alignment by Combining Multiple Reified Alignments, in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL2006)*, pp. 153–160, Trento, Italy.

K. YAMAMOTO and Y. MATSUMOTO (2003), Extracting Translation Knowledge from Parallel Corpora, in Michael CARL and Andy WAY, editors, *Recent Advances in Example-Based Machine Translation*, Kluver Academics, The Netherlands.

Deniz YURET (1998), *Discovery of linguistic relations using lexical atrraction*, Ph.D. thesis, Department of Computer Science and Electrical Engineering, MIT.