Simple linguistic methods for improving a word alignment algorithm

Ana-Maria Barbu

Research Institute for Artificial Intelligence (RACAI) – Calea 13 Septembrie 13, PC 050711,
Bucharest – Romania
abarbu@racai.ro

Abstract

This paper approaches word alignment problems and aims to show how some linguistic methods can combine with the statistical ones in order to get better results. In this sense, the system that participated in the shared task HLT/NAACL 2003 is presented in its actual version including the improvements the paper focuses on. This system, called TREQ-AL, works on parallel bilingual texts and needs some common pre-processing, such as tokenization, sentence-alignment, tagging, which are also shortly presented. In order to fulfill the word-alignment task, TREQ-AL relies on the output of a statistical translation-equivalence extractor along with which is described in a main section of the paper. The linguistic improvements taken into account are shown in another section and they refer to the cognate detection, the precedence constraint, pair assignments, collocations and language-specific rules. Instead of conclusions, graphical evaluation data are offered in the final section of the paper.

Keywords: word alignment, translation equivalence, linguistics, statistics.

1. Introduction

The work this paper relies on was roughly developed in the shared task organized as part of HLT/NAACL 2003 workshop on “Building and Using Parallel Texts: Data Driven Machine Translation and Beyond”. The task consisted of finding correspondences between words and phrases in the parallel texts of an aligned bilingual corpus. Assuming sentences in language L1 translated into language L2, the task supposed to build a word alignment system indicating which word token in the text of language L1 corresponded to which word token in the text of language L2. Corpora for two pairs of languages were provided: French-English (20 million words) and Romanian-English (1 million words) in order to train and test the competitive systems. All data were pre-tokenized. Final results should indicate the number of each parallel sentence and pairs of numbers corresponding to numbered tokens (both words and punctuation marks) associated in the two texts of that sentence. Tokens not translated were associated with nulls. More details about this evaluation exercise can be found in Mihalcea and Pedersen (2003).

Our team participated in Romanian-English subtask, beside other six teams from around the world, with the system called TREQ-AL (for TRanslation-EQuivalence ALigner). The complex work devoted to this task is partially described in Tufis *et al*. (2003). This system has been built in less than three weeks and in despite of this short time the results were quite comparable with those of the most powerful systems. Since that moment we have improved the word-alignment algorithm especially by developing its linguistic component. Our new results, better than those of the other participant systems in the workshop—either purely linguistics-based system like Zhao and Vogel (2003) or purely statistics-based ones like Dejean *et al.* (2003), have confirmed our intuition that linguistic methods can complete the statistical ones.

Figures can clearly express the improvement we talk about, if we take a look at the values of three main measures: Precision (P)(which indicates how many alignments are correct from the total alignments offered by the system), Recall (R) (which indicates how many correct alignments are found from the total alignments that should be found) and F-measure (F) (which is defined as 2*P*R/(P+R) and is a "correlation" of the first two). For the version of TREQ-AL at the moment of the shared task, we have got the following values: P=81.29%, R=60.26 and F=69.21, for non-null alignments. After developing the algorithm the measures amounted to significantly better values: P=84.95%, R=65.13 and F=73.73.

The paper aims at describing the ways the mentioned growth was possible and, in general, how linguistics can help statistics in natural language processing, in the particular application of word-alignment for bilingual texts. The content of the paper is structured on the following sections. The first one shortly presents the basic steps for processing parallel texts. A main section is devoted to the statistical part of the task, as a primary treatment; more precisely, it describes the central algorithm TREQ for translation equivalents extraction, which is completed, afterwards, with non-linguistic techniques of word to word alignments. The second main section emphasizes linguistic corrections made to improve the results. This kind of adjustments relies on the syntactic similarities displayed by Romanian and English. Some of these are very general, so that they can be inferred, at least, for some other Romance and Germanic languages. Paper ends with conclusions about gains of the approach.

2. Pre-processing steps of a parallel text

A parallel text is an association of two (or more) texts in different languages, which represent translations of each other. In order to get the most informative parallel text, appropriate for complex and efficient treatments, raw texts need to pass through more steps, described below.

2.1. *Segmentation*

A rough computational definition for 'word' is, as regarding written texts, any item separated by white spaces. However this concept is not operable in natural language, in that there are items, for instance in English, like "he's" (for *he is*) or "couldn't" (for *could not*) which do not represent only one word, but two. Without doubt, there are such elisions in many languages and they have to be splitted (by white space insertions) for a good machine understanding. Moreover, punctuation is conventionally written next to the previous item, without any space. What results does not form one word, as it can not be found in any lexicon. So punctuation marks have to be splitted, too.

On the other hand, there are a lot of multiword lexical tokens in any language. English verbs with particle (e.g. *cut off*, *set on*) or grammatical phrases (e.g. *in despite of*) are good examples in this sense. All these words with one lexical meaning have to be treated as a compound unit and marked as such. Usually, this is done by replacing spaces with underscores (e.g. *in_despite_of*). This task is called segmentation (or tokenization) and the program that performs it, segmenter (or tokenizer).

Note that a good delimitation of lexical tokens is important for translation purposes and, in our case, for word alignment, where 'word' has just the meaning of lexical token. For our work, we use Philippe di Cristo's multilingual segmenter Mtseg (http://www.lpl.univ-aix.fr/projects/multext/MtSeg) developed for MULTEXT project.

2.2. *Sentence alignment*

After tokenizing the two parts of a parallel text, each sentence of them gets a unique identifier. Then texts and sentence identifiers are given as input to the sentence aligner, whose task is to put in correspondence one or more sentences in a language with one or more sentences in the other language, through their identifiers. Each correspondence marks an alignment unit, also called from now on *translation unit*. Such a sentence aligner is CharAlign, fully described in Gale and Church (1993).

Obviously, the sentence alignment step is crucial for word-to-word alignment, due to the fact that the latter is only a refinement of the former. Errors in sentence alignment can trigger even more errors at word alignment level. That is why the results of this process should be checked out. A verification criterion could be the percentage of alignment units consisting of correspondence of one sentence to only another one, which should be more than 94%, as we have noticed in our experiments on a parallel corpus of six languages. So just those translation units that do not display one-to-one correspondence could hide alignment errors.

2.3. *Tagging and Lemmatization*

Another very useful step for natural language processing is that of indicating the proper part of speech for each lexical token. Sometimes, more parts of speech are appropriate for the same word, in the case of homographs. In English, there are plenty of homographs, so for example *well* appears in lexicon as an adverb, adjective, noun and verb. In text, *well* has to get only one part of speech, depending on its context. This is the task of a tagger. So tagging is the process of assigning a morpho-lexical label to each token and of solving ambiguity cases.

First of all, taggers need a training stage in which they learn, on the one hand, words and their appropriate morpho-syntactic tag(s) and, on the other hand, sequences of such tags as they are encountered into pre-tagged texts. With these data, taggers build a *language model* and afterwards they apply it to unknown texts.

Tagger performances depend on the size and the quality of the tagset, in that the morpho-syntactic labels have to be chosen so that they are not too many and offer relevant grammatical information. For our work, we use TnT (Brants, 2000), a trigram HMM tagger, whose accuracy amounts to more than 99%, for a Romanian tagset of 92 non-punctuation tags.

Lemmatization is a process of finding the normal form (lemma) for each word, especially for the inflected ones. That can be done either by program or, the simplest, by looking up into a monolingual lexicon of inflected forms. Working on lemmas is very important in natural language processing from statistical point of view: the number of wordforms submitted to statistical methods decreases and that of their occurrences grows.

In the example below, one can see the result of the pre-processing steps described until now for an one-sentence parallel text in *xml* format. Note that the element *tu* marks a translation unit, *seg* -the language, *s* -a sentence, *w*- a wordform, *c* -a punctuation mark, while the attribute *id* specifies the sentence/translation unit identifier, *lemma*—the normal form, and *ana*—the morpho-syntactic label.

Exp. 1

```
<tu id="Ozz.60">
<seg lang="ro"><s id="Oro.60"><w lemma="de_exemplu" ana="R">de_exemplu
</w> <c>,</c> <w lemma="ca" ana="C">ca</w> <w lemma="istoric" ana="N">
istoricul</w> <w lemma="Treptow" ana="N">Treptow</w> <w lemma="fi" ana="V">
este</w> <w lemma="american" ana="A">american</w> <c>.</c> </s></seg>
<seg lang="en"><s id="Oen.60"><w lemma="that" ana="Di">that</w> <w
lemma="historian" ana="N">historian</w> <w lemma="Treptow" ana="N">Treptow
```

```
</w> <w lemma="be" ana="V">is</w> <w lemma="an" ana="Di">an</w> <w
lemma="American" ana="A">American</w> <c>,</c> <w lemma="for_example"
ana="R">for_example</w> <c>.</c> </s></seg>
</tu>
```

3. From TREQ to TREQ-AL

3.1. *TREQ system*

The TREQ system requires sentence-aligned parallel text, tokenized, tagged and lemmatized, as we have already discussed. Its aim is to get translation equivalences from the text, in other words to create a bilingual dictionary based on that text.

The algorithm makes use of two underlying assumptions:

1. a lexical token is translated by only one token in the other language (Melamed, 2000);

2. the two members of a translation equivalence have the same part-of-speech.

These assumptions are restrictive, indeed, but they do not prevent additional processing units from recovering some of the missed or incomplete translations, as we shall see in TREQ-AL's case.

The baseline of TREQ is quite simple (for a complete presentation see (Tufis & Barbu, 2002)). For each translation unit, every word (actually, its lemma) is paired with every word of the same part-of-speech in the other language. Thus, lists of translation candidates result for each part-of-speech. For instance, the lists extracted from the example 1, in the previous section, are given below:

Exp. 2
```
 N (noun): (istoric historian), (Treptow historian),
             (istoric Treptow), (Treptow Treptow)
 V (verb): (fi be)
 A (adjective): (american American)
 R (adverb): (de_exemplu for_example)
```

Note that the words without correspondent in the other language are ignored (e.g. Di: *that* and *an* –in English, and C: *ca* –in Romanian).

Let a pair be noted p=(w_1 w$_2$). Then the algorithm computes, as if there are more parallel texts only with nouns, verbs etc., the score for each pair p with the following loglikelihood formula:

$$LL(w_1\, w_2) = 2 * \sum_{j=1}^{2} \sum_{i=1}^{2} n_{ij} * \log \frac{n_{ij} * n_{**}}{n_{i*} * n_{j*}}$$

where,

n_{11} = the number of p's occurrences in the whole text;
n_{12} = the number of pairs in which there is w_1 but not w_2, i.e. (w_1 ¬w_2);
n_{21} = the number of pairs in which there is w_2 but not w_1, i.e. (¬w_1 w_2);
n_{22} = the number of pairs in which there is neither w_1 nor w_2, i.e. (¬w_1 ¬w_2);
n_{1*} = the number of pairs in which there is w_1 (irrespective of its associations);
n_{*1} = the number of pairs in which there is w_2 (irrespective of its associations);
n_{2*} = the number of pairs in which there is w_1 does not appear;
n_{*2} = the number of pairs in which there is w_2 does not appear;
n_{**} = the total number of pairs.

After the score-calculating stage, the translation units are inspected again, one by one, and for each of them the candidates with the highest scores are chosen as translation equivalences if the scores are higher than a confidence threshold (empirically set to 9). Note that if the pair p

wins, other pairs containing w_1 or w_2 are ignored. Thus, the TREQ dictionary is made up of all the selected translation equivalences taken once.

3.2. *TREQ-AL system*

TREQ-AL has as input the TREQ lexicon and the parallel text to be aligned at the word level. The alignment is expressed, this time, in word-position terms, that is, the words are represented by their position in the translation unit, separately for each language. For the example 1, retaken here simplified and with words numbered, TREQ-AL should produce the indicated list of assignments (where '–1' means that the corresponding word is not translated):

Exp. 3 **RO:** 0>de_exemplu 1>, 2>ca 3>istoric 4>treptow 5>fi 6>american 7>.
EN: 0>that 1>historian 2>treptow 3>be 4>an 5>American 6>, 7>for_example 8>.
word alignment: (0 7), (1 6), (2 0), (3 1), (4 2), (5 3), (6 5), (7 8), (-1 4)

In order to get such results, TREQ-AL goes through the following processing steps reiterated with each translation unit.

3.2.1. *Dictionary looking-up*

First, for each word in the source language (here Romanian), TREQ-AL looks for the appropriate translation equivalent(s) into the TREQ lexicon. For those words that are not found in the lexicon, the system searches cognates (that is, similar words with the same meaning in the two languages) among not assigned target words. The looking-up is done irrespective of the part-of-speech, in order to avoid tagging errors. For instance, again in the example 1 (section 2.3), the English word *that* (position 0) is wrongly tagged as determiner (Di), and usually it could not be aligned with the Romanian conjunction *ca* (position 2). However, TREQ produces the conjunction pair (*ca that*) from many other occurrences in the parallel text and therefore this assignment can be recovered.

This step results in a list of (possibly non-consecutive) positions of those source words for which one or more translation equivalents were found, as the example 4 shows in the first column.

Exp. 4

<table>
<tr><td colspan="3"><b>RO:</b> 0>sa 1>sine 2>aplica 3>lege 4>si 5>in 6>caz 7>un 8>apropiat 9>al 10>domn 11>talpes 12>, 13>si 14>al 15>nu 16>mai 17>sti 18>cine 19>!</td></tr>
<tr><td colspan="3"><b>EN:</b> 0>that 1>the 2>law 3>be 4>also 5>enforce 6>in 7>the 8>case 9>of 10>a 11>person 12>close 13>to 14>mr 15>talpes 16>and 17>to 18>who 19>know 20>who 21>else 22>.</td></tr>
<tr><td>2- aplica /enforce-5*</td><td>2-5 aplica /enforce-5</td><td>2-5 aplica /enforce-5</td></tr>
<tr><td>3- lege /law-2/be-3/enforce-5</td><td>3-5 lege /law-2/be-3/enforce-5</td><td>3-3 lege /be-3</td></tr>
<tr><td>4- si /that-0/and-16</td><td>4-0 si /that-0/and-16</td><td>4- -1 si /*</td></tr>
<tr><td>5- in /in-6/of-9/to-13/to-17</td><td>5-6 in /in-6/of-9/to-13/to-17</td><td>5-6 in /in-6</td></tr>
<tr><td>6- caz /case-8</td><td>6-8 caz /case-8</td><td>6-8 caz /case-8</td></tr>
<tr><td>7- un /that-0/a-10</td><td>7-10 un /that-0/a-10</td><td>7-10 un /a-10</td></tr>
<tr><td>8- apropiat /close-12</td><td>8-12 apropiat /close-12</td><td>8-12 apropiat /close-12</td></tr>
<tr><td>11- talpes /talpes-15</td><td>11-15 talpes /talpes-15</td><td>11-15 talpes /talpes-15</td></tr>
<tr><td>13- si /that-0/and-16</td><td>13-16 si /that-0/and-16</td><td>13- -1 si /*</td></tr>
<tr><td>16- mai /also-4/else-21</td><td>16-21 mai /also-4/else-21</td><td>16-21 mai /else-21</td></tr>
<tr><td>17- sti /be-3/know-19</td><td>17-19 sti /be-3/know-19</td><td>17-19 sti /know-19</td></tr>
<tr><td>18- cine /that-0/who-18/who-20</td><td>18-20 cine /that-0/who-18/who-20</td><td>18-20 cine /who-20</td></tr>
<tr><td><b>Dictionary looking-up</b></td><td><b>Up-bottom alignment</b></td><td><b>Bottom-up alignment</b></td></tr>
</table>

*The alignment-line structure: ROposition – ENposition ROword /ENword1-ENposition1/…

3.2.2. *Up-bottom alignment*

The next step after dictionary looking-up is the up-bottom (or left-to-right) alignment, which processes the text in its normal reading sense. The target of this step is to do primary assignments and to coarsely solve the translation ambiguity.

The way of choosing a target word w_j from an ambiguity list depends on three factors: the cognate status -*cog*, the positional distance to the previous assignment -d_a and the relative distance to the source position -d_r. So, a target position j wins if it gets the best (in general, minimal) value for one of these dimensions:

$$j \Leftrightarrow \min \{cog, d_a, d_r\}$$

In the cases of non-ambiguity, the unique translation equivalent represents the proper link.

Note that at the end of this step, a target position can be assigned to more than one source position if it satisfies the selection criteria, as one can see in the example 4 (EN-position 5 assigned to RO-2 and 3).

Another important task fulfilled in this step is the detecting of a*lignment chains*, that is, sequences of at least four consecutive words in the source part associated with consecutive or close to each other words in the target part. For instance, in the example 4 (the second column), the links (5 6), (6 8), (7.10), (8 12) are memorized as an alignment chain, because Romanian positions are consecutive and no distance between two successive EN-positions is bigger than 2. The alignment chains are of great confidence in the ulterior word alignment process.

3.2.3. *Bottom-up alignment*

This step tries to refine and correct the primary assignation. It achieves a bottom-up (or right-to-left) alignment and takes into account more information than the previous step. The alignment criterion is a function depending on the following data:

- the distance to the lower assignment;
- the distances to the upper two assignments;
- the distance between source positions (especially relevant in the cases of gaps);
- the a*lignment chains*;
- the precedence constraint (presented later).

The result is a strict one-to-one word mapping, which can reflect modifications or even deletions (marked in the example 4 with '-1') of the links in the previous step, if no translation equivalent satisfies the alignment criterion. Note that this criterion affects both the ambiguous and non-ambiguous positions.

The next two steps use general linguistic knowledge for aligning the words that remain unaligned because there is no translation equivalent for them or the existing one(s) missed the alignment criterion.

3.2.4. *Alignment zones*

The system delimitates and count off, in each part of the translation unit, contiguous pieces of text that begin with a conjunction, a preposition or a punctuation mark and end with the token preceding the next conjunction, preposition, punctuation or end of the sentence. These are used as alignment zones in that they are mapped from one language to the other via the links assigned in the previous steps. That helps to filter out the links that exhibit aberrant zone mapping, for instance if the source words in zone 1 are aligned with target words in zones 2, 7 and again 2, then the link inducing the mapping with zone 7 is deleted. It should be said that it is possible to get some unmapped zones, namely those which contain no aligned words.

3.2.5. *The final word-alignment*

Now, the algorithm looks for aligning un-linked words inside the zones mapped at the previous step. First, the words of the same part-of-speech are aligned and then the system tries to do cross-part-of-speech or multiple alignments according to some language-specific rules.

For an unmapped zone, the search space for new alignments is that between the closest links on both sides of that zone.

Any word in a language or another that has not been aligned after these processing steps is automatically assigned a null link.

4. Linguistic improvements for TREQ-AL

In order to fill the gaps in the word alignment and to solve ambiguity classes, beside the procedure presented until now, we applied the following linguistic assumptions and methods.

4.1. *Cognate detection*

Even if TREQ has a special module for treating cognates, for statistical reasons not all the cognates in the parallel text are validated as translation equivalences. Such a reason could be a very low number of occurrences in the text, which triggers a low statistical score not passing the necessary threshold. For recuperating this loss, TREQ-AL recalculates the cognate score for every possible translation pair not found in the dictionary, during the looking-up step. However some parts-of-speech, namely the functional ones, such as prepositions, conjunctions, pronouns, are ignored. The cognate-detection demarche turns out to be very benefic for the result of the word alignment, because an important number of translation equivalences are recovered. The linguistic base in the cognate detection is both the fact that many languages have plenty of words with common etymons, and the fact that, nowadays, there is an important terminology migration between languages.

A problem we had to solve was to set up the minimal limit for declaring words to be cognates. That has to be done so that the alignments gaps left by TREQ be filled with as many as possible correct equivalences. A too high threshold leaves many gaps unsolved, while a too low one induces alignment errors. From our experiments, we got the optimal cognate limit of 0.65. Without doubt, this limit depends on the language pair and the text nature. For illustration, we give below some Romanian-English pairs of cognates and their corresponding scores.

Exp.5

RO-word	**EN-word**	**Cognate score**
statistica	statistic	0.95
rominia	romanian	0.80
tipar	type	0.75
noiembrie	november	0.68
coruptie	corruption	0.65

4.2. *Precedence constraint*

Choosing the appropriate translation of a word from a list of many possible translations is an important step, because an error at this level triggers errors in other points of the sentence. In order to help the disambiguation process we set on the precedence constraint, relying on the general linguistic fact that certain parts-of-speech always precede others. For instance, prepositions, articles, determiners, even conjunctions precede nouns, adjectives, adverbs and sometimes verbs (especially participles).

At the first reading of the bilingual text, sequences of positions in the sentence are memorized if they start with prepositions, articles, determiners or conjunctions, and contain nouns, adjectives, adverbs and verbs only. That simulates somehow a chunker task for prepositional and noun phrases and exploits the fact that conjunctions (either coordinating or subordinating ones) always precede conjuncts.

The precedence constraint applies at the level of bottom-up inspection. Its role is of preventing the choosing of a translation from an ambiguity list if the order indicated by the memorized sequence is not respected, or of giving priority to the positions indicated in the sequence. In the apparently trivial example below, given the (preposition-noun) sequences 3-4 in Romanian and 5-6 in English, and the assignation of the Romanian position 4 to the English position 6 (done by TREQ), there is no difficulty in choosing, with priority, the English position 5 from the ambiguity `/on-3/on-5/in-13` as assignment for the Romanian position 3.

Exp. 6 **RO:** 0>poate 1>ca 2>intimplare 3>de 4>vineri 5>seara 6>de_la 7>otv 8>el9>vrea 10>determina 11>pe 12>cel 13>mai 14>mult 15>sa 16>exclama 17>exaspera 18>: 19>ajunge 20>!

EN: 0>maybe 1>what 2>happen 3>on 4>otv 5>on 6>friday 7>night 8>will 9>make 10>most 11>people 12>exclaim 13>in 14>exasperation 15>: 16>that 17>be 18>enough 19>!

0-0	poate /maybe-0
1-16	ca /that-16
2-2	intimplare /happen-2
3-3	de /on-3/on-5/in-13
4-6	vineri /friday-6
5-7	seara /night-7
…	

Note that the up-bottom process, illustrated above, initially aligned the Romanian position 3 with the English position 3, because this one was consecutive with 2. The precedence constraint proved this alignment to be wrong and corrected it from 3 to 5.

4.3. *Pair alignments*

This linguistic assumption applies at the level of finding alignments which translation equivalence dictionary says nothing about. It consists in taking pairs of parts-of-speech depending, in some extent, on each other. For instance, if there are sequences adjective-noun (irrespective of their mutual order) both in Romanian and in English and nouns are paired together, then it is very likely to associate the adjectives too. So for example, in the parallel text below such an adjective-noun sequence is given by the Romanian positions 0 and 1 (i.e.

chinuit arestare) and by the English positions 2 and 3 (*lame arrest*). TREQ only found the equivalence *arestare-arrest* and aligned the Romanian position 1 with English position 3. The assumption discussed here allows to put in correspondence the adjectives *chinuit* (at Romanian position 0) and *lame* (at English position 2) and creates, therefore, the new correct alignment 0-2.

Exp. 7

```
RO:   0>chinuit 1>arestare 2>al 3>lui 4>treptow
EN:   0>treptow 1>'s 2>lame 3>arrest

1-3*   arestare    /arrest-3*
4-0*   treptow/treptow-0*
```

The same holds for other kinds of pairs, such as noun-noun, preposition-noun/adjective/numeral, conjunction-verb/adverb, article-noun/numeral etc.

4.4. *Dictionary collocations*

In some cases, TREQ associates a word in a language with two words in the other language that happen to form a collocation. For instance, one can find, as translation equivalences in the TREQ lexicon, both *liceu=school* and *liceu=high*, while *high school* is an English collocation, and the proper translation equivalence is *liceu=high school*. In order to use this fact in word alignment, we assume that if two words in a text are consecutive and both are translation equivalents of the same word in the other language, then both of them should be aligned with the one. Thus, it results a 2:1 alignment. Consider the following example.

Exp. 8

```
Dictionary entries:
post,N tv,N      146.58
post,N station,N    280.97
Bilingual text:
0>post 1>acesta 2>pentru 3>care 4>sorin 5>rosca  …
0>this 1>tv 2>station 3>for 4>which 5>"  …
Alignment map:
0-1     post     /tv-1/station-2
1-0     acesta  /this-0/a-20
2-3     pentru  /for-3
3-4     care     /which-4
4-11    sorin    /sorin-11
5-12    rosca    /rosca-12
```

One can see that the pairs *post-tv* and *post-station* have got good scores (the figures on the right) for counting as valid translation equivalences in the dictionary. On the other hand, in the bilingual text to be aligned, both English words are adjacent. This entitles the assumption that the two English words translate together the Romanian one and therefore the right alignment is Romanian:0- English: 1 2.

4.5. *Language-specific rules*

Besides these assumptions we have applied some language-specific rules concerning Romanian versus English syntax particularities or cross-linguistic differences in part-of-speech mapping. Without getting into details, we only mention some few examples. Usually, the English phrases of two nouns: noun1 noun2, e.g. *chocolate candy*, are translated into Romanian as noun2-the preposition ‘de’-noun1: *bomboană de ciocolată*. On the other hand, Romanian has a lot of articles and particles mapping into English determiners and prepositions, respectively. The language-specific module is a distinct unit in TREQ-AL, in order to keep the generality of the algorithm, and this module can be adapted for other pairs of languages than Romanian and English.

5. Conclusions

As we have already said, the applying of the linguistic methods discussed above managed to improve the results of the algorithm TREQ-AL from the values obtained at the moment of the shared task: P=81.29%, R=60.26 and F=69.21, to significantly better ones at the moment: P=84.95, R=65.13, F=73.73. In order to get a graphical illustration about the improvement contribution of each method, we have done the following evaluation experiment. We have disabled each method in turn and we have noticed the new results of the algorithm with respect to the considered measures. The data presented in the table below show how much a measure grows or decreases by disabling the respective method.

Linguistic method	**Precision [%]**	**Recall [%]**	**F-measure [%]**
Cognate detection	-0.42	-0.64	-0.57
Precedence constraint	-1.92	-0.43	-1.00
Pair alignments	+0.01	-0.30	-0.19
Collocations	+0.15	-0.22	-0.09
Language specific rules	+1.07	-3.14	-1.68

Table 1. Method percentage contribution

As one can see, the language specific rules bring the most important contribution to the f-measure value. By deactivating this module the precision grows indeed but the recall dramatically decreases and the f-measure, either. It turns out that this is a necessary module and, as a general conclusion, that a language-oriented algorithm could be better than a general one. The next important share (of 1%) is due to the precedence constraint. This method increases the precision with almost 2% and the recall with about 0.5%, while the cognate detection improves both the precision and the recall with about half a percent.

We hope these data offer enough evidence about how much some simple linguistic assumptions can contribute to the general results of a statistical analysis on parallel texts.

References

Brants T. (2000). TnT – A statistical Part-of-Speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference*, ANLP-2000, April 29-May 3, Seattle, WA.

Dejean H., Gaussier E., Goutte C. and Yamanda K. (2003). Reducing Parameter Space for Word Alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, May-June, Edmonton, Canada: 23-26.

Gale W.A. and Church K.W. (1993). A program for Aligning Sentences in Bilingual Corpus. *Computational Linguistics*, vol. (19/1): 75-102.

Melamed D. (2000). Models of translation equivalence among words. *Computational Linguistics*, vol. (26/2): 221-249.

Mihalcea R. and Pedersen T. (2003). An Evaluation Exercise for Word Alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, May-June, Edmonton, Canada: 1-10.

Tufis D. and Barbu A.-M. (2002). Revealing Translators' Knowledge: Statistical Methods in Constructing Practical Translation Lexicons for Language and Speech Processing. *International Journal of Speech Technology*, vol. (5): 199-209.

Tufis D., Barbu A.-M. and Ion R. (2003). TREQ-AL: A word alignement system with limited language resources. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, May-June, Edmonton, Canada: 36-39.

Zhao B. and Vogel S. (2003). Word Alignment Based on Bilingual Bracketing. In *Proceedings of the HLT-NAACL 2003* Workshop on *Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, May-June, Edmonton, Canada: 15-18.