Improved Alignment Based Algorithm for Multilingual Text Compression

Ehud S. Conley and Shmuel T. Klein

Abstract. Multilingual text compression exploits the existence of the same text in several languages to compress the second and subsequent copies by reference to the first. This is done based on bilingual text alignment, a mapping of words and phrases in one text to their semantic equivalents in the translation. A new *multilingual* text compression scheme is suggested, which improves over an immediate generalization of bilingual algorithms. The idea is to store the necessary markup data within the source language text; the incurred compression loss due to this overhead is smaller than the savings in the compressed target language texts, for a large enough number of the latter. Experimental results are presented for a parallel corpus in six languages extracted from the EUR-Lex website of the European Union. These results show the superiority of the new algorithm as a function of the number languages.

1. Introduction

In countries like Canada, Belgium and Switzerland, where speakers of two or more languages live side-by-side, all official texts have to be published in multilingual form. The current legislation of the ever expanding European Union requires the translation of all official texts into the languages of all member states. As a result, there is a growing corpus of important texts, large parts of which are highly redundant, since they do not have any information content of their own, and are just transformed copies of some other parts of the text collection.

We wish to exploit this redundancy to improve compression efficiency in such situations, and introduce the notion of *Multilingual Text Compression*: one is given two or more texts, which are supposed to be translations of each other and are referred to as *parallel* texts. One of the texts will be stored on its own (or compressed by means of pointers referencing only the text itself), the other texts can be compressed by referring to the translation, using appropriate dictionaries.

Data compression in general, and text compression in particular, have for long been prominent topics in the Information Retrieval literature, as full text IR systems are voracious consumers of storage space. This is true not only for

*This is an extended version of a paper that has appeared in the Proceedings of the LATA'11 conference. The work has been done while the first author was a PhD student at Bar Ilan University.

the underlying textual database itself, but also for the auxiliary overhead, such as indices, dictionaries, thesauri, etc., see, for example, [15, 18]. This work concentrates on multilingual information retrieval systems and how their data could be compressed.

The need for compression itself is well established, and technological advances over the past decades did not change this fact significantly. Some of the main reasons for this are (1) the fact that compression does not only save space, but also reduce the number of required IO accesses, thus saving time; (2) the amount of data we wish to store is ever increasing; (3) storage space for less powerful machines, such as hand held devices, is still very expensive, see [18].

In a certain sense, multilingual text compression is an extension of *delta-coding*, in which source and target files S and T are given, with the assumption that T is very similar to S, for example in the case of several versions of the same software package. Highly efficient compression schemes have been designed for that case [5, 3], and the compressibility is obviously a function of the similarity of the input files. Our problem extends the delta-coding paradigm to the case where similarity is not based on the appearance of identical strings, but allows the use of some transformation to pass from a given text fragment to its matching part.

The basis for enabling multilingual text compression is first the ability to match the corresponding parts of related texts by identifying semantic correspondences across the various sub-texts, a task generally referred to as *alignment*. As the methods for detailed alignment are quite sensitive to noise, they usually use a rough alignment of the text as an auxiliary input. They might also use an existing multilingual glossary, but they always generate their own probabilistic glossary, which corresponds to the processed text (see [4, 8, 11, 2]).

The compression of parallel texts was first treated almost two decades ago in [16], but without using text alignment tools. The idea of using alignment was raised in [6], and a detailed algorithm was presented in [7]. Different, though basically similar algorithms have recently been suggested in [13, 1]. However, all these algorithms relate only to bilingual parallel texts; therefore, should three or more parallel texts be compressed, the algorithms would be applied to each source-target pair of texts independently.

The current work proposes a new compression scheme for parallel texts, referred to as *multilingual*. Similar to our previous *bilingual* algorithm [7], the new method also exploits alignment to increase space efficiency. However, the current algorithm stores the information regarding the aligned source-text fragments within the source text rather than within each of the target texts. On the one hand, significant savings are made for each target text, but on the other hand, a large overhead is paid for the source text. Nevertheless, major parts of the additional information are shared by many of the bilingual alignments and the incurred overhead converges to a constant rate for a large-enough number of target texts, which makes it worthy paying in such cases, as our empirical results clearly confirm.

The next section presents the details of the algorithm, and experimental results are reported in Section 3.

2. Multilingual compression algorithm

2.1. Model concept

An alignment for a given source-target pair of texts can be represented by a directed bipartite graph of source and target nodes with an edge from each target node to exactly one source node. Each node holds an identifier of a text and the boundaries of a word sequence within that text. An edge indicates that the source and target sequences represented by the connected nodes are matching according to the alignment.

k alignments for k text pairs, where all k target texts are translations of the same source text, can be described by a similar graph. The source and target sets of the common graph would be unions, respectively, of the k source and k target sets taken from the k bilingual graphs that correspond to the k given alignments. As all nodes of the source sets refer to the same text, overlaps are certainly possible. Moreover, since alignments usually connect semantically-equivalent source and target units, it is highly reasonable to expect many source overlaps within a set of alignments for a common source text. This assumption has also been clearly confirmed by empirical results, as detailed in Section 3.

The top half of Figure 1 displays three parallel paragraphs, in French, English and German, along with corresponding French-English and French-German alignments. The details of the used notations will be given below in Section 2.2. For the current example it suffices to know that the quadruples in angled brackets refer to pairs of aligned text sequences. For example, the French expression `jour de son adoption`, starting at position 8 and of length 4, corresponds in English to `date of its adoption`, which starts at position 9 and has also length 4, so this correspondence is indicated by $\langle 8, 4, 9, 4\rangle$. Similarly, the German counterpart is `Tag seiner Annahme`, starting at position 6 and of length only 3, so the corresponding quadruple is $\langle 8, 4, 6, 3\rangle$. Note also in this example that while there is a correspondence between `prend effet` and `take effect`, which is indicated by $\langle 5, 2, 5, 2\rangle$, the parallel German expression is `tritt...in Kraft`, which is not a sequence of consecutive words, so the alignment algorithm just ignored it.

The bottom half of Figure 1 depicts the respective alignment graph, built according to the above definition. The two numbers in parentheses within each node are the beginning position and the length of the implied sequence, respectively. Note that the two original source sets included a total of five nodes, whereas the union set contains only three. That is because two of the three French source sequences are shared by both English and German target counterparts. This is a representing example for the source node overlaps mentioned above.

The high rate of overlapping source nodes yields a union source set of size much smaller than the sum of the sizes of the k original sets, as demonstrated by Figure 1. This result hints that it might be worthy to encode that union set within the source text rather than storing each of the k original alignments in the corresponding target text, as does the bilingual algorithm. Obviously, the locations

n	S (Fr)	T^1 (En)	A_{S,T^1}	T^2 (Ge)	A_{S,T^2}
1	La	This		Dieser	
2	présente	Common	⟨3,2,2,2⟩	gemeinsame	⟨3,2,2,2⟩
3	position	Position		Standpunkt	
4	commune	shall		tritt	
5	prend	take	⟨5,2,5,2⟩	am	
6	effet	effect		Tag	⟨8,4,6,3⟩
7	le	on		seiner	
8	jour	the		Annahme	
9	de	date	⟨8,4,9,4⟩	in	
10	son	of		Kraft	
11	adoption	its		.	
12	.	adoption			
13		.			

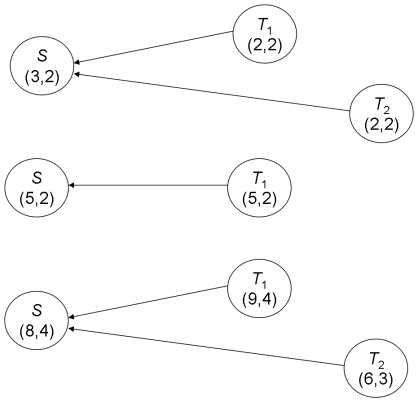

FIGURE 1. Example of bipartite graph representation of French-English and French-German alignments for the same sentence

of the target sequences as well as some data needed for the restoration of those sequences should still be encoded in each target text.

Indicating aligned source sequences within the source text is not just a technical modification, but rather a conceptual change in the perception of text-alignment relations. This change also leads to further storage savings, as explained below.

The bilingual model relates to the alignment data as local to the target text. Therefore, the location of each aligned source fragment is indicated at the respective target position by the signed offset from the rough alignment of the two parallel sections (sentences, paragraphs or the like). Yet, in order to compute

the rough alignment for each target position, the decoder must know in advance the lengths of the two sections. The length of the source section can be found by decoding it on its own. The target's length, however, is computable only after the text is decompressed using the alignment data. Consequently, it is obligatory to store the length of each target section.

Given a source text $S = s_1 s_2 \ldots$ and a target text $T = t_1 t_2 \ldots$, the rough alignment is calculated as follows: define $al(j) = \left\lfloor \frac{|S|}{|T|} j + \frac{1}{2} \right\rfloor$ as the expected position within S of the term corresponding to t_j in T. In other words, $s_{al(j)}$ is the source word parallel to t_j if taking into account only the proportion between the lengths of S and T. The accurate alignment may then be expressed by the signed offset from $s_{al(j)}$.

Unlike the bilingual case, the multilingual model regards the alignment as two independent sets of word sequences, one belonging to the source text and the other to the target text, with certain semantic equivalences between elements of these two sets. The target text is compressed by replacing the elements of the target set with references, which include pointers to their equivalents in the source set. The source set itself is encoded within the source text, independently of the target text.

Now that all source sequences can be listed before the target's deciphering begins, the section length data becomes redundant. The source sequence equivalent to each aligned target sequence can be implied by its offset from an item in the source list that is expected to be its counterpart. This offset is measurable in units of aligned sequences rather than single words, which include the unaligned words as well. Naturally, this indication method yields much smaller numbers. As a result, the value range narrows and thus the compression improves.

Figure 2(a) shows a pair of parallel pieces in French and English, accompanied by a respective alignment. Figure 2(b) presents the outputs of the bilingual and multilingual algorithms for the English text, playing the role of the target, based on the given inputs. The bold numbers immediately following the `#REF#` escape sequences are the offset values, interpreted differently for each of the algorithms, as detailed below.

To understand the values in the figure, we need the following information: the lengths of S and T were respectively 39 and 25, of which only the beginning is displayed. The values -3, used by the bilingual algorithm, indicate the fact that each of the two implied source sequences begins three *words* (tokens) before the position where it is expected to reside according to the initial rough alignment. More specifically, the target sequence `European Council` starts at position 5 in T and will be encoded by reference to the source sequence `Conseil européen` in S. Using the rough alignment, this reference is expected to start at source position number $\lfloor \frac{5 \times 25}{39} + 0.5 \rfloor = 8$, but in fact it is found 3 tokens before, at position 5. The word `juin` (= `June`) is also found 3 tokens before the location where it is supposed to be. With a difference, the 0 values output by the multilingual algorithm instead of the -3's tell the decoder that the translations of the first and second aligned

target sequences are the first and second *aligned source sequences*, just as expected according to their ordering. Relating to aligned sequences rather than all tokens has a similar effect of decreasing the absolute values of most offset fields, thereby yielding a significant improvement in compressibility.

Nonetheless, annotating the source text with aligned sequence data while still indicating the locations of aligned target sequences within the target text introduces a significant space overhead. Experimental results show that for a bilingual text, this overhead is even greater than the savings detailed above. On the other hand, the union source set for k targets has less elements than the sum of elements of the k original sets. Hence, as the overhead is linear in the size of the set, it may be worthy to pay it once in order to gain the profits suggested above for each of the k targets. This observation is also supported by our empirical results.

Note that the union source set might include several sequences which do not have correspondents in all k target texts. These cases are treated by the algorithm with a small additional overhead, but this loss is still far from canceling the profits of the model.

2.2. Algorithm details

In the sequel, we shall refer to the *lemma* of a given word (in plural: lemmata), which is the dictionary form in which this word usually appears. Reminiscent of the bilingual algorithm, the multilingual algorithm assumes the following resources:

1. S, T^1, T^2, …, T^k: The single source and k target texts, respectively, where $\forall h \in [1,k]$, T^h is a translation of S.
2. A_{S,T^1}, A_{S,T^2}, …, A_{S,T^k}: Word- and phrase-level alignments of the text pairs (S,T^1), (S,T^2), …, (S,T^k), correspondingly.

   Let $s_{i,l}$ denote the word sequence of length l within S beginning at the ith word. Similarly, let $t^h_{j,m}$ denote the word sequence of length m within T^h beginning at the jth word. A_{S,T^h} consists of a set of connections of the form $\langle i,l,j,m\rangle$, each of which indicating the fact that $s_{i,l}$ and $t^h_{j,m}$ have been determined as matching phrases. We assume that for any pair (j,m) there is at most one connection of the form $\langle i,l,j,m\rangle$ within A_{S,T^h}. From here and below, s_i and t^h_j stand for $s_{i,1}$ (the ith word of S) and $t^h_{j,1}$ (the jth word of T^h), correspondingly.
3. S^{lem}, $(T^1)^{lem}$, $(T^2)^{lem}$, …, $(T^k)^{lem}$: Lemmatized forms of S, T^1, T^2, …, T^k, respectively.

   Let $(s_{i,l})^{lem}$ and $(t^h_{j,m})^{lem}$ denote the lemma sequences corresponding to $s_{i,l}$ and $t^h_{j,m}$, respectively. That is, the concatenations of the lemmata of s_i, s_{i+1}, …, s_{i+l-1} and t^h_j, t^h_{j+1}, …, t^h_{j+m-1}, correspondingly.
4. L_S: A lemmata dictionary corresponding to S. The entries of this dictionary are the words appearing in S. Each entry stores a list of all possible lemmata of the keyword, sorted in descending order of frequency.

   Let $L_S(s)$ denote the lemma list for the word s. For instance, if S is an English text, then $L_S(\texttt{working}) = (\texttt{work}, \texttt{working})$.

n	S (French)	T (English)	$A_{S,T}$
1	(	(	
2	4	4	
3	)	)	
4	Le	The	
5	Conseil	European	$\langle 5, 2, 5, 2 \rangle$
6	européen	Council	
7	a	on	
8	approuvé	17	
9	,	June	$\langle 12, 1, 9, 1 \rangle$
10	le	2004	
11	17	endorsed	
12	juin	⋮	
13	2004		
14	,		
	⋮		

(a)

n	S (French)	Bilingual	Multilingual
1	(	(	(
2	4	4	4
3	)	)	)
4	Le	The	The
5	Conseil	#REF#, **-3**, 1, 0, 0	#REF#, **0**, 0, 0
6	européen		
7	a	on	on
8	approuvé	17	17
9	,	#REF#, **-3**, 0, 0	#REF#, **0**, 0
10	le	2004	2004
11	17	endorsed	endorsed
12	juin	⋮	⋮
13	2004		
14	,		
	⋮		

(b)

FIGURE 2. Sample outputs of the bilingual and multilingual algorithms for the same English target piece compressed using a French source

5. V_{T^1}, V_{T^2}, ..., V_{T^k}: Variant dictionaries corresponding to the target texts. For each $h \in [1, k]$, the entries of V_{T^h} are the lemmata of all words appearing in T^h. Each entry stores a list of all possible morphological variants of the key lemma, sorted in descending order of frequency.

   Let $V_{T^h}(t)$ denote the variant list for lemma t. For example, if T^3 is a French text, then $V_{T^3}(\texttt{normal}) = (\texttt{normal}, \texttt{normale}, \texttt{normaux}, \texttt{normales})$.
6. G_{S,T^1}, G_{S,T^2}, ..., G_{S,T^k}: Bilingual glossaries corresponding to the text pairs (S, T^1), (S, T^2), ..., (S, T^k), respectively. The entries of these glossaries are source-language lemma sequences. Each entry includes a list of possible translations of the key sequence into target-language sequences, sorted in descending order of frequency. The translations also appear in lemmatized form.

   Let $G_{S,T^h}(s)$ denote the translation list of the source-language sequence s into the language of T^h. For instance, if S and T^3 are English and French texts, correspondingly, then $G_{S,T^3}(\texttt{mineral water}) = (\texttt{eau mineral})$. Note that the word `eau` (water) in French is feminine, which requires a feminine-form adjective, namely `minerale`, whereas the adjective `mineral`—the corresponding lemma—is the masculine singular form.

2.2.1. Source annotation. The new algorithm inserts markup symbols into the source text for all aligned source sequences appearing in any of the k alignments. Each of these symbols indicates a different sequence length and is inserted just before the first word of the aimed sequence within the text. When more than one sequence begins at the same word, the respective annotations are inserted one after another preceding the start word. The order by which such successive marks are introduced can be determined arbitrarily. However, the compression procedure for the target texts relates to the order of these annotations when computing offset values for the aligned target sequences. Hence, this order must be decided before any target text is compressed.

The fact that the entries of the bilingual glossaries are lemmata sequences means that the aligned source fragments must be stored in a way that enables retrieving the lemma of each aligned word. Keeping only the lemmatized version of the source text (S^{lem}) is unacceptable because the original source text, being an integral part of the multilingual corpus, must be restorable like the k target texts. Due to space-efficiency consideration, we have chosen to store the inflected form of each source word along with its lemma index, if more than one lemma exists.

The above two requirements leave the following options:

1. Storing the original source words with additional lemmatization information: for each aligned word s, $s = s_i$, with more than one possible lemma (i.e. $|L_S(s)| > 1$), add the index, starting the count with 0, of $(s_i)^{lem}$ within $L_S(s)$ following s_i. For example, if $s_i = \texttt{working}$, $(s_i)^{lem} = \texttt{work}$ and $L_S(\texttt{working}) = (\texttt{work}, \texttt{working})$, then output 0 following `working`.
2. Storing the source lemmata with additional inflection information: for each lemma s^{lem}, $s^{lem} = (s_i)^{lem}$, with more than one possible variant (i.e. $|V_S(s^{lem})| > 1$), output the index of s_i within $V_S(s^{lem})$ (note that for this option, L_S is

no longer needed; instead, we need V_S, the variant dictionary for S). For instance, if $(s_i)^{lem} = \texttt{book}$, $s_i = \texttt{books}$ and $V_S(\texttt{book}) = (\texttt{book}, \texttt{books}, \texttt{booking})$, then output 1 following the lemma `book`. Of course, only aligned words must be stored that way. The unaligned words can be given in their inflected forms.

At first sight, these two options may look equivalent. However, it turns out that the former option is more space-efficient. That is because the average number of lemmata per variant is much lower than the average number of variants per lemma. Moreover, it is much more likely for a variant to have a single lemma than for a lemma to have a single variant, which results in more index omissions for the former option. The latter option, however, enables accessing the bilingual glossaries using the given lemmata without having to consult the lemmata dictionary first, which means faster decoding of the target texts. The additional space overhead, paid only once, may become negligible when many target texts are compressed using the same source.

Figure 3 presents the formal pseudo-code of the algorithm for annotating the source text with alignment and morphological information. In the given procedure, the first option is applied. That is, the original words of the text are output and a lemma index is attached to each aligned word. $I(x, y)$ denotes the index of the lemma x within the lemma list y. If $|y| = 1$, then $I(x, y) = \epsilon$ (the empty string); align_mark (l) stands for the special codeword indicating the beginning of an aligned sequence of length l; and the variable *aligned* is used to count the number of next aligned tokens, for which lemma indices are needed.

The output of the annotation procedure can be compressed using any encoding, for instance, Huffman coding with two Huffman trees: H_1^S will store the words and aligned-sequence marks, whereas H_2^S will hold the lemma indices. As alternative option, three trees might be used—one for the variant indices and two separate trees for unaligned words and aligned lemmata.

2.2.2. Target compression. Following the annotation of the source text, each aligned source sequence can be referred to using its ordinal number. At that point, each target text T^h may be compressed independently using the algorithm given below.

Beginning at the first position $j = 1$ within T^h, use A_{S,T^h} to find the longest sequence $t_{j,m}^h$ having a corresponding sequence $s_{i,l}$ in S. The redundant connections can be removed in advance, thereby avoiding redundant source sequences and their direct and indirect overheads. If a corresponding sequence is found, replace $t_{j,m}^h$ with the concatenation of a pointer to $s_{i,l}$ with some indices. These indices are necessary for the restoration of the missing target words. The substituting reference consists of the following details:

1. $ord\,(i, l) - expected$: Offset of $s_{i,l}$ from the expected sequence. This value is actually a pointer to $s_{i,l}$. ord (i, l) denotes the ordinal number of $s_{i,l}$ among the aligned sequences annotated within S. Initially, $expected = 0$; after using a connection $\langle i, l, j, m\rangle$, *expected* is assigned the ordinal number of the next sequence still not used which follows $s_{i,l}$. As an example, suppose that a certain target sequence, $t_{j,m}^h$, has just been replaced with a pointer to the 4^{th}

ANNOTATE_SOURCE

$U = \{(i,l) \mid \exists h \in [1,k], j, m : \langle i,l,j,m\rangle \in A_{S,T^h}\}$
$i \longleftarrow 1$
aligned $\longleftarrow 0$
while $i \leq |S|$ **do**
    for each l **such that** $(i,l) \in U$ **do**
        output align_mark (l)
        if $l >$ *aligned* **then** *aligned* $\longleftarrow l$
    end for each
    output s_i
    if *aligned* > 0 **then**
        lemma $\longleftarrow I((s_i)^{lem}, L_S(s_i))$
        output *lemma*
        aligned $\longleftarrow$ *aligned* $- 1$
    end if
end while

FIGURE 3. Annotation of source text with alignment and lemmatization information

aligned source sequence, $s_{i,l}$; in this case, $\mathsf{ord}\,(i,l) = 4$. In addition, suppose that source sequence #5 had already been used to compress another target sequence. Now, by default, *expected* should have been set to $\mathsf{ord}\,(i,l) + 1 = 5$. However, as source sequence #5 is currently marked "used", the new value of *expected* will be 6, unless the corresponding sequence is marked "used" as well. Hence, for the next aligned target sequence, the offset 0 would imply source sequence #6, 1 would mean #7, -1 would indicate #5, and so on. Obviously, source sequence #4 should now join the set of used sequences.

2. Index of $(t^h_{j,m})^{lem}$ within $G_{S,T^h}(s^{lem}_{i,l})$. In the case of a single translation, this index is omitted (same as emitting ϵ).
3. Indices of $t^h_j \ldots t^h_{j+m-1}$ within $V_{T^h}((t^h_j)^{lem}) \ldots V_{T^h}((t^h_{j+m-1})^{lem})$, respectively. Again, ϵ is used in the case of singletons.

The above reference is output preceded by a special codeword meaning "reference". The next iteration will work for $j = j + m$.

If no m is found such that $\langle i,l,j,m\rangle \in A_{S,T^h}$, then t^h_j is written to the output stream and j is incremented by 1. The process continues while $j \leq |T^h|$.

The way the expected source sequence (*expected*) is determined is derived from the nature of the alignment. Given a pair of aligned source and target sequences, it is probable that the source counterpart of the next target sequence be the next source sequence. Of course, differences in word and phrase ordering as well as the existence of additional source sequences, aligned with sequences in other target texts, often yield some small deviations, which are overcome using the offset values. Another quite realistic assumption is that a source sequence already referred to once during the process will be rarely used again. The search for the next unreferenced sequence is always very local, meaning that it is performed in time $O(1)$.

For a single alignment, the expected source sequence for target number n could be simply n itself. The unification of source sequences from k alignments into one common list does not permit such an assumption. In this case, relating to the last known connection as an anchor point is a more conceivable heuristics.

Figure 4 displays the formal pseudo-code of the algorithm for compressing target text number h, i.e. T^h. $I(x, y)$ has the same meaning as in Figure 3. The set *used* holds the ordinal numbers of already-used source sequences.

The output for each target text T^h can be encoded using three Huffman codes: $H_1^{T^h}$ will contain the unaligned words as well as the special `#REF#` escape sequence, $H_2^{T^h}$ will hold the offset values, and $H_3^{T^h}$ shall store the translation and variant indices. Indeed, if the distributions of the offsets and indices for all k texts are similar, then it might be worthy using two common codes, H_2^T and H_3^T, for all offsets and indices, respectively. As opposed to the indices, the offset values also include negative integers. Therefore, their distributions are significantly different, which makes it preferable to use two distinct codes.

One of the important properties of both our bilingual and multilingual schemes is that sections are compressed independently of other sections. The use of Huffman coding rather than BZIP, LZ or any other encoding with inter-section dependencies for representing both source and target texts enables storing them in blocks of any size without any decrease in space efficiency. The Huffman codes we used are not those obtained by applying the algorithm to the individual characters; such a 0-order entropy coder yields rather poor compression. We preferred the so-called *HuffWord* variant [14], in which the elements to be encoded are the different *words* of the text. The additional overhead of storing the words instead of the characters is needed anyway in our application, as these words form the dictionary of the text. Since entire words are encoded as single units, HuffWord incorporates higher order dependencies, and its compression is comparable with that of GZIP and BZIP.

Nevertheless, adaptive encodings are also applicable in our case, but the compression savings might be hurt in case of partition into small blocks. Indeed, methods based on the Burrows-Wheeler transform and adaptive methods like those based on LZ need large enough blocks to "learn" the statistics and adapt themselves to yield significant savings. When these blocks are too small, and this might

COMPRESS_TARGET (h)

```
j ⟵ 1
used ⟵ ∅
expected ⟵ 0
while j ≤ |T^h| do
   found ⟵ false
   for m ⟵ m_max downto 1 do
      if ∃i,l such that ⟨i,l,j,m⟩ ∈ A_{S,T^h}   // ⟨i,l,j,m⟩ is unique
         src_ord ⟵ ord(i,l)
         offset ⟵ src_ord − expected
         used ⟵ used ∪ {src_ord}
         expected ⟵ src_ord + 1
         while expected ∈ used do expected ⟵ expected + 1
         trans ⟵ I((t^h_{j,m})^lem, G_{S,T^h}((s_{i,l})^lem))
         for n ⟵ 0 to m − 1 do
            variant_n ⟵ I(t^h_{j+n}, V_{T^h}((t^h_{j+n})^lem))
         reference ⟵ concatenation(#REF#, offset, trans,
                        variant_0, ..., variant_{m−1})
         output reference
         j ⟵ j + m
         found ⟵ true
         break
      end if
   end for
   if not found then
      output t^h_j
      j ⟵ j + 1
   end if
end while
```

FIGURE 4. Compression of target text T^h using the source text

be the case in our intended application in which we require the possibility of accessing individual paragraphs or even sentences, these methods might yield poor compression. A Huffman code, on the other hand, could use the same tree for the entire file and still permit access to each of the small parts independently of the others, see, e.g., Figure 4 in [7].

The decompression algorithm is straightforward. Note that it needs only the dictionary files, as all relevant information included in the other files is encoded within the compressed text itself.

Figures 5 and 6 give an example of the algorithm's input and output, correspondingly. The index n in the first columns denotes the ordinal token number. The second column of Figure 5 lists the tokens of an English paragraph, taken from the experimental multilingual corpus (see Section 3). The third and fifth columns are the French and German parallels of that paragraph, respectively. The fourth and sixth columns show the connections suggested by the English-French and English-German alignments. As an example, the connection $\langle 6, 2, 6, 2\rangle$ in A_{S,T^1} indicates the fact that the English sequence `common position`, beginning at token number 6 and consisting of 2 tokens, is aligned with the French sequence `position commune`, also beginning at token number 6 (but of the French paragraph) and consisting of 2 tokens. Note that the English-German alignment (A_{S,T^2}) also relates to the same two English tokens; nevertheless, it aligns each of them separately with its German counterpart. This difference merely originates from the way in which the alignment algorithm computes the probabilities of candidate connections, which also takes into account some offset and length probabilities.

The second column of Figure 6 displays the English text, playing the role of the source text, with annotation of the aligned sequences as well as their lemmatization indices. In this specific case, ϵ's are output as lemmatization indices, because all 3 aligned tokens have only one possible lemma. The codeword `#AL1#` indicates that the next token is the beginning of an aligned sequence of length 1. Likewise, `#AL2#` marks the beginning of an aligned sequence of length 2. Notice that token number 6 is the beginning of two distinct sequences. The former, of length 1, comes from A_{S,T^2}, whereas the latter, of length 2, originates in A_{S,T^1}. At the same time, token No. 7 is also a 1-token sequence, also coming from A_{S,T^2}.

Finally, the third and fourth columns present the compressed forms of the French and German paragraphs, respectively. The two aligned sequences in each target text have been replaced with suitable references. For instance, the French sequence `position commune` has been substituted with the reference $1, \epsilon, 0, 0$. The offset 1 results from the fact that the previous replacement relates to source sequence number 0, namely, `purpose`. Thus, *expected* for the current connection is 1, which refers to the sequence `common`, rather than `common position`, enumerated as number 2. The offset, therefore, is $\mathsf{ord}\,(6, 2) - \mathit{expected} = 2 - 1 = 1$.

The sequence `common position` has a single translation in G_{S,T^1}, that is, `position commune`.[1] As a result, no translation index is written to the output stream (expressed in the figure by ϵ). When the decoder reaches the discussed reference, it will first identify the implied English sequence using the offset value. Then, after lemmatizing its words using L_S, it will look up the sequence in G_{S,T^1}.

[1] Note that the French lemmatizer has given the feminine form `commune` as the lemma of `commune`. That is because it referred to the feminine noun `commune` (community) rather than to the feminine form of the adjective `commun` (common). However, this has no significance for our algorithm.

At that point, it will find out that the proposed translation is unique, and therefore will not read an index at that stage. The next step will be seeking the word `position` in V_{T^1}, finding there several options such as `position`, `positions`, `POSITION` etc.. Now the decoder must read the next index, i.e. 0, in order to choose the right variant. The same process will be performed for the lemma `commune`, which also has a few possible variants.

The next section exhibits our empirical results on a real-life corpus and discusses their consequences.

n	S (English)	T^1 (French)	A_{S,T^1}	T^2 (German)	A_{S,T^2}
1	For	Aux		Im	
2	the	fins	$\langle 3,1,2,1 \rangle$	Sinne	
3	purpose	de		dieses	
4	of	la		gemeinsamen	$\langle 6,1,4,1 \rangle$
5	this	présente		Standpunkts	$\langle 7,1,5,1 \rangle$
6	common	position	$\langle 6,2,6,2 \rangle$	bedeutet	
7	position	commune		der	
8	,	,		Ausdruck	
9	the	on		:	
10	following	entend			
11	definitions	par			
12	shall	:			
13	apply				
14	:				

FIGURE 5. Example of paragraphs with corresponding alignments

3. Results and analysis

In order to assess our algorithms, we extracted a 6-language corpus from the *EUR-Lex* website [9], which holds the European Union's legislative publications of the last few years in all EU members' languages (currently 23). Our subset was formed of all texts published between January 1st and May 31st 2005 in the following languages: German (de = Deutsch), English (en), Spanish (es = Español), French (fr), Italian (it) and Portuguese (pt).

Table 1 displays the size of each part of the corpus in MBs and in million words. Notice that the Portuguese text is significantly smaller than the others. That is because a very large document, constituting around 11.5% of each of the other texts, had no Portuguese version. It should also be noted that a few other small pieces are missing in most of the texts since they have not been translated or due to technical problems. This situation simulates a real-life corpus, where not all contents exist in all languages. This fact makes the current results even more relevant.

n	Annotated S (English)	Compressed T^1 (French)	Compressed T^2 (German)
1	For	Aux	Im
2	the #AL1#	#REF#, 0, ϵ, 0	Sinne
3	purpose,ϵ	de	dieses
4	of	la	#REF#, 1, 0, 0
5	this #AL1# #AL2#	présente	#REF#, 1, ϵ, 1
6	common,ϵ #AL1#	#REF#, 1, ϵ, 0, 0	bedeutet
7	position,ϵ		der
8	,	,	Ausdruck
9	the	on	:
10	following	entend	
11	definitions	par	
12	shall	:	
13	apply		
14	:		

FIGURE 6. Compression of French and German texts using their English parallel

Unit	de	en	es	fr	it	pt
MB	27.37	25.98	28.03	28.56	27.55	24.28
MW	4.46	4.79	5.10	5.20	4.93	4.41

TABLE 1. Full sizes of the parallel texts

The lemmatization of the texts was done using the *Tree Tagger* [17], a language-independent part-of-speech tagger and lemmatizer, currently adapted to several European languages. As the paragraphs of each document in the various languages were not precisely aligned with each other, we applied a simple adjustment of the *DKvec* algorithm [10] to all 30 possible pairs of texts in order to obtain a better paragraph-level alignment. Finally, the bilingual glossaries and detailed alignments were automatically generated by an extension of the *word_align* algorithm [8] to multi-word sequences. The average length of an aligned target sequence for all 30 alignments was around 1.7 words per sequence. The rate of aligned target words was within the range of 28–40%, depending on the available level of monolingual pre-processing, the relative nature of the languages of the aligned texts and some other text-specific factors (average rates presented in Table 10). Note that the matrix in Table 2 is not symmetric, as the percentage of words in language

	Target												
	de		en		es		fr		it		pt		Avg.
Source	MW	%	MW	%	MW	%	MW	%	MW	%	MW	%	(%)
de			1.45	30.2	1.43	28.0	1.48	28.5	1.43	29.1	1.30	29.4	29.0
en	1.43	32.1			1.85	36.2	1.94	37.4	1.82	37.0	1.63	37.0	35.9
es	1.35	30.2	1.75	36.5			2.00	38.5	1.90	38.4	1.72	39.1	36.6
fr	1.41	31.6	1.84	38.4	2.01	39.3			1.99	40.3	1.77	40.3	38.0
it	1.41	31.6	1.78	37.1	1.95	38.3	2.07	39.7			1.75	39.6	37.3
pt	1.25	28.1	1.57	32.7	1.73	34.0	1.80	34.5	1.72	34.8			32.8
Avg.		30.7		35.0		35.2		35.7		35.9		37.1	34.9

Table 2. Alignment coverage in million words (MW) and percents

A that are covered by parallel terms in language B is not necessarily equal to the percentage of words in B that are covered in A.

Table 2 presents the amount of aligned target words as suggested by the alignment corresponding to each source-target language pair, along with their percentage among the entire words of the target text. One can easily realize that the alignment coverage rates for German, both as source and as target language, are significantly lower than the coverage rates for other languages. This can be explained by the word agglomerations, which are very frequent in German. The separate stems comprising each agglomerated word are not given by the *TreeTagger*. The translations of each stem into other languages are often separate words. As a result, the alignment algorithm could only collect a smaller amount of reliable statistics, which justify aligning source and target sequences to each other. Another relevant example has already been given above in the comments to Figure 1.

The rates for Portuguese as source language are also relatively low. However, this has to do with the absence of a large document, as mentioned above. More specifically, the words of the discussed document, which exists in all other languages, could not be aligned with any Portuguese source words. This observation is also supported by the similarity of the ratio between coverages and the ratio between the sizes of the texts. As a target, however, the Portuguese text behaves similarly to all other texts (except for the German). That is because all existing Portuguese documents had parallels in the other languages.

As established above, the basic idea of alignment-based compression is to exploit parallelism to achieve better results compared with general-purpose, monolingual methods. Therefore, it is necessary to apply several such methods on the test corpus in order to examine the extent of improvement obtained by the various multilingual schemes.

Table 3 gives the sizes and compression rates of the outputs of 3 principal methods—Bzip2, Gzip and HuffWord for each of the languages, taken on their own.

We define the *compression rate* as the fraction, given in percent, of the size of the compressed file divided by the original size. Table 4 details the average compression rates yielded by 3 principal methods—Bzip2, Gzip and HuffWord—for

	Language												
	de		en		es		fr		it		pt		Avg.
Method	MB	%	MB	%	MB	%	MB	%	MB	%	MB	%	(%)
BZip2	4.88	17.8	4.63	17.8	4.81	17.2	4.86	17.0	4.79	17.4	4.42	18.2	17.5
GZip	6.37	23.3	5.95	22.9	6.23	22.2	6.36	22.3	6.19	22.5	5.66	23.3	22.7
HWord	5.91	21.6	6.09	23.4	6.35	22.7	6.56	23.0	6.39	23.2	5.64	23.2	22.8

Table 3. Results for monolingual methods

all 6 possible combinations of 5 languages. The first column, titled Targets, indicates the 5 texts chosen as targets for the specific row, whereas the second column gives the sixth language, acting here as Source. Table 4 thus presents monolingual compression results and is needed for a fair comparison with the performances of our algorithms on each combination of target languages. Indeed, our suggested multilingual algorithm also compresses, for our test data, a set of five texts in different languages using a single source text in a sixth language. BZip2 and GZip were first applied to the target texts, considering each as a single file (1file), then each file was split into its 3076 documents, and each fragment was compressed on its own (split). The increase in the file sizes can be immediately recognized. Note that HWord gives identical results for both settings.

		Method				
		BZip2		GZip		HWord
Targets	Source	1file	split	1file	split	
en es fr it pt	de	17.5	24.5	22.6	27.3	23.1
de es fr it pt	en	17.5	24.6	22.7	27.5	22.7
de en fr it pt	es	17.6	24.7	22.8	27.6	22.9
de en es it pt	fr	17.7	24.7	22.8	27.6	22.8
de en es fr pt	it	17.6	24.7	22.8	27.6	22.8
de en es fr it	pt	17.4	24.3	22.6	27.3	22.8

Table 4. Averages for monolingual methods

Table 5 presents the results achieved by the bilingual algorithm for each source-target pair of texts along with the average compression rates for each source and target texts and the aggregate average.

It should be noted that the numbers for the bilingual and multilingual algorithms do not include the sizes of the auxiliary files, since in the scenario of a large multilingual Information Retrieval system, dictionaries and glossaries are needed anyway and are not stored exclusively as an aid for compression. However, even if those sizes are to be considered, it should be kept in mind that, according to Heaps' Law [12], the size of a dictionary for a text of size n is expected to be αn^{β}, where $0.4 \leq \beta \leq 0.6$. The total size of the auxiliary dictionaries for the current evaluation corpus, compressed using Bzip2 (rather than a dictionary-oriented

compression scheme), is about 9% of the decompressed text. Should a 20GB corpus be compressed, the corresponding dictionaries would comprise less than 1% of the original text. Obviously, specific dictionary compression can further decrease that rate.

	Target												
	de		en		es		fr		it		pt		Avg
Src	MB	%	MB	%	MB	%	MB	%	MB	%	MB	%	(%)
de			4.92	18.9	5.27	18.8	5.41	18.9	5.26	19.1	4.63	19.1	19.0
en	4.70	17.2			4.90	17.5	5.00	17.5	4.92	17.8	4.34	17.9	17.6
es	4.85	17.7	4.71	18.1			4.95	17.3	4.84	17.6	4.22	17.4	17.6
fr	4.77	17.4	4.60	17.7	4.75	16.9			4.75	17.2	4.21	17.3	17.3
it	4.77	17.4	4.67	18.0	4.79	17.1	4.88	17.1			4.23	17.4	17.4
pt	4.93	18.0	4.88	18.8	4.97	17.7	5.14	18.0	5.01	18.2			18.1
Avg		17.6		18.3		17.6		17.8		18.0		17.8	17.8

Table 5. Results for bilingual algorithm

As stated in Section 2, the multilingual algorithm uses an annotated source text. Obviously, this annotation is not for free. Table 6 displays the overheads paid for the markup of aligned source sequences ("Al.") as well as for lemmatization indices ("Lem."). As already remarked, we used one Huffman code for text words and alignment marks (for sequence lengths 1–7), and another Huffman code for the lemma indices. Therefore, the overhead of the alignment data is calculated by subtracting the size of the HuffWord file encoding the pure text from that of the file encoding the annotated text, whereas the lemmatization overhead is simply the cost of storing the series of lemma indices using a distinct Huffman code. For the sake of briefness, we present only the average overheads in the case of a single target text. This is acceptable because the differences from the average values are very small. In addition, this impreciseness has no effect on the average compression rates.

A glance taken at the overhead table reveals that the ratio between the annotation cost for five targets ($k = 5$) and for a single target ($k = 1$) is less than 2. That is, the additional overhead for another four targets is smaller than the cost paid for the first target. Table 7 details the growing sizes of the text+alignment HuffWord files for the English text as source, when each time marks for an additional alignment are incorporated into the text. The ratio between the additional overheads for the kth and $(k-1)$th alignments is around 0.6, which leads to the thought that if some more targets were added, the additional overhead would quickly converge towards 0. Consequently, the annotation overhead in the case of large enough k's may be deemed a constant independent of k, which permits ignoring it and taking into account only the sizes of the compressed targets.

Table 8 depicts the results yielded by the multilingual algorithm for a single target text in each execution ($k = 1$). The percentages were calculated by summing

	Text	Text + 1 alignment						Text + 5 alignments					
	Size	Size	Al.		Lem.		Tot.	Size	Al.		Lem.		Tot.
S.	MB	MB	MB	%	KB	%	%	MB	MB	%	KB	%	%
de	5.91	6.43	0.52	8.81	2.83	0.05	8.86	6.77	0.85	14.46	4.53	0.07	14.54
en	6.09	6.71	0.62	10.27	8.95	0.14	10.41	7.19	1.10	18.09	14.46	0.23	18.32
es	6.35	7.00	0.65	10.25	24.29	0.37	10.62	7.52	1.17	18.38	40.47	0.62	19.00
fr	6.56	7.22	0.67	10.15	35.52	0.53	10.68	7.79	1.23	18.82	57.92	0.86	19.68
it	6.39	7.04	0.65	10.19	30.12	0.46	10.66	7.56	1.18	18.46	48.10	0.74	19.20
pt	5.64	6.23	0.59	10.38	23.14	0.40	10.78	6.70	1.06	18.77	35.29	0.61	19.38

TABLE 6. Source annotation overheads

k	Text + k alignments	Diff (MB)	Add ovrhd (%)	ratio
0	6.09			
1	6.67	0.58	9.55	
2	6.92	0.26	4.21	0.44
3	7.06	0.13	2.17	0.52
4	7.15	0.09	1.51	0.70
5	7.20	0.05	0.87	0.57

TABLE 7. Additional overhead as function of k (English text as source)

the overhead with the size of each compressed target and dividing by the size of the decompressed target.

	Avg.	TARGET												
	ovhd.	de		en		es		fr		it		pt		Avg.
SRC.	(MB)	MB	%	MB	%	MB	%	MB	%	MB	%	MB	%	(%)
de	0.52			4.50	19.3	4.85	19.2	4.98	19.3	4.84	19.5	4.25	19.7	19.4
en	0.63	4.26	17.9			4.38	17.9	4.47	17.9	4.40	18.3	3.88	18.6	17.7
es	0.67	4.38	18.5	4.16	18.6			4.37	17.7	4.27	18.0	3.73	18.1	17.6
fr	0.70	4.29	18.2	4.04	18.3	4.16	17.3			4.17	17.7	3.68	18.0	17.2
it	0.68	4.30	18.2	4.13	18.5	4.22	17.5	4.31	17.5			3.72	18.1	17.4
pt	0.61	4.50	18.6	4.38	19.2	4.46	18.1	4.61	18.3	4.49	18.5			18.2
Avg.			18.3		18.8		18.0		18.1		18.4		18.5	17.9

TABLE 8. Results for multilingual algorithm with a single target ($k = 1$)

Table 9 exhibits the performances of the multilingual algorithm for $k = 5$. Note that the sizes of the compressed targets are slightly larger compared with their respective counterparts for $k = 1$. That is because the integration of all five sets of aligned source sequences into one list increases some of the offset values used in each compressed target. However, as already stated, the annotation overhead is paid only once for the entire corpus. The compression rates were computed by accumulating the overhead and the sizes of the five compressed targets and then dividing the result by the sum of the sizes of the five decompressed targets. The results for $k = 1$ were slightly worse than those achieved by the bilingual algorithm

(averages given in Table 10). This had been quite expected in light of the high overhead paid for a stand-alone alignment.

The rightmost column of Table 9 shows the compression rate when overheads are excluded. For $k = 5$, the average overhead constitutes about 0.8% of the entire decompressed corpus. As Table 7 hints, overhead is unlikely to grow significantly even for a sixth target, so we may already deem it maximal. Hence, for $k = 23$, the current number of languages in the European Union, the overhead's relative part may drop below 0.2%.

	Oh.	TARGET						Inc. Oh.	Exc. Oh.
SRC.	(MB)	de	en	es	fr	it	pt	(%)	(%)
de	0.86		4.58	4.92	5.05	4.91	4.32	18.3	17.7
en	1.11	4.38		4.50	4.60	4.53	3.99	17.0	16.2
es	1.21	4.51	4.30		4.48	4.39	3.83	17.0	16.1
fr	1.29	4.42	4.19	4.28		4.29	3.79	16.7	15.7
it	1.23	4.43	4.27	4.34	4.42		3.83	16.8	15.9
pt	1.09	4.62	4.51	4.56	4.71	4.60		17.5	16.7
Avg.								17.2	16.4

TABLE 9. Results for multilingual algorithm (with $k = 5$)

	Alignment	Bilingual	Multilingual			BZIP2		GZIP		HWORD
SRC	Coverage		$k = 1$	$k = 5$	$k \to \infty$	1file	split	1file	split	
de	29.0	19.0	19.4	18.3	17.7	17.5	24.5	22.6	27.3	23.1
en	35.9	17.6	17.7	17.0	16.2	17.5	24.6	22.7	27.5	22.7
es	36.6	17.6	17.6	17.0	16.1	17.6	24.7	22.8	27.6	22.9
fr	38.0	17.3	17.2	16.7	15.7	17.7	24.7	22.8	27.6	22.8
it	37.3	17.4	17.4	16.8	15.9	17.6	24.7	22.8	27.6	22.8
pt	32.8	18.1	18.2	17.5	16.7	17.4	24.3	22.6	27.3	22.8
Avg	34.9	17.8	17.9	17.2	16.4	17.5	24.6	22.7	27.5	22.8

TABLE 10. Result summary

Finally, Table 10 puts side-by-side the average alignment coverage rates and the compression rates achieved by the monolingual, bilingual and multilingual algorithms. For the monolingual methods, the term "Source" refers to the text *not* taken into average account (equivalent to the SOURCE column in Table 4). The column labeled "$k \to \infty$" includes the same data presented in the "Exc. Oh." column of Table 9. Indeed, the compression rate which could be obtained for a large number of targets with compressibility similar to that of the current test texts converges to the rate computed excluding the source annotation overhead. That is because for large k's, this overhead converges towards a relatively small constant and may therefore be neglected.

A close observation into the results would recognize a tight correlation between alignment coverage and compression rates. This is, of course, expected, because each aligned target sequence is replaced with a reference, which is shorter, in average, than the corresponding HuffWord encoding. Hence, the higher the coverage rate, the better the compression. It may be assumed that if the German text could be stemmed before being aligned, we would get even better results. In general, if the alignment algorithm yielded denser outputs, the results could improve significantly.

BZIP2 does not perform as well for small blocks as it does for large blocks, as opposed to Huffman coding, which is indifferent to block size. Therefore, in cases where extraction of relatively small pieces is desired, even the bilingual scheme with Huffman coding would be preferable over regular BZIP2, not to mention the other two monolingual methods. Results also show that it is unworthy using the multilingual scheme rather than the bilingual one for small k. However, for large enough k, the multilingual algorithm is advantageous even if each target is to be stored in a bulk, rather than split into pieces. In that situation, of course, the encoding method of the compression procedure's output may be changed from HWORD to BZIP2, which is expected to yield a further improved compression.

4. Conclusion

The current work suggests the first specific methods for compressing multilingual parallel texts, based on text alignment. The algorithm has been tested on a real-life multilingual corpus and achieved significant improvements over general-purpose methods.

A prominent advantage of the compression scheme is its static nature, which enables applying it to blocks of any size without changing compression efficiency. This property is particularly important for Information Retrieval systems, where users are frequently interested in relatively small pieces of texts. Compressing each small piece *per se* permits transferring and deciphering the desired piece only, thereby saving a lot of expensive communication and processing time.

For example, many IR systems present, in response to a query, a list of snippets or so-called KWIC (Key Word In Context) excerpts. Each such KWIC shows the keywords of the query and in addition a few words preceding and following them in the retrieved locations. Since the access to these locations is by means of pointers, there is a need to allow the decompression of relatively small text fragments. This is not possible or would require a prohibitive overhead with adaptive compression methods like LZ, or even with BZIP which would then be applied with small blocks.

The tight correlation between alignment coverage and compression efficiency calls for further improvement of alignment methods as well as some monolingual tools, the quality of which has a crucial effect on alignment coverage rate. Some other related work can be done on specific methods for compressing multilingual

dictionaries. Recall that many IR systems hold dictionaries for various purposes and, in large systems, their sizes are negligible anyway. However, a significant decrease in those sizes could encourage small- and medium-scale systems which do not need dictionaries for purposes other than deciphering, to use the methods yielded by this research.

Acknowledgment: The first author would like to express his deep gratitude for Drs. Mordecai & Monique Katz and the Mozes S. Schupf Fellowship Program for their support for his work on this research.

References

[1] J. Adiego, N. R. Brisaboa, M. A. Martínez-Prieto, and F. Sánchez-Martínez: *A two-level structure for compressing aligned bitexts*, in SPIRE, 2009, pp. 114–121.

[2] Ahrenberg, L., Andersson, M., and Merkel, M.: *A knowledge-lite approach to word alignment*, in Parallel Text Processing, J. Véronis, ed., Kluwer Academic Publishers, Dordrecht, 2000, pp. 97–116.

[3] Ajtai, M., Burns, R. C., Fagin, R., and Long, D. D. E.: *Compactly encoding unstructured inputs with differential compression*. Journal of the ACM, 49(3) 2002, pp. 318–367.

[4] Brown, P. F., Della Pietra, S., Della Pietra, V. J., and Mercer, R. L.: *The mathematics of statistical machine translation: parameter estimation*. Computational Linguistics, 19(2) 1993, pp. 263–311.

[5] Burns, R. C. and Long, D. D. E.: *Efficient distributed backup and restore with delta compression*, in Workshop on I/O in Parallel and Distributed Systems (IOPADS), ACM, 1997.

[6] E. S. Conley and S. T. Klein: *Compression of multilingual aligned texts*, in DCC, 2006, p. 442.

[7] E. S. Conley and S. T. Klein: *Using alignment for multilingual text compression*. Int. J. Found. Comput. Sci., 19(1) 2008, pp. 89–101.

[8] Dagan, I., Church, K. W., and Gale, W. A.: *Robust bilingual word alignment for machine-aided translation*, in Proc. of the Workshop on Very Large Corpora: Academic and Industrial Perspectives, Columbus, Ohio, 1993, pp. 1–8.

[9] *EUR-Lex*: http://eur-lex.europa.eu/.

[10] P. Fung and K. Mckeown: *Aligning noisy parallel corpora across language groups: Word pair feature matching by dynamic time warping*, in In Proceedings of the First Conference of the Association for Machine Translation in the Americas, 81–88, 1994, pp. 81–88.

[11] Gaussier, É., Hull, D., and Aït-Mokhtar, S.: *Term alignment in use : Machine-aided human translation*, in Parallel Text Processing, J. Véronis, ed., Kluwer Academic Publishers, Dordrecht, 2000, pp. 253–274.

[12] J. Heaps: *Information Retrieval : Computational and Theoretical Aspects*, Academic Press, Inc., New York, NY, 1978.

[13] M. A. Martínez-Prieto, J. Adiego, F. Sánchez-Martínez, P. de la Fuente, and R. C. Carrasco: *On the use of word alignments to enhance bitext compression*, in DCC, 2009, p. 459.

[14] A. Moffat: *Word-based text compression*. Software — Practice & Experience, 19 1985, pp. 185–198.

[15] Moffat, A. and Zobel, J.: *Adding compression to a full-text retrieval system*. Software — Practice & Experience, 25(8) 1995, pp. 891–903.

[16] Nevill, C. and Bell, T.: *Compression of parallel texts*. Information Processing & Management, 28 1992, pp. 781–793.

[17] *TreeTagger – a language-independent part-of-speech tagger*: http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.

[18] Witten, I. H., Moffat, A., and Bell, T. C.: *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, New York, 1994.

Ehud S. Conley
Department of Computer Science
Jerusalem College of Technology
Jerusalem 91160, Israel
e-mail: conley@jct.ac.il

Shmuel T. Klein
Department of Computer Science
Bar Ilan University
Ramat Gan 52900, Israel
e-mail: tomi@cs.biu.ac.il