Natural Language Engineering **3** (4): 255–277. Printed in the United Kingdom

© 1997 Cambridge University Press

Example retrieval from a translation memory

LAMBROS CRANIAS, HARRIS PAPAGEORGIOU
and STELIOS PIPERIDIS
Institute for Language and Speech Processing (ILSP), 22 Margari str., 115 25 Athens, Greece

(*Received 8 August 1994; Revised 20 February 1997*)

Abstract

Clustering of a translation memory is proposed to make the retrieval of similar translation examples from a translation memory more efficient, while a second contribution is a metric of text similarity which is based on both surface structure and content. Tests on the two proposed techniques are run on part of the CELEX database. The results reported indicate that the clustering of the translation memory results in a significant gain in the retrieval response time, while the deterioration in the retrieval accuracy can be considered to be negligible. The text similarity metric proposed is evaluated by a human expert and found to be compatible with the human perception of text similarity.

1 Introduction

Large-scale realistic Natural Language Processing (NLP) applications usually presuppose a massive knowledge acquisition about language, collected in computational grammars, lexicons and domain models. To avoid this often prohibitive task, two non-traditional approaches to the important NLP application of Machine Translation (MT) have appeared in recent years:

- Statistical MT, which estimates probability distributions of words, based on very large aligned bilingual text corpora, and then uses these estimates to produce the maximum-likelihood translation of an input sentence.
- Example-Based Machine Translation (EBMT), which is often realised as a translation aid, and constitutes a more modest approach.

EBMT is based on the idea of performing translation by imitating translation examples of similar sentences (Nagao, 1984). In this type of translation system, a large number of bi/multi-lingual translation examples have been stored in a textual database (translation memory) and input expressions are rendered in the target language by first retrieving from the database the example which is most similar to the input, and then processing this example.

There are three key issues which pertain to example-based translation:

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

- establishment of correspondences between units in a bi/multi-lingual text at sentence, phrase or word level;
- a mechanism for retrieving from the database the unit that best matches the input;
- exploiting the retrieved translation example to produce the actual translation of the input sentence.

Many authors (Brown, 1991; Gale and Church, 1991a; Kay and Roescheisen, 1993; Simard *et al.*, 1992; Papageorgiou *et al.*, 1994) have proposed methods for establishing correspondences between sentences in bilingual corpora. These methods are purely statistical with little or none linguistic information (which is limited to cognates or part-of-speech (pos) labels) and achieve suprisingly high success rates (around 97%). Several others (Brown *et al.*, 1993; Kay and Roescheisen, 1993; Dagan *et al.*, 1993; Kumano and Hirakawa, 1993; Gale and Church, 1991b; Sadler and Vendelmans, 1990; Kaji *et al.*, 1992) have tackled the problem of establishing correspondences between words and phrases in bilingual texts. The establishment of correspondences below sentence level is, however, a very difficult task, and the degree to which it can support practical applications such as EBMT depends upon the actual texts to be processed.

The third key issue of EBMT, that is exploiting the retrieved translation example, is sometimes dealt with by integrating into the system conventional MT techniques (Kaji *et al.*, 1992; Sumita and Tsutsumi, 1988). Simple modifications of the translation proposal, such as word substitution, would also be possible, provided that alignment of the translation archive at word level was available.

In establishing a mechanism for the best match retrieval, which is the topic of this work, the crucial tasks are:

1. determining whether the search is for matches at sentence or subsentence level, that is determining the 'text unit',
2. the definition of the metric of similarity between two text units, and
3. search space reduction.

As far as (1) is concerned, the obvious choice is to use as the text unit the sentence. This is because sentence boundaries, if not unambiguous, are at least manageable through a text handling process. Sentences can, however, be quite long. And the longer they are, the less possible it is that they will have an exact match in the translation archive, and the less flexible the EBMT system will be.

On the other hand, if the text unit is the subsentence we face one major problem, that is, the possibility that the resulting translation of the whole sentence will be of low quality, due to boundary friction and incorrect chunking. In practice, EBMT systems that operate at subsentence level involve the dynamic derivation of the optimum length of segments of the input sentence by analysing the available parallel corpora. This requires a procedure for determining the best 'cover' of an input text by segments of sentences contained in the database (Nirenburg *et al.*, 1993). It is assumed that the translation of the segments of the database that cover the input sentence is known. What is needed, therefore, is a procedure for

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

aligning parallel texts at subsentence level. If subsentence alignment is available, the approach is fully automated, but is quite vulnerable to the problem of low quality, as mentioned above, as well as to ambiguity problems when the produced segments are rather small.

Despite the fact that many running EBMT systems employ the sentence as the text unit, it is believed that the potential of EBMT lies in the exploitation of fragments of text smaller than sentences and the combination of such fragments to produce the translation of whole sentences (Sadler and Vendelmans, 1990). Automatic subsentential alignment is, however, a problem yet to be solved.

Turning to the definition of the metric of similarity, the requirement is usually twofold. The similarity metric applied to two sentences (by 'sentence' from now on we will refer to both sentence and subsentence fragments) should indicate how similar the compared sentences are, and perhaps the parts of the two sentences that contributed to the similarity score. The latter could be just a useful indication to the translator using the system, or a crucial functional factor of the system, as explained later.

The similarity metrics reported in the literature can be characterised depending on the text patterns on which they are applied. So, the word-based metrics compare individual words of the two sentences in terms of their morphological paradigms, synonyms, hyperonyms, hyponyms, antonyms, pos tags... (Nirenburg *et al*., 1993). Each such category is assigned a weight which is applied as a multiplying coefficient to the count of the corresponding category. Then, summation over all categories produces the similarity score of the two sentences under comparison. Another approach is the use of a semantic distance d ($0 \preceq d \preceq 1$) which is determined by the Most Specific Common Abstraction (MSCA) obtained from a thesaurus abstraction hierarchy (Sumita and Iida, 1991). The score assigned to the comparison of two words reflects their proximity in the thesaurus abstraction hierarchy. Then, a similarity metric is devised, which evaluates the similarity of two sentences, by combining the individual contributions towards similarity stemming from word comparisons.

The word-based metrics are the most popular, but other approaches include syntax-rule driven metrics (Sumita and Tsutsumi, 1988), character-based metrics (Sato, 1992) as well as some hybrids (Furuse and Iida, 1992). The character-based metric has been applied to Japanese, taking advantage of certain characteristics of Japanese. The syntax-rule driven metrics try to capture the similarity of two sentences at the syntax level. This seems very promising, since similarity at the syntax level, perhaps coupled with lexical similarity in a hybrid configuration, would be the best the EBMT system could offer as a translation proposal. The real time feasibility of such a system is, however, questionable, since it involves the complex task of syntactic analysis. However, as Nirenburg *et al*. (1993) argue, "for EBMT to succeed, it should be shown not to rely on an extensive apparatus of linguistic and domain-oriented language analysis, which forms the basis of the 'traditional' rule-based MT, which EBMT set out to supplant". A way round the problem of syntactic analysis is proposed in this work by the definition of a similarity metric which is based on both limited surface structure (as captured by the functional

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

words of a sentence) and content (part-of-speech and lemmas of the non-functional words).

Coming to the third task involved in a matching mechanism, that is search space reduction, we notice that for an EBMT application to be useful its response time should be moderate allowing for real time applications. This means that the search space for the input sentence should, somehow, be limited. The usual method employed to limit the search space is to use full-text retrieval based on content words. The problem with this approach, or any other similar approach, is that the criterion applied to limit the search space has little (or even nothing) to do with the actual similarity metric which will then be applied to locate the best available match in the corresponding limited part of the corpus. Hence, these methods are not at all optimal, especially if the translation archive contains long sentences, and many best matches are missed. To overcome this problem a new approach is presented, namely clustering.

The work presented in this paper constitutes the evolution of the work presented by Cranias *et al.* (1994). Furthermore, the methods employed are described here in every detail and evaluation results as well as examples are thoroughly discussed. In section 2 the similarity metric proposed is described in detail. The statistical system presented consists of two phases, the Learning and the decision making or Recognition phase, both described in section 3. Both phases are based on statistical pattern recognition techniques. In the Learning phase the sentences of the corpus are manipulated using a specialised k-means algorithm to determine a reference model of the system. In the Recognition phase, the decision is taken by comparing a new input sentence against the reference model. Finally, section 4 contains the results and their evaluation.

2 Similarity metric

To encode a sentence into a vector, we exploit information about the Functional Words/phrases (FW) appearing in it, as well as about the lemmas and part-of-speech (pos) tags of the words appearing between FWs/phrases. Based on the combination of FWs/phrases data and pos tags, a simple view of the surface syntactic structure of each sentence is obtained.

To identify the FWs/phrases in a given corpus, the following criteria are applied:

- FWs introduce a syntactically standard behaviour,
- most of the FWs belong to closed classes,
- the semantic behaviour of FWs is determined through their context,
- most of the FWs determine phrase boundaries,
- FWs have a relatively high frequency in the corpus.

According to these criteria, prepositions, conjunctions, determiners, pronouns, certain adverbials etc. are regarded as FWs.

Having identified the FWs of the corpus, we distinguish *groups of FWs* on the basis of their interchangeability in certain phrase structures. For example, the sentences:

- **To** create **a** file, press F.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

- **In order to** create **a** file, press F.

demonstrate that the FW **to** and the functional phrase **in order to** should be grouped together. The grouping also caters for the multiplicity of usages of a certain word which has been identified as a FW, since a FW can be a part of many different groups. For instance, the following groups are obtained in English:

- Relatives: that, whom, which,...
- Determiners: that, the, this, a,...
- Concessive Conjunctions: while, though, whereas,...

In this way, FWs can serve the retrieval procedure with respect to the following levels of contribution towards the similarity score of two sentences:

- Identity between FWs of retrieved candidate and input (**I**).
- FWs of retrieved candidate and input not identical but belonging to the same group (**G**).

To obtain the lemmas and pos tags of the remaining words in a sentence, we use a part-of-speech Tagger with no disambiguation module (we make use of a morphological lexicon), since this would be time consuming and not 100% accurate. Instead, we introduce the concept of **ambiguity class**, and represent each non-FW by its ambiguity class and the corresponding lemma(s) (for example, the unambiguous word 'eat' would be represented by the which is the set verb and the lemma 'eat') (in English, for an ambiguous word, the corresponding lemmas will usually be identical. But this is rarely true for Greek). Hence, the following two levels of contribution to the similarity score stem from non-FWs:

- overlapping of the sets of possible lemmas of the two words (**L**),
- overlapping of the ambiguity classes of the two words (**T**).

Hence, each sentence of the source part of the translation archive is represented by a pattern, which is expressed as an ordered series of the above-mentioned feature components. For example, the sentence: "**The** implication **of this is** obvious" would be encoded as something like: < FW1 pos1 lemma1 FW2 FW3 FW4 pos2 lemma2 >.

A similarity metric is defined between two such vectors, and is used in both the Learning and Recognition phases. Comparing a test vector against a reference vector is, however, not straightforward, since there are generally axis fluctuations between the vectors (vectors are not necessarily aligned and are of most probably different lengths). To overcome these problems we use a two-level **Dynamic Programming (DP)** technique (Sakoe and Chiba, 1978; Ney, 1984). The first level treats the matches at the FW level, while the second is reached only in case of a match in the first level, and is concerned with the lemmas and tags of those words within FW boundaries. Both levels utilise the same (DP) model, which is next described.

We have already referred to the (I) and (G) contributions to the similarity score due to FWs. But this is not enough. We should also take into account whether the FWs appear in the same order in the two sentences, whether an extra (or a few) FW

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

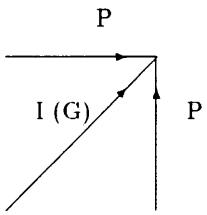


Fig. 1. The DP allowable transitions.

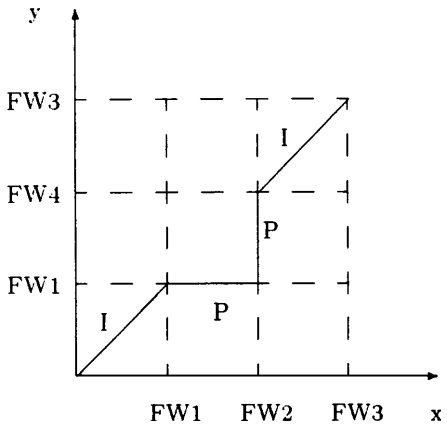


Fig. 2. An example of the application of the DP algorithm.

intervenes (‘**addition**’) in one of the two sentences, whether certain FWs are missing (‘**deletion**’)... To deal with these problems, we introduce a third contribution to the similarity score, which is negative and is called the **penalty score (P)**. Therefore, when two FWs match (either identically or just belong to the same group) the I or G score is assigned to this match and the accumulated score propagates diagonally on the xy-plane, where x stands for the direction of development of one sentence and y of the other. When the two FWs mismatch, the P score (which is negative) is assigned to them and the accumulated score propagates along a horizontal or vertical direction. In Figure 1 the allowable transitions in the xy-plane are shown.

An example is given to clarify the DP model. In Figure 2 one of the possible paths (actually the best path) is shown. FW1 is common in the two sentences, so the path starts with a diagonal transition and score I. Then FW2 does not match with FW4 in the other sentence, so we have an FW intervention and hence a negative local score, P, across the horizontal. The accumulated score is so far I+P. In the same way, FW4 intervenes, so now the score propagates (with the ‘addition’ of P) along the vertical direction. Finally, FW3 matches, so we have a diagonal transition with the addition of the local score I.

Whenever a diagonal transition is investigated, the system calls the second level DP-algorithm which produces a local additional score due to the potential similarity of lemmas and tags of the words lying between the corresponding FWs. This score is calculated using exactly the same DP-algorithm as that treating FWs (allowing

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

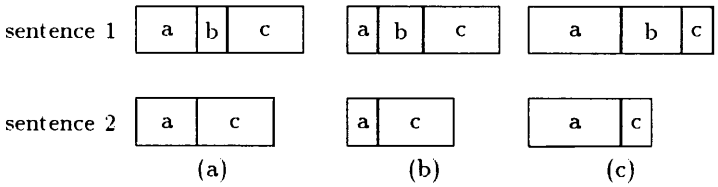


Fig. 3. Examples of matched sentences.

additions, deletions,...), provided that we use (**L**), (**T**) and (**PT**) (a penalty score attributed to a mismatch at the tag-level) in place of (**I**), (**G**) and (**P**), respectively.

The (DP) model, which produces the optimal solution in terms of I, G, P, L, T and PT, is described formally in the Appendix.

The outcome of the DP-algorithm is the similarity score between two vectors, as well as the second best similarity scores emerging from different parts of the two vectors. Moreover, the algorithm determines, through the backtracking procedure, the relevant parts of the two vectors that contributed to this score. This is essential for the sentence segmentation described in the next section.

The calculation of the similarity score allows for different lengths of the two sentences, the similarity of different parts of the two sentences (last part of one with the first part of the other), and finally, a variable number of additions and deletions. The score produced corresponds to two *coherent* parts of the two sentences under comparison. Emphasis should be given to the variable number of additions and deletions. The innovation of the penalty score (which is in fact a negative score) provides the system with the flexibility to afford a different number of additions or deletions depending on the accumulated similarity score up to the point where they start. For example, in Figure 3(a), the algorithm will combine parts a and c to produce the similarity score (reduced of course because of the intervention of b), while in 3(b) it might not, because part a is now small compared to part b. Similar remarks hold for 3(c), where part a is indeed big enough to allow the intervention of part b, which is, however, rejected because of the fact that part c is small and will not carry the score propagated from part a at a level higher than that achieved at the end of part a.

It should also be noted that the similarity score produced is based mainly on the surface syntax of the two sentences (as this is indicated by the FWs and pos tags), and in the second place on the actual words of the two sentences. This is quite reasonable, since the two sentences could have almost the same words in the source language, but no similarity at all in the source or target language (due to different word order), while if they are similar in terms of FWs as well as in terms of the pos tags of the words between FWs, then the two sentences would almost certainly be similar (irrelevant of a few differences in the actual words) in the target language as well (which is the objective).

The DP-algorithm proposed seems to be tailored to the needs of the similarity metric, but there is yet a crucial set of parameters to be set, that is A = I, G, P, L, T, PT. The DP-algorithm is just the framework for the utilisation of these parameters.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

When comparing two sentences, the first constraint is that the similarity score is 100% if the two sentences are identical. Hence, the set A should not be fixed, but rather should take on values normalised by the lengths of the two sentences under comparison. Next, we analyse each entry of A in turn.

- **I**: identical FWs

We assign a fixed percentage to the match of FWs, called **PFW**, while the remaining percentage, that is (100-PFW), will be assigned to matches at lemma and pos tag level. Hence, I is defined as

$$I = \frac{PFW}{average\ length\ in\ FWs}.$$

To account for the most probably different length of the two sentences under comparison, normalisation is based on the average length in FWs of the two sentences.

- **G**: FWs belonging to the same group

This case is identical to the previous one, with the only difference lying on a **weight (W)** which is assigned to 'group similarity' of FWs. Hence, G is defined as

$$G = \frac{(PFW) \cdot (W)}{average\ length\ in\ FWs}.$$

- **P**: penalty score due to a mismatch of FWs

We want P (as well as PT) to reflect the user's choice of penalising an addition or deletion of a functional word (or non functional word when refering to PT). Thus, two new parameters are involved in the definition of P. First, **THRESHOLD (TH)** which is a lower acceptable (presentable) similarity score (could be set by the user), and then **APFW**, which is the percentage of the total number of FWs which the system can afford to mismatch. That is, if up to (APFW)* (average length in FWs) FWs mismatch between two fragments producing the similarity score and the rest FWs of the two fragments appear in identical order, then the total similarity score will be above the TH. Hence, P is defined as

$$P = \frac{(PFW) \cdot (100 - TH)}{(APFW) \cdot (average\ length\ in\ FWs)}.$$

- **L**: overlapping of the sets of possible lemmas of the two words

Using the same reasoning as in I, L is defined as

$$L_{ij} = \frac{100 - PFW}{(1 + average\ length\ in\ FWs) \cdot (longest\ non\ FW\ sequence)_{ij}}.$$

$1 \preceq i \preceq 1 + length\ in\ FWs\ of\ the\ first\ sentence$
$1 \preceq j \preceq 1 + length\ in\ FWs\ of\ the\ second\ sentence$

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

Notice that L changes value inside each block of non FWs, and is normalised by the longer of the two non-FW blocks under comparison. This is to account for any difference in the number of the non-FWs belonging to each of the blocks under comparison.

- **T**: overlapping of the ambiguity classes of the two words

In a similar manner to G, T is defined as

$$T_{ij} = \frac{(100 - PFW) \cdot (W)}{(1 + average\ length\ in\ FWs) \cdot (longest\ non\ FW\ sequence)_{ij}}.$$

$1 \preceq i \preceq 1 + length\ in\ FWs\ of\ the\ first\ sentence$
$1 \preceq j \preceq 1 + length\ in\ FWs\ of\ the\ second\ sentence$

- **PT**: penalty score due to mismatch of the pos tags of non-FWs

In a similar manner to P, PT is defined as

$$PT_{ij} = \frac{(100 - PFW) \cdot (100 - TH)}{(APOS) \cdot (1 + average\ length\ in\ FWs) \cdot (longest\ non\ FW\ sequence)_{ij}}.$$

$1 \preceq i \preceq 1 + length\ in\ FWs\ of\ the\ first\ sentence$
$1 \preceq j \preceq 1 + length\ in\ FWs\ of\ the\ second\ sentence$

where APOS is used instead of APFW to refer to the percentage of the total number of non-FWs in block i and j of the two sentences which the system can afford to mismatch.

Thus, the values of set A are determined dynamically depending on the sentences under comparison. The parameters that are fixed (that is, PFW, W,...) are those influencing the efficiency of the similarity metric. To determine optimal values of these parameters, a thorough trial and error investigation is suggested, with the results evaluated by professional translators, who should have the last word in fine-tuning the similarity metric. What is important, however, is that small fluctuations from the 'optimal' set of values of A do not have a great influence on the similarity metric. Typical values are next suggested:

PFW = 40, W = 0.3, TH = 70, APFW = 0.3, APOS = 0.2

3 Learning and recognition phases

For the search space to be effectively limited, the translation archive should be clustered into groups of sentences (*clusters*), where each group is represented by its centre, which is just a sentence of the archive. A sentence is chosen as the cluster centre if it exhibits high enough similarity with all the other sentences in the cluster, and can therefore effectively 'represent' those sentences. Then, a sentence of the translation archive can become a best matching candidate for the input sentence

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

only if the latter exhibits a high enough similarity with the corresponding centre of the cluster.

Clustering is a well studied subject of pattern recognition, and several algorithms have been proposed. That implemented in this work is the modified k-means (MKM) algorithm proposed by Wilpon and Rabiner (1985), which has been successfully applied to the problem of isolated word speech recognition. It produces the optimum segmentation of the corpus into clusters (based on the actual similarity metric employed), and determines each cluster centre by using the minmax criterion Wilpon and Rabiner (1985). The MKM-algorithm starts with all the sentences allocated to one cluster, and proceeds by splitting at each step the least homogeneous cluster (the cluster with the minimum average similarity score of the sentences to the cluster centre) and then reallocating all sentences. The final number of clusters can be automatically determined subject to some cluster quality constraint, but is usually determined externally based upon speed requirements.

However, the clustering of sentences exhibits a certain peculiarity. No matter which similarity metric is employed, the actual score produced from the comparison of two vectors of most probably different lengths, corresponding to two sentences, will have contributions emerging from certain parts of the two vectors, while the remaining parts will not contribute to the final score. This case is different to the case of, say, speech recognition where the frames of speech are of a specific length, and therefore the entries of the corresponding vectors are aligned and all contribute to the produced score of similarity. The implication of this particularity is quite obvious. It is highly probable (especially for long sentences) that a sentence is allocated to a cluster centre because of a good match due to a part of it, while the remaining part has nothing to do with the cluster to which it will be allocated. Hence, this part will remain hidden to an input sentence applied to the system at the recognition phase. This situation is graphically depicted in Figure 4.

To remedy this situation, which is quite frequent due to the fact that sentences tend to be quite long, an iterative procedure is proposed, and is depicted in Figure 5.

Once the clustering of the initial corpus has been completed, the segmentation phase is applied, that is, we search for 'hidden' parts of the sentences. Hence, for each sentence allocated to a cluster its hidden parts, if any, are located and matched against the sentences allocated to the other clusters. If good enough matches are traced, the relevant sentences are segmented and then the updated corpus is reclustered. After a number of iterations, convergence is obtained (no new sentence segments are created), and the whole clustering procedure is terminated.

This clustering set-up works nicely to speed up the search for the best available match in a translation archive. However, if we are to exploit the located best match in an EBMT environment, it is clear that we should know its translation as well. If the located best match is a whole sentence we do know its translation, thanks to automatic sentence alignment which is rather successful. But to know the translation of a sentence segment would require alignment below sentence level. Automatic subsentential alignment, however, is not yet mature enough to produce high fidelity results. Hence, one resorts to the use of semi-automatic methods (in our

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

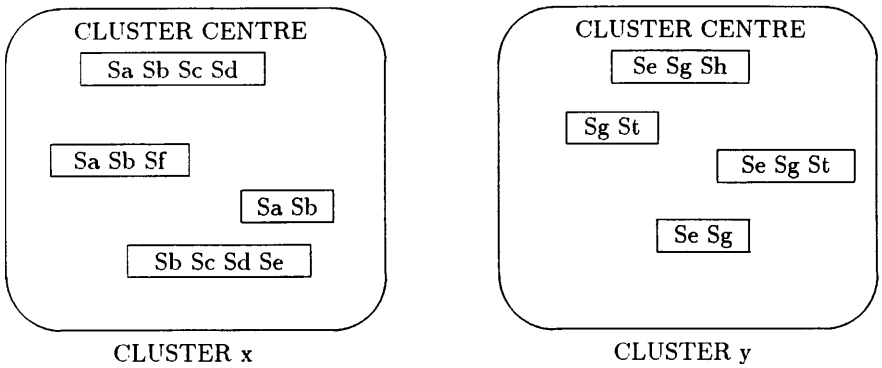


Fig. 4. An example of sentences allocated to clusters. Si stands for a segment i or for any other segment "similar enough" to segment i. In cluster x there exist two 'hidden' segments, that is, Se and Sf, while in cluster y the only hidden segment is St. Take, for example, segment St. Since it only appears in cluster y and always next to Sg, a good guess is that in an input sentence it will again appear next to Sg and will therefore be located through Sg. On the other hand, Se in cluster x is indeed hidden and since it also appears in sentences allocated to other clusters, sentences containing it should be further segmented.

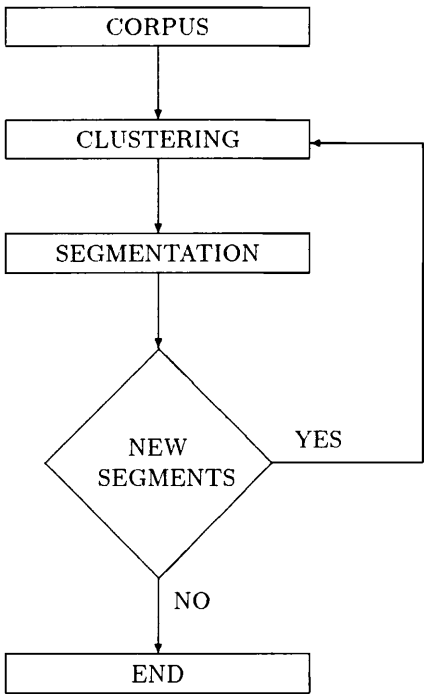


Fig. 5. The iterative clustering procedure.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

application with the CELEX database – the computerised documentation system on community law – because of the strict format in which the text appears, a rough segmentation of the sentences is straightforward and can therefore be automated). So, to locate the hidden part(s) of a sentence, we first expand the limits of the part that contributed to the score with the cluster centre to cover a 'translatable unit' (as indicated by the subsentential alignment) and then, if needed, segment the sentence.

Instead of using this iterative method of clustering, we could right from the start segment all the 'translatable units' of the corpus, and then apply a single run of the clustering algorithm. The reason we are not doing this is because in a specific corpus quite a lot of these segments tend to appear side-by-side in a sentence. Hence, by clustering whole sentences and then segmenting only in the case of a good match with a part of a sentence allocated to a *different* cluster, we can avoid the overgeneration of clusters and segments. When the iterative procedure has converged, the sentences of the original corpus will have been segmented to 'translatable units' in an optimum way, so that they are efficiently represented by a set of sentences (or sentence segments) which are the cluster centres.

On the other hand, if alignment at subsentential level is not available, the segmentation of the sentences of the corpus is pointless, in the sense that even if a good enough match with a segment in the translation memory is located, its translation will not be known (of course, one can argue that even if we do not know the translation of the segment, we do know the whole sentence which contains this translation, and can therefore offer it to the translator who will easily locate the part he/she is interested in). Then, the degree of success of the Learning phase depends on the length of the sentences. The longer they are, the less successful the Learning phase.

In the Recognition phase, the input sentence is first compared to the cluster centres and then to the sentences allocated to the favourite cluster. Thus, the search space is effectively limited. However, there is yet a problem to be solved. To determine the 'translatable units' in a sentence, its translation needs to be known. This is because the subsentential alignment is based on criteria applied across the two languages under investigation. Therefore, we cannot know what the 'translatable units' of the input sentence are. We only have potential 'translatable unit' markers. This is not really a restriction, however, since by setting a high enough threshold for the match with a segment (translatable piece of text) in the corpus, we can be sure that the part of the input sentence that contributed to this good match will also be translatable and we can, therefore, segment this part. This process continues until the whole input sentence has been 'covered' by sentences (or sentence segments) of the corpus.

4 The application – evaluation

The development of the matching method presented was part of the research work conducted under the LRE I project TRANSLEARN. The aim of the project, which ran until mid-1995, was to provide a computational methodology and, in more practical terms, a toolbox which will aid the human translator working in a particular sublanguage in the following ways:

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

- relieve him/her from the repetitive part of his/her work, mostly dealing with specialised types of text;
- enhance productivity and translation quality by assisting him/her through proposed alternative solutions, as well as by providing sophisticated ancillary tools.

The project considers four languages: English, French, Greek and Portuguese.

The application on which we are developing and testing the method is implemented on the Greek-English language pair of records of the CELEX database, the computerised documentation system on Community Law, which is available in all Community languages. The matching mechanism is, so far, implemented on the Greek part, providing English translation proposals for Greek input sentences.

The sentences contained in the CELEX database tend to be quite long, but due to the strict format in which they appear (corresponding to articles, regulations,...), we were able to provide the Learning phase with some potential segmentation points of these sentences in both languages of the pair (these segmentation points are in one-to-one correspondence across languages, yielding the 'subsentence alignment').

In tagging the Greek part of the CELEX database, we came across 35 different ambiguity classes, which are utilised in the matching mechanism. The identification and grouping of the Greek FWs was mainly done with the help of statistical tools applied to the CELEX database.

We tested the system on 50 files of the CELEX database, i.e. 3000 sentences. We implemented two versions: one working with whole sentences and not employing any segmentation phase; and one working with sentence segments when this is needed (see the examples in section 4.1, corresponding to the case of the system working with segments, i.e. during the Learning phase, the sentences of the translation memory were segmented and clustered, and during the Recognition phase, the input sentence can be split into potential segments). If alignment at the subsentential level could be automated while being successful, we would only care to test the version of the system working with segments. But since the subsentential alignment in our system is semi-automatic and tailored to the CELEX database, the evaluation of the overall translation results is quite difficult, since we do not know how much of the accuracy comes from judicious selection of segment boundaries.

The number of clusters employed by the system also affects the efficiency of the proposed method. To investigate the effect of the number of clusters on the efficiency of the system, we tested three versions in each case. These correspond to the number of clusters fixed to 1%, 2% and 3% of the number of sentences in the database. In the case of the system working with segments, the Learning phase resulted in 3554 3710 and 3748 'segments', respectively. We used as test suite 200 sentences of the CELEX database which were not included in the translation archive. In the Recognition phase, for the case of segments, the system proposed translations for 214, 228 and 229 'segments' respectively.

To evaluate this translation aid system is, however, not a straightforward matter. There can be no absolute metric indicating the efficiency of the overall system. Since we believe that what matters most is the user's feeling that the translation proposal

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

Table 1. *Human evaluation (sentences)*

	(1%) 30 clusters	(2%) 60 clusters	(3%) 90 clusters
A	30 (15%)	32 (16%)	32 (16%)
B	58 (29%)	64 (32%)	66 (33%)
C	52 (26%)	46 (23%)	48 (24%)
D	60 (30%)	58 (29%)	54 (27%)
Total	200 sent.	200 sent.	200 sent.

Table 2. *Human evaluation (sentences or sentence segments)*

	(1%) 30 clusters	(2%) 60 clusters	(3%) 90 clusters
A	49 (23%)	57 (25%)	57 (25%)
B	77 (36%)	89 (39%)	92 (40%)
C	41 (19%)	39 (17%)	41 (18%)
D	47 (22%)	43 (19%)	39 (17%)
Total	214 sent. or seg.	228 sent or seg	229 sent or seg

is helpful, we asked one professional translator to assign each translation proposal of the system to one of four categories:

A: The proposal is the correct (or almost) translation.
B: The proposal is very helpful in order to produce the translation.
C: The proposal can help in order to produce the translation.
D: The proposal is of no use to the translator.

The evaluation results are shown in Tables 1 and 2 for the case of sentences and sentence segments, respectively.

From Table 1 it can be seen that in the case of 30 clusters, the system will be very useful (categories A and B) to the translator for 44% of the input sentences, and some information (categories A, B and C) can at least be obtained from 70% of the retrievals. In the case of 60 and 90 clusters, the results do not change significantly, even though they are better. The results shown in Table 2 do have the same tendency as far as the number of clusters is concerned, but are rather better due to the exploitation of sentence segmentation.

The actual evaluation results seen independently of the number of clusters and the sentence-subsentence differentiation, indicate that the idea of providing a translator with similar translation examples has great potential (it should, however, be mentioned, that the CELEX database is quite suitable for Translation Memory applications due to its great degree of repetitiveness, but on the other hand, the number of stored sentences can be considered rather small for a Translation Memory application). To check whether the produced similarity score reflects the user's judgement of the actual similarity, we present results in Table 3 for the case of

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

Table 3. *Evaluation of the similarity metric*

	Average score	Standard deviation
category A sentences	92%	2·8%
category B sentences	74%	7·1%
category C sentences	49%	15·8%
category D sentences	14%	20·2%

Table 4. *Response time gain*

	Decrease in response time
30 clusters	85%
60 clusters	90%
90 clusters	86%

90 clusters working with whole sentences. The column 'average score' indicates the average similarity score (the similarity score of the input sentence with its best matching sentence contained in the translation archive) of the sentences of each category, while the column 'standard deviation' indicates the deviation of the scores from the mean value (recall that the sentences were allocated to categories by the human translator).

Hence, it seems that the proposed combination of surface structure and content is compatible with the human perception of text similarity, even though standard deviations are quite high for all but category A.

The role of clustering in making the retrieval of the best matching example from the database more efficient can be investigated along two axes. The *decrease in the system response time* and the *deterioration in the system accuracy*.

As far as the response time is concerned, this is dramatically decreased with the use of clustering compared to the alternative of searching exhaustively through the translation archive. In Table 4 the average percentage of decrease of the response time is shown for the three different implementations of the system working with whole sentences (results on different number of clusters corresponding to the segment version of the system would not be compatible due to the potentially different number of stored sentences and segments).

As shown in Table 4, the response time was at its lowest value in the case of 60 clusters. This result is in agreement with the theoretically expected optimum number of clusters (as far as the response time is concerned). If we denote by N the number of stored sentences and by x the number of clusters of the system, then the system, in order to respond to an input, should make

$f(x) = x$ comparisons to the cluster centres $+N/x$ comparisons within the favourite cluster.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

Table 5. *System accuracy (sentences)*

	MISSED	MISSED BY
30 clusters	11%	10·1%
60 clusters	9%	8·1%
90 clusters	8%	7·9%

This description presupposes that the database sentences are evenly allocated to the x clusters, which is of course not true but still the closest we can get to the truth. To determine the optimum value of x, we set the first derivative of the response time function to zero, i.e.

$$\frac{d}{dx}f(x) = 0$$

which results in $x = \sqrt{N}$.

By substituting 3000 for N, we get $x \approx 55$.

As far as the actual results shown in Table 4 are concerned, the theoretically expected value of the decrease in response time can be estimated if again we make the assumption that the database sentences are evenly distributed to the x clusters. If we set x to its optimum value, that is $\sqrt{N}$, the number of comparisons in the case of clustering would be $\sqrt{N} + N/\sqrt{N}$ compared to the alternative of N comparisons. Hence, we have a

$$\left(1 - \frac{2}{\sqrt{N}}\right) \cdot 100\%$$

decrease in the system response time. For $N = 3000$ we get a 96% decrease. The deviation from the experimental value is small (experimental value 90%, see Table 4) and can be attributed to the violation of our assumption about the sentence distribution in the clusters. This violation is mainly due to the existence of a rather big cluster containing the sentences without any functional words (quite many in the CELEX database).

Turning to the second investigation axis for clustering, that is system accuracy, it would be interesting to see how many best matches are missed because they are not placed in the cluster with the best matching centre. Results are shown in Table 5 for the case of whole sentences (for compatibility with Table 4). In Table 5, the column 'MISSED' indicates the percentage of the input sentences for which the best match in the corpus was not located in the single favourite cluster, while the column 'MISSED BY' indicates the average deviation of the located best matches from the actual best matches in the translation archive for these cases.

It can be seen from Table 5 that there is a tendency (even though a small one) for the accuracy of the system to improve with the number of clusters. Furthermore, this tendency is also reflected in the translator's judgement, as shown in Table 1. It can

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

Table 6. *System accuracy (sentences or sentence segments)*

	MISSED	MISSED BY
30 clusters	5%	7·2%
60 clusters	3%	6·1%
90 clusters	3%	6·1%

be attributed to 'hidden' parts of sentences allocated to clusters (parts that are not represented by the cluster centres). The more clusters, the better the sentences of the translation archive are represented by the cluster centres. Hence, the more clusters and the smaller the sentences contained in the database, the better the accuracy of the overall system.

Therefore, a compromise should be made between response time and accuracy. We believe that this compromise should be in favour of the response time rather than accuracy, since not only is the improvement in accuracy rather small as we employ more clusters, but also the inaccuracy itself is low regardless of the number of clusters (column 'MISSED BY' in Table 5). On the other hand, the response time is a very crucial factor in an operational system.

What is also interesting to see is the effect of the iterative clustering procedure depicted in Figure 5 on the system accuracy. As shown in Table 6 (which contains results for the system working with segments), by applying the segmentation phase and then reclustering the corpus the system accuracy improved significantly. This is due to the hidden parts of sentences which emerged to the 'surface' by the use of segmentation. The combination of this result with the much better user satisfaction when the system utilises sentence segments (as shown in Tables 1 and 2) fully justifies the use of segmentation and reclustering. However, one should always bear in mind that efficient segmentation presupposes automatic subsentential alignment.

However, the efficacy of the use of clustering is not without cost. The time required for clustering of a big corpus can be prohibitive, even though clustering needs to be executed only once. A way round this problem, is to cluster a small part of the corpus and then allocate each remaining sentence to its best matching centre. The same approach can be employed to update the database of the EBMT system at the operation phase, so that the database becomes increasingly user customised. Of course, with this approach clustering is not optimum. To test if it is still efficient, we clustered 600 sentences of our corpus (which accounts for 20% of the corpus), and then updated the database with the remaining 2400 sentences. The evaluation and accuracy results for the system working with whole sentences are shown in Tables 7 and 8, respectively.

By comparing Tables 5 and 8, we notice a significant deterioration in accuracy. This is, however, not reflected in the evaluation results in Table 7, where it can be seen that the number of sentences falling to categories A and B (which are the crucial categories for the system) has not changed compared to the case of optimum clustering (Table 1). The suboptimum clustering only affected categories C and D.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

Table 7. *Human evaluation (suboptimum clustering)*

	(1%) 30 clusters	(2%) 60 clusters	(3%) 90 clusters
A	30 (15%)	32 (16%)	32 (16%)
B	56 (28%)	64 (32%)	66 (33%)
C	34 (17%)	32 (16%)	32 (16%)
D	80 (40%)	72 (36%)	70 (35%)
Total	200 sent.	200 sent.	200 sent.

Table 8. *System accuracy (suboptimum clustering)*

	MISSED	MISSED BY
30 clusters	16%	12·5%
60 clusters	12%	9·9%
90 clusters	12%	10·2%

This result is very reasonable: a sentence (s1) of the test suite which is very similar to a sentence (s2) in the database (and therefore about to fall in category A or B) will probably be allocated to the same cluster to which (s2) has been allocated (and therefore (s2) will be located in the favourite cluster for (s1)). This, however, does not necessarily hold if (s1), and its best matching sentence (s2) in the database exhibits little similarity (and therefore (s1) about to fall in category C or D). Hence, the deterioration in accuracy depicted in Table 8 stems from categories C and D, which are of little concern to the user.

The efficacy of the updating method (leading to suboptimum clustering) might not be so high if the updating sentences do not belong to the same sublanguage as the sentences having been used for clustering. This is, however, not a practical problem, since in EBMT a system operating with different sublanguages usually utilises a different database for each sublanguage.

4.1 Examples

Two examples demostrating the system's functionality are next presented. They correspond to the case of the system utilising sentence segments. Since the matching mechanism has so far been implemented in the Greek language, these sentences, which are presented in English so that they are widely understandable, are in fact the translation of the corresponding matched sentences in Greek.The segment of the input sentence appearing in square brackets is the 'active' segment, i.e. the segment for which a translation proposal is being made. Bold characters correspond to words of the input and reference sentence that have been matched. The level of matching can be: identical FW, FW belonging to the same group, identical lemma of non FW and same pos tag of non FW.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

EXAMPLE 1
INPUT SENTENCE:[**HAVING REGARD TO COUNCIL REGULATION (EEC) NO 1035/72 OF 18 MAY 1972 ON THE COMMON ORGANIZATION OF THE MARKET IN FRUIT AND VEGETABLES (1), AS LAST AMENDED BY REGULATION (EEC) NO 1119/89 (2)**], AND IN PARTICULAR THE LAST SUBPAPAGRAPH OF ARTICLE 18 (1) THEREOF.

REF. SENTENCE: **HAVING REGARD TO COUNCIL REGULATION (EEC) NO 426/86 OF 24 FEBRUARY 1986 ON THE COMMON ORGANIZATION OF THE MARKET IN** PRODUCTS PROCESSED FROM **FRUIT AND VEGETABLES (16), AS LAST AMENDED BY REGULATION (EEC) NO 2247/88 (17)**

SIMILARITY SCORE: **93%**

INPUT SENTENCE: HAVING REGARD TO COUNCIL REGULATION (EEC) NO 1035/72 OF 18 MAY 1972 ON THE COMMON ORGANIZATION OF THE MARKET IN FRUIT AND VEGETABLES (1), AS LAST AMENDED BY REGULATION (EEC) NO 1119/89 (2), [**AND IN PARTICULAR** THE LAST SUBPAPAGRAPH OF **ARTICLE 18 (1) THEREOF**]

REF. SENTENCE: **AND IN PARTICULAR ARTICLE 2 (1) THEREOF**.

SIMILARITY SCORE: **57%**

EXAMPLE 2

INPUT SENTENCE: [**WHEREAS REGULATIONS (EEC) NO 1408/71 (4)**] AND (EEC) NO 574/72 (5), AS UPDATED BY REGULATION (EEC) NO 2001/83 (6) AND AS LAST AMENDED BY REGULATION (EEC) NO 3811/86 (7), ARE SUBJECT TO THE TECHNICAL ADAPTATIONS SET OUT IN CHAPTER VII.

REF. SENTENCE: **WHEREAS** COMMISSION **REGULATION (EEC) NO 3627/88 (14)** SHOULD BE REPEALED

SIMILARITY SCORE: **55%**

INPUT SENTENCE: WHEREAS REGULATIONS (EEC) NO 1408/71 (4) AND (EEC) NO 574/72 (5), [**AS UPDATED BY REGULATION (EEC) NO 2001/83 (6) AND**] AS LAST AMENDED BY REGULATION (EEC) NO 3811/86 (7), ARE SUBJECT TO THE TECHNICAL ADAPTATIONS SET OUT IN CHAPTER VII.

REF. SENTENCE: **AS** LAST **AMENDED BY REGULATION (EEC) NO 3773/87**

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

(2), AND IN PARTICULAR ARTUICLE 1 THEREOF.

SIMILARITY SCORE: **63%**

INPUT SENTENCE: WHEREAS REGULATIONS (EEC) NO 1408/71 (4) AND (EEC) NO 574/72 (5), AS UPDATED BY REGULATION (EEC) NO 2001/83 (6) AND [AS LAST AMENDED BY REGULATION (EEC) NO 3811/86 (7), ARE SUBJECT TO **THE** TECHNICAL **ADAPTATIONS SET OUT IN CHAPTER VII**]

REF. SENTENCE: IN SO FAR AS **THE CONDITIONS SET OUT IN PARAGRAPH 3** ARE NOT FULFILLED, THE SECURITIES SHALL BE FORFEIT.

SIMILARITY SCORE: **32%**

Notice that since the normalisation of the score is based on the lengths of the sentences under comparison, the scores indicated above would be much lower (even for the cases of very good matches) if the system did not segment the input sentence (because then normalisation would have to take into account the length of the input sentence as well, instead of the length of its segments).

These examples also demonstrate how the similarity metric allows additions or deletions of FWs or not, to produce the similarity score as well as the parts of the two sentences that contributed to this score.

5 Future work

We intend to run the experiment on a much larger part of the CELEX database as well as on other bilingual corpora, so as to make more concrete remarks about the efficiency of the proposed method. We believe that a larger corpus will lead to even better results than those presented here. On the other hand, the set of FWs used and their grouping can also be further improved by our experience of using the system. Finally, we intend to run the experiment on the English part of the CELEX database. For this, the set of English FWs and its grouping should be determined.

6 Appendix

The FWs of the two sentences A and B are contained in the arrays A (1:lenA) and B (1:lenB) corresponding to the two axes of the xy-plane. Let D[i][j] be the maximum similarity score between the sentence segment starting at the beginning of A till the ith FW of A and the sentence segment starting at the beginning of B till the jth FW of B. indexA[i][j] and indexB[i][j] are used to keep information about the previous node run by the path currently visiting the node (i,j) (see Figure 2). Whenever a diagonal transition is investigated, the system calls a second level DP-algorithm which produces a local additional score due to the potential similarity of lemmas and tags of the words lying between the corresponding FWs. Thus, **S(i,j)** denotes

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

the contribution to the similarity score due to the words appearing between FWs i and i+1 in one sentence and between j and j+1 in the other. S(i,j) is calculated using exactly the same DP-algorithm as the one described next (allowing additions, deletions,…) provided that we use (**L**), (**T**) and (**PT**) (a penalty score attributed to a mismatch at the tag-level) in place of (**I**), (**G**) and (**P**) respectively.

The Algorithm

for i=0:lenA D[0][i]=0;
for i=0:lenB D[i][0]=0;

maximum=0;
for i=1:lenA
  for j=1:lenB

$$D[i][j] = max \begin{pmatrix} D[i-1][j] + P \\ D[i][j-1] + P \\ D[i-1][j-1] + C(A(i), B(j)) \end{pmatrix}$$

$$indexA[i][j] = arg_i max \begin{pmatrix} D[i-1][j] + P \\ D[i][j-1] + P \\ D[i-1][j-1] + C(A(i), B(j)) \end{pmatrix}$$

$$indexB[i][j] = arg_j max \begin{pmatrix} D[i-1][j] + P \\ D[i][j-1] + P \\ D[i-1][j-1] + C(A(i), B(j)) \end{pmatrix}$$

  if D[i][j]>maximum
    maximum=D[i][j];
    il=i;
    jl=j;

maximum=maximum+S(il, jl);

where,

$$C(A(i), B(j)) = \begin{pmatrix} I + S(i-1, j-1)\ ,\ if\ A(i)\ and\ B(j)\ are\ identical \\ G + S(i-1, j-1)\ ,\ if\ A(i)\ and\ B(j)\ belong\ to\ the\ same\ group \\ -D[i-1][j-1]\ ,\ otherwise \end{pmatrix}$$

Backtracking

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

```
i=il;
j=jl;
repeat
   a=indexA[i][j];
   b=indexB[i][j];
   i=a;
   j=b;
until ( D[i][j]=0 )
```

and the resulting similarity score (**maximum**) corresponds to the segment starting from the i+1 FW and ending to the il FW in one sentence and the segment starting from the j+1 FW and ending to the jl FW in the other sentence.

To clarify the situation, an example of the D, indexA, indexB matrices at the end of the application of the DP algorithm is given, corresponding to the example of Figure 2.

$$D = \begin{bmatrix} I & 0 & 0 \\ I+P & I+P+P & I+P+P+P \\ I+P+P & I+P+P+P & I+P+P+I \end{bmatrix}$$

$$indexA = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 2 \\ 2 & 3 & 2 \end{bmatrix}$$

$$indexB = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{bmatrix}.$$

References

Brown, P. F. *et al.* (1991) Aligning sentences in parallel corpora. *Proc. ACL 91* 169–176.

Brown, P. F. *et al.* (1993) The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, June: 263–311.

Cranias, L., Papageorgiou, H. and Piperidis, S. (1994) A matching technique in EBMT. *Proc. COLING 94* Kyoto, 100–104.

Dagan, I., Church, K. and Gale, W. (1993) Robust bilingual word alignment for machine aided translation. *Proc. Workshop on Very Large Corpora 93* Kyoto, 1–8.

Furuse, O. and Iida, H. (1992) Cooperation between Ttransfer and analysis in example- based framework. *Proc. COLING 92* Nandes, 645–651.

Gale, W. A. and Church, K. W. (1991) A program for aligning sentences in bilingual corpora. *Proc. ACL 91*, California, 177–184.

Gale, W. A. and Church, K. W. (1991) Identifying word correspondence in parallel text". *Proc. DARPA Workshop on Speech and Natural Language.*

Kaji, H., Kida, Y. and Morimoto, Y. (1992) Learning translation templates from bilingual text. *Proc. COLING 92* Nandes, 672–678.

Kay, M. and Roescheisen, ? (1993) Text translation alignment. *Computational Linguistics* **19**: 121–143.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press

Kumano, A. and Hirakawa, H. (1993) Building an MT dictionary from parallel texts based on linguistic and statistical information. *Proc. ACL 93* Ohio, 76–81.

Nagao, M. (1984) A framework of a mechanical translation between Japanese and English by analogy principle. *Artificial and Human Intelligence* 173–180.

Ney, H. (1984) The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Trans. on ASSP* **32**.

Nirenburg, S. *et al*. (1993) Two approaches to matching in example-based machine translation. *Proc. TMI 93* Kyoto.

Papageorgiou, H., Cranias, L. and Piperidis, S. (1994) Automatic alignment in parallel corpora. *Proc. ACL 94* New Mexico.

Sadler, V. and Vendelmans, R. (1990) Pilot implementation of a bilingual knowledge bank. *Proc. COLING 90* Helsinki, 449–451.

Sakoe, H. and Chiba, S. (1978) Dynamic programming algorithm optimisation for spoken word recognition. *IEEE Trans. ASSP* **26**.

Sato, S. and Nagao, M. (1990) Toward memory-based translation. *Proc. COLING 90* Helsinki, 247–252.

Sato, S. (1992) CTM: an example-based translation aid system. *Proc. COLING 92* Nandes, 1259–1263.

Simard, M., Foster, G. and Isabelle, P. (1992) Using cognates to align sentences in bilingual corpora. *Proc. TMI 92* Montreal, 67–81.

Sumita, E. and Tsutsumi, Y. (1988) A translation aid system using flexible text retrieval based on syntax-matching. *TRL Research Report*, Tokyo Research Laboratory, IBM.

Sumita, E. and Iida, H. (1991) Experiments and prospects of example-based machine translation. *Proc. ACL 91* California, 185–192.

Wilpon, J. and Rabiner, L. (1985) A modified k-means clustering algorithm for use in isolated word recognition. *IEEE Trans. ASSP* **33**: 587–594.

https://doi.org/10.1017/S1351324997001447 Published online by Cambridge University Press