Natural Language Engineering **19** (1): 33–60. © Cambridge University Press 2011
doi:10.1017/S135132491100026X

A unified alignment algorithm for bilingual data

CHRISTOPH TILLMANN[1] and
SANJIKA HEWAVITHARANA[2*]
[1]*IBM T.J. Watson Research Center, Yorktown Heights, New York, NY 10598, USA*
email: ctill@us.ibm.com
[2]*Carnegie Mellon University, Pittsburgh, PA 15213, USA*
email: sanjika@cs.cmu.edu

(*Received 17 December 2010; revised 5 July 2011; accepted 18 July 2011; first published online 13 September 2011*)

Abstract

The paper presents a novel unified algorithm for aligning sentences with their translations in bilingual data. With the help of ideas from a stack-based dynamic programming decoder for speech recognition (Ney 1984), the search is parametrized in a novel way such that the unified algorithm can be used on various types of data that have been previously handled by separate implementations: the extracted text chunk pairs can be either sub-sentential pairs, one-to-one, or many-to-many sentence-level pairs. The one-stage search algorithm is carried out in a single run over the data. Its memory requirements are independent of the length of the source document, and it is applicable to sentence-level parallel as well as comparable data. With the help of a unified beam-search candidate pruning, the algorithm is very efficient: it avoids any document-level pre-filtering and uses less restrictive sentence-level filtering. Results are presented on a Russian–English, a Spanish–English, and an Arabic–English extraction task. Based on simple word-based scoring features, text chunk pairs are extracted out of several trillion candidates, where the search is carried out on 300 processors in parallel.

1 Introduction

This paper presents a unified algorithm for extracting translation correspondences from non-sentence-aligned bilingual data in the area of statistical machine translation (SMT). Originally, such algorithms have been proposed for parallel corpora, which can be aligned largely monotonically with no re-orderings and only limited insertions and deletions between the two sentence collections (Brown *et al.* 1991; Gale and Church 1991; Melamed 1999; Moore 2002; Zhao and Vogal 2002; Utiyama and Isahara 2003; Deng, Kumar and Byrne 2006; Ma 2006). More recent work has focused on extracting parallel data from comparable noisy data. Here, the data might be processed at the document level (Snover, Dorr and Schwartz 2008), or the document-level processing is used as a pre-filtering step (Utiyama and Isahara 2003;

[*]Part of this research was conducted during the second author's internship at IBM T.J. Watson Research Center, Yorktown Heights, New York.

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

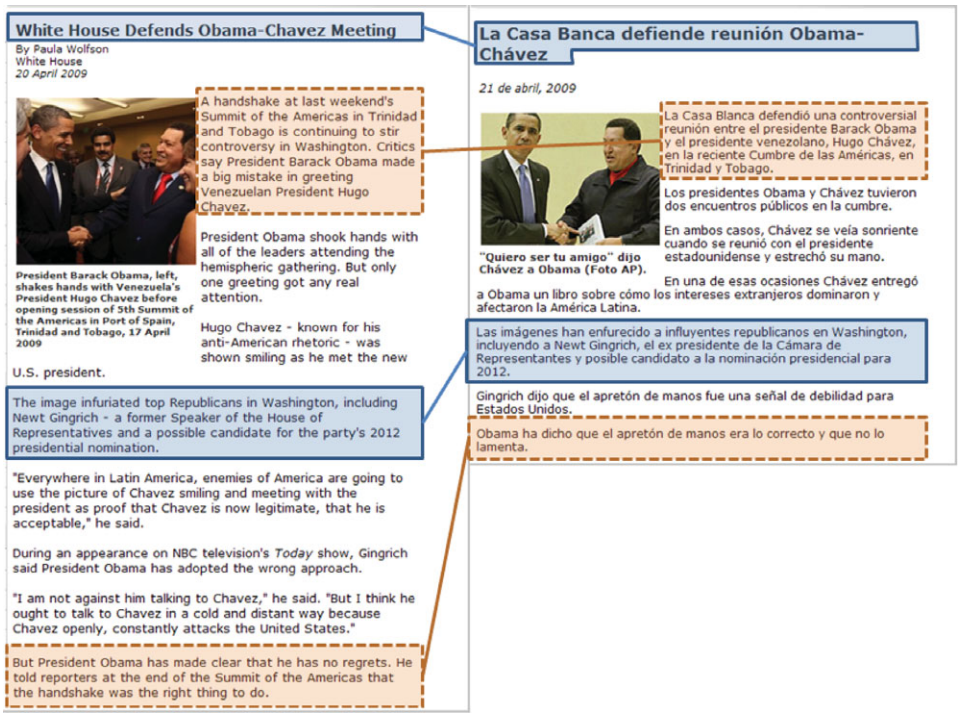


Fig. 1. (Colour online) Illustration of different levels of parallelism present in an English (left) and Spanish (right) comparable document pair. The sentences marked with solid lines are parallel sentences. The sentences marked with broken lines are not parallel, but contain sub-sentential fragments that are parallel. Both documents are from the Voice of America News (VOA News).

Munteanu and Marcu 2005; Quirk, Udupa and Menezes 2007). In Munteanu and Marcu (2005) and Quirk *et al.* (2007), one-to-one sentence-level pairs are then derived from the document-aligned data. This sentence alignment can be further used to extract sub-sentential fragment pairs (Munteanu and Marcu 2006; Quirk *et al.* 2007). Figure 1 highlights these different levels of parallelism present in a typical comparable document pair: sentence level as well as sub-sentential translation correspondences. In the current paper, we present a single unified search implementation that can be used on all the different extraction tasks handled by various above-cited algorithms. As the degree of parallelism (e.g. parallel data, noisy parallel, or comparable data as defined by Fung and Cheung 2004) for given bilingual data might be unknown, the unified algorithm could be applied more flexibly.

In this paper, translation pair extraction is handled as a chunk-alignment problem: No document-level pre-filtering is used and sentence-level pre-filtering is handled as a novel beam-search pruning step. For comparable data, the extraction of many-to-many translation pairs is included in a straight-forward way. Sub-sentential fragments are extracted with a smaller amount of pre-filtering than previous work. The unified algorithm uses techniques from large-scale dynamic programming (DP) beam-search algorithms in speech recognition (SR) (Ney 1984): The search space is

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

dynamically constructed, and just two stacks of active hypotheses or states are used to process the parallel data at word level. This way the algorithm is also similar to SMT decoding algorithms (Koehn 2004; Och and Ney 2004; Tillmann 2006). In order to carry out the sentence alignment, the algorithm uses simple word-based features: lexical scoring features based on the so-called IBM Model-1 (Brown *et al.* 1993), coverage features, and similarity features based on character matching. These can be computed incrementally to support the beam-search extraction algorithm.

The main contribution of the current paper lies in a combination of techniques from previous work on extracting parallel data, stack-based decoding for SR, alignment algorithms for SMT, and the use of incrementally computed features for phrase-based SMT decoders. As a result, the classifier-based extraction algorithm can explore huge search spaces at an unprecedented scale. The paper is organized as follows: Section 2 presents the extraction task for various data conditions as a so-called link alignment problem, which is classified into different alignment types and link-type sets. The extraction task is formally compared to a pattern recognition formulation of the SR problem. Section 3 describes the feature-based link scoring model used in this paper. Section 4 demonstrates the unified stack-based search, which is the main contribution of this paper. This section presents detailed pseudo-code of the unified search, which depends on a careful stack-based implementation to maintain efficiency while being applicable to different alignment types. Section 5 presents details about the beam-search pruning as well as the efficient incremental feature computation. In Section 6, we show experimental results. Section 7 compares the unified algorithm with previous work in the literature and discusses future work.

2 Link sequence search problem

We extract bilingual pairs of text chunks: A chunk consists of a sequence of source or target words (Deng *et al.* 2006). Chunk pairs are either sub-sentential fragment pairs, one-to-one, or many-to-many sentence-level pairs. We call an interval pair of source and target positions that corresponds to a chunk pair in a bitext a *link*. A bitext is a parallel text (Melamed 1999), which is indexed at the word level in the current paper. Even though we are interested in the words covered by a link, conceptually our algorithm computes a chunk pair alignment that constitutes a sequence of links. Based on a link, the words can always be recovered after the search has finished. For the experiments in Section 6, we consider 1-to-1, 1-to-4, and 4-to-4-type links at the sentence level as well as 10-to-10-type links at the word level when extracting sub-sentential fragment pairs. Here the numbers refer to the maximum link size in terms of sentences or words in the source or target side. The link concept is illustrated in Figure 2. For the 1-4-type links, a single sentence is aligned to up to four sentences in the other language (this restriction is used by Ma 2006), i.e. this link set $\mathcal{L}$ contains nine types: $\mathcal{L} = \{1\text{-}0, 1\text{-}1, 1\text{-}2, 1\text{-}3, 1\text{-}4, 0\text{-}1, 2\text{-}1, 3\text{-}1, 4\text{-}1\}$. Here special 'null' links for insertions and deletions are included. Similarly, for the 4-to-4-type links up to four source and target sentences are aligned to each other. Here we exclude link types with the same length, i.e. 2-to-2, 3-to-3, and 4-to-4 links, as these can be handled by sequences of 1-to-1 links. Similarly, sub-sentential links consist of up to 10 source

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

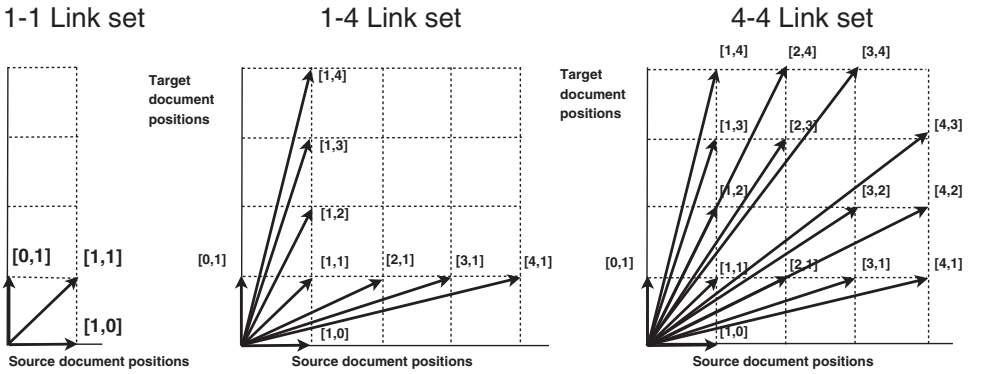


Fig. 2. Illustration of different link sets $\mathcal{L}$ that are used in this paper. A link corresponds to a pair of text chunks that cover a certain number of sentences or words. During search, link sequences are extracted based on alignment coverage/continuity restrictions (see Figure 3 for an illustration). A link corresponds to a local transition between bitext (sentence or word) positions. A pattern recognition definition for link sequences is given in Section 2.1. 10-to-10-word level transitions for extracting sub-sentential fragments are not shown here.

and target words. Link types can be handled flexibly at run time: they correspond to state expansions in the dynamically constructed search space.

The search algorithm handles different degrees of sentence-level order differences between source and target texts. Different alignment types are handled jointly within a unified implementation. The alignment types are illustrated in Figure 3.

Type I. Monotone search. In Brown *et al.* (1991), Gale and Church (1991), Chen (1993), Moore (2002), Zhao and Vogel (2002), and Ma (2006), the sentence alignment search is restricted to monotone alignments, which results in an efficient search algorithm. Our current constant memory algorithm carries out a full monotone search without pruning. The memory requirements are constant with respect to the source document length, but linear with respect to the target document length. The monotone search can be carried out at the sentence as well as word levels. The word-level search is used to compute a sub-sentential fragment alignment.

Type II. Sentence-level re-ordering search. A novel restricted sentence level re-ordering search is introduced. It uses a target coverage vector to handle some sentence-order differences in mostly monotone data. The search space is restricted by a local re-ordering window similar to the way a decoder for SMT handles word-level re-ordering. It can also be related to the binary search in Deng *et al.* (2006).

Type III. Comparable data search. When searching for sentence-level pairs in comparable data, no monotonicity assumption is made for source and target sentences. Typically, only one-to-one link pairs are extracted (Munteanu and Marcu 2005; Quirk *et al.* 2007; Tillmann and Xu 2009). The current algorithm is capable of extracting many-to-many links as a straight-forward extension of the one-to-one link search.

If the source and target sentences are from a matching document pair, a monotone search or almost monotone search can be carried out (Type I and Type II defined

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

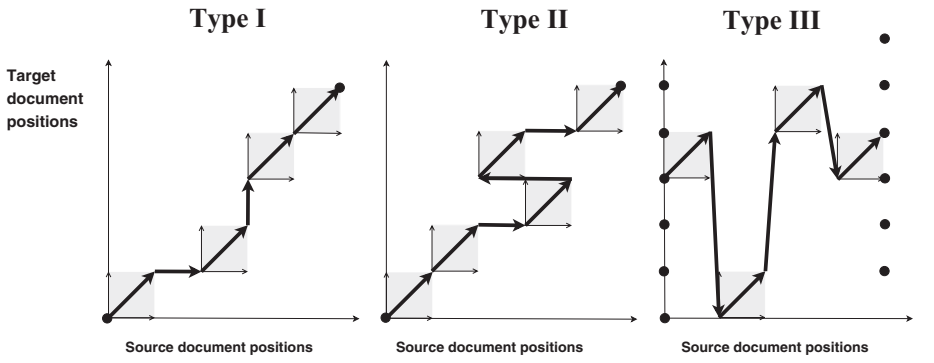


Fig. 3. The unified algorithm handles three alignment types. Only one-to-one links are shown, but many-to-many links (see Figure 2) are handled uniformly across all alignment types. The search algorithm finds the shortest path through the bitext space. The path along with start and end point restrictions is shown as bold lines and points. For details, see Section 2.1.

above). If a document-level alignment is unavailable, e.g. for comparable data, a non-monotone Type III search is carried out. Here the set of target chunks considered as matches is restricted based on a publication date window (Munteanu and Marcu 2005): for a given source sentence S only those target sentences T are considered that have been published within a window of ± 3 days. The link search for sub-sentential fragment pairs is handled as a combination of the search for alignment Types I and III: We pre-match candidates at the sentence-level and carry out a monotone search within the candidate sentences. In comparison, Munteanu and Marcu (2006) and Quirk *et al.* (2007) extract fragments based on a two-step filtering pipeline: (1) matching document pairs are extracted based on an Information-Retrieval (IR) criterion; (2) from within the matching documents, sentence pairs are extracted based on a sentence-level matching criteria. Finally, sub-sentential fragments are extracted from these sentence pairs. In this paper, we all together skip the document-level pre-filtering, and formulate the sentence-level pre-filtering step as a special pruning step within the unified beam-search algorithm: even sub-sentential fragment extraction is carried out in a single run over the data.

2.1 Link extraction as a pattern recognition problem

In this section we define the link alignment problem in more technical detail. We present a geometrical pattern recognition definition of the chunk alignment problem that paves the way for the formal description of the unified algorithm in Section 4. We are trying to align a collection of source sentences $\{s_m\}$ and target sentences $\{t_n\}$, where $0 \leq m \leq \mathcal{S}$ and $0 \leq n \leq \mathcal{T}$. $\mathcal{S}$ and $\mathcal{T}$ are the number of source and target sentences. We are searching for the so-called *link* alignment sequences l_1^N that consists of N links (N is unknown before the search). A link sequence l_1^N is a list of interval pairs that jointly cover the unaligned bitext based on certain alignment coverage and continuity restrictions that depend on the search types. A single link is a interval pair that covers a square in the bitext space:

$$l = \left[\left[j_{beg}(l) \, , \; j_{end}(l) \right] , \left[i_{beg}(l) \, , \; i_{end}(l) \right] \right] \tag{1}$$

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

A link starts at bitext point $[j_{beg}, i_{beg}]$ and ends at bitext point $[j_{end}, i_{end}]$, where source and target positions are restricted as follows: $j_{beg}, j_{end} \in \{1, \ldots, \mathcal{J}\}$ and $i_{beg}, i_{end} \in \{1, \ldots, \mathcal{I}\}$. Here, $\mathcal{I}$ and $\mathcal{J}$ are the number of words in the target document and source document, respectively. The source chunk words that cover the source interval $[j_{beg}(l), j_{end}(l)]$ are denoted by $S(l)$. The target chunk words $T(l)$ are defined accordingly. Based on the link sequence l_1^N, we consider the list of N text chunk pairs $[S(l_i), T(l_i)]_{i=1}^{i=N}$ from the unaligned data for extraction.

We now describe different coverage and continuity restrictions for link sequences that are handled jointly by the unified search algorithm. By default, each source text position j is covered by exactly one source link interval $[j_{beg}(l), j_{end}(l)]$. Link source interval sequences are restricted with respect to their begin and end positions: $j_{beg}(l) = j_{end}(l-1) + 1$ as well as $j_{beg}(l_0) = 0$ and $j_{end}(l_N) = \mathcal{J}$. The target coverage varies for different search types. For the Type I search, the link sequence coverage is also monotone over the target text, and link target sequences are restricted as follows: $i_{beg}(l) = i_{end}(l-1) + 1$ as well as $i_{beg}(l_0) = 0$ and $i_{end}(l_N) = \mathcal{I}$. For the Type II alignments, each target position is covered by at most a single link, and some local chunk re-ordering is allowed. For Type III alignments, only the default source coverage is used. Here, we also allow for overlapping link sequences during the search: A link ends at each source sentence position s and the same source word position j may be covered by several links. Unaligned source and target chunks are handled based on source and target null-coverage links as well. Details depend on different search types and are explained along with the unified search algorithm. In order to process the data at the word level, we maintain the following mapping index $u_S(m)$:

$$m \rightarrow (\text{start position of } m\text{-th sentence}, \text{end position of } m\text{-th sentence}) \tag{2}$$

i.e. for each source sentence s_m we compute its word-level start and end position (a target index $u_T(n)$ is defined accordingly). For a parallel sentence collection, these positions are computed up-front before the search. In order to carry out a sentence-level search with many-to-many sentence-level links, we look up the bitext word start positions of its first source and target sentence, as well as the bitext word end position of its final source and target sentences. The sentence-level link is then processed as a word-level link, which may cover several source and target sentences.

The unified search algorithm to be defined in Section 4 is a shortest path finding algorithm. Each link sequence corresponds to a path through the bitext space, where a single link is re-presented by its link diagonal. The shortest path is restricted based on the search type restrictions described above. For a single link, the matching cost is computed as the matching score between the pair of text chunks that are covered by that link. Unlike work on word alignment algorithms, transitions between separate links are assigned zero cost, and the cost of a link sequence is defined as

$$c\left(l_1^N\right) = \sum_{i=1}^{N} \varrho(S(l_i), T(l_i)) \tag{3}$$

where $\varrho(S, T)$ is the cost for matching source chunk $S(l)$ with target chunk $T(l)$ as defined in the following section. We search for the lowest scoring link sequence $\hat{l}_1^N$

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

such that

$$\hat{l}_1^N = \arg\min_{l_1^N} c(l_1^N) \tag{4}$$

The link alignment search is handled as a *pattern recognition* problem. The source document is the *input pattern* that is assumed to be decomposed into chunks. Those source chunks are aligned to *reference patterns* that are target word chunks. In particular, the search for Type III alignments can be directly compared to a pattern recognition formulation of the SR search problem (Ney 1984). The source chunks correspond to words in the acoustic input signal. Each document position within a chunk corresponds to an acoustic time frame in the SR search. Each time frame in the SR search is represented by an acoustic vector that is matched against the reference patterns. Similarly, each source chunk word position together with its target chunk match is re-presented by a partial feature vector, which is used in the Maximum Entropy (ME) classifier in Section 3. A sum of these local feature vectors is used to compute a matching score for the source chunk against a target chunk with which it is aligned. As the chunk alignment problem considers links of different granularity, it is a more general pattern search algorithm than the SR search in Ney (1984) that considers only one-to-one word-level mappings.

3 Link scoring model

In this section, we show how the link scoring function $\varrho(S, T)$ for a parallel link pair (S, T) in (3) is computed based on a Maximum Entropy (ME) classifier (Munteanu and Marcu 2005). We compute an ME-based score for each link pair *incrementally* at the word level. To facilitate the incremental position-by-position computation, we use a word-based scoring function, which is based on the so-called IBM Model-1 (Brown *et al.* 1993). Similar approaches that rely on word-to-word translation models have been presented by Chen (1993), Zhao and Vogel (2002), Resnik and Smith (2003), and Ma (2006). The Model-1 is trained on some parallel data available for a language pair, i.e. the data used to train the baseline systems in Section 6. The Model-1 scores itself are used as features for the ME classifier along with several coverage-based and fertility features derived from the Model-1 alignment. The Model-1-based scoring function is based on two terms: a target-to-source score, $\sigma_{T\to S}$, and a source-to-target score, $\sigma_{S\to T}$:

$$\sigma_{T\to S}(S, T) = \sum_{j=1}^{J} \underbrace{\frac{1}{J} \cdot \log\left(\frac{1}{I} \cdot \sum_{i=1}^{I} p(s_j|t_i)\right)}_{\sigma(s_j,T)} \tag{5}$$

$$\sigma_{S\to T}(S, T) = \sum_{i=1}^{I} \underbrace{\frac{1}{I} \cdot \log\left(\frac{1}{J} \cdot \sum_{j=1}^{J} p(t_i|s_j)\right)}_{\tau(t_i,S)} \tag{6}$$

where $S = s_1^J$ is a source chunk of length J, and $T = t_1^I$ is a target chunk of length I. Here the single word translation probabilities $p(\cdot, \cdot)$ are smoothed to avoid zero

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

probability entries. The lexical scoring features are normalized with respect to the chunk lengths J and I. For the beam-search pruning in Section 5.1, we make use of a simplified scoring function $\varrho'(S,T)$ where just these two features are used: $\varrho'(S,T) = \sigma_{T\to S}(S,T) + \sigma_{S\to T}(S,T)$ and candidates are directly pre-filtered at the sentence level. In Tillmann and Xu (2009), the use of $\varrho'(S,T)$ together with a single selection threshold ρ proved to be very efficient in selecting sentence pairs from comparable data. The classifier probability is defined as follows:

$$p(c|S,T) = \frac{exp\left(w^T \cdot f(c,S,T)\right)}{Z(S,T)} \tag{7}$$

where $c \in \{0,1\}$ is a binary variable. $p(c|S,T) \in [0,1]$ is a probability where a value $p(c=1|S,T)$ close to 1.0 indicates that S and T are translations of each other. $w \in \mathbb{R}^n$ is a weight vector obtained during training. $f(c,S,T)$ is a feature vector where the features are co-indexed with respect to the alignment variable c. Finally, $Z(S,T)$ is an appropriately chosen normalization constant, i.e. $Z(S,T) = exp(w^T \cdot f(c=0,S,T)) + exp(w^T \cdot f(c=1,S,T))$. In the feature vector $f(c,S,T)$, each feature f_i occurs twice, once for each class $c \in \{0,1\}$: $c=1$ indicating *parallelism* and $c=0$ indicating *non-parallelism*. In order to apply a scoring function based on a complex feature set over a large set of candidates, we compute the ME score for each link pair incrementally at the word level. The incrementally computed score is then applied in the beam search algorithm in Section 4 in a single pass over the source text. The link scoring function $\varrho(S,T)$ used in the path score computation in (3) is defined as

$$\varrho(S,T) = -w^T \cdot f(c=1,S,T) = -\sum_{j=1}^{J} w^T \cdot f_j(c=1,s_j,T) \tag{8}$$

where the global feature vector computation is split into a sum over the positions j of source chunk S. Note the minus sign in the above definition: in the path cost definition in (3), we are looking to minimize the overall cost. The feature vector $f_j(\cdot)$ is an N dimensional vector with $N = 2 \cdot 11$ real-valued features. The actual features are based on the IBM Model-1 probability computation and are described in Section 5.3 along with the feature vector definitions $f_j(c=\{0,1\},\cdot)$. We use the binary classifier defined in (7) to define a decision rule for chunk pair selection as follows:

$$\begin{aligned} \arg\max_T Pr(T|S,c=1) &= \arg\max_T \frac{Pr(S,T,c=1)}{Pr(S,c=1)} \\ &= \arg\max_T \frac{Pr(c=1|S,T)\cdot Pr(T|S)}{Pr(S,c=1)} \\ &\approx \arg\max_T p(c=1|S,T) \;=\; \arg\min_T \varrho(S,T) \end{aligned} \tag{9}$$

$Pr(T|S,c=1)$ is the probability to choose a chunk T as a translation of source chunk S. Here we have assumed $Pr(T|S)$ to be a constant. For each chunk pair (S_i,T_i) on the final link sequence l_1^N, we compute the classifier probability $p(c=1|S_i,T_i)$ separately and apply a selection threshold. As a result, a link on the globally lowest scoring path can still be discarded, i.e. we consider M-N-type non-aligned links.

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

This differs from related work in the literature where only 0-1- and 1-0-type links are used for un-aligned source and target sentence.

In order to train the classifier, we use a manually annotated collection of sentence pairs with varying degree of parallelism. These collections have been obtained as follows. We run the sentence pair extraction algorithm in Tillmann and Xu (2009) on some portion of the comparable data. The sentence pairs are sorted based on the symmetrized Model-1 score, and we sample sentence pairs across the entire score range. This sentence pair collection is passed to a human annotator. A sentence pair is considered parallel if at least 75 per cent of source and target words have a corresponding translation in the other sentence. Otherwise, it is labeled as non-parallel. The weight vector $w \in \mathbb{R}^N$ in (8) is obtained using the on-line ME training algorithm in Tillmann and Zhang (2007). The classifier performance based on the incremental features is demonstrated in Section 6 for three language pairs: Spanish–English, Russian–English, and Arabic–English.

4 Unified search algorithm

This section presents a novel unified search algorithm. The basic processing pipeline proceeds in a single left-to-right run over the source positions of the input document. This kind of sequential processing is common for monotone Type I alignments, but is extended here to Type III comparable data as well: the data are always processed at the word level. As has been shown in Tillmann (2009) and Tillmann and Xu (2009), the resulting search space is huge. For a given source chunk, the algorithm searches over up to several hundred thousand target chunk candidates, and the overall number of chunk pairs considered reaches several trillion for the extraction tasks presented in Section 6. Typically, the matching search for a single sentence takes a few seconds, such that a single processor may handle up to 50,000 input sentences per day. For all three alignment types, the algorithm finds a global path that corresponds to a link sequence that covers all source document words. The target coverage varies based on the different alignment types. For a Type I and Type II searches, the algorithm uses DP to search over an exponential number of chunk alignments efficiently. For a Type III alignment, the search must consider all possible links at each source input starting position. This can be handled very efficiently, because of the incremental way the scoring features are computed, and the fact that the feature value computation can be shared across different link types that cover the same bitext chunk positions. Here the unified search handles many-to-many link types in a single run over the data along one-to-one links.

Algorithm 1 processes the source document in a single run over all word positions j, where $1 \leq j \leq \mathcal{J}$. It maintains two state lists: the beam of currently active *hypotheses* or states Γ and the list of newly generated hypotheses Γ'. These will become a future beam in the next round of extension. A search state σ is defined as follows:

$$\sigma = [\, l \,; d] \tag{10}$$

where l is a word-level link and d is the partial matching score. In addition, Type II search states include a coverage vector. This simple state definition is

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Algorithm 1 The unified sentence alignment algorithm is a *template*. Two list processing functions adopt the algorithm to the alignment Types I–III. $\mathcal{R}(\cdot)$ specifies how hypotheses are recombined, and $\mathcal{E}(\cdot)$ specifies how a list of hypotheses is extended into new ones. Details are explained in Section 4.

input: Document pairs with $\mathcal{J}$ source and $\mathcal{I}$ target words.
Initialization: $\Gamma := \{\sigma_0 = [l_{ini}; 0.0]\}$
for each source position $j = 0, \ldots, \mathcal{J}$ **do**
Candidate pruning:
  if j is sentence boundary **then**
    Compute candidates: $\Theta(S_m)$ (m-th sentence S_m starts at j)
Matching:
  $\Gamma^{closed} := \{[l; d] \in \Gamma \mid j_{end}(l) = (j-1)\}$
  Separate hypotheses into 'open' and 'complete': $\Gamma = \Gamma^{open} \cup \Gamma^{closed}$
  if $|\Gamma^{closed}| > 0$ **then**
    Apply recombination and book-keep: $\Gamma^{closed} := \mathcal{R}(\Gamma^{closed})$
  Extend complete hypotheses and re-append open: $\Gamma = \mathcal{E}_{\Theta(S)}(\Gamma^{closed}) \cup \Gamma^{open}$
Scoring:
  for each $[l; d] \in \Gamma$ **do**
    New σ' with score $d' = d + \Delta_l(j)$
    $\Gamma' := \Gamma' \cup \sigma'$
  Prune state set Γ'; **Swap** Γ, Γ'; $\Gamma' = \emptyset$;
Output: Extract lowest scoring link sequence l_1^N from trace-back array.

sufficient, as there are no long-range dependencies between links other than the path coverage restrictions for different types of searches. The different alignment types are handled by defining *two* different list processing functions: the *recombine* function and the *extension* function. The 'recombine' function determines the kind of dynamic programming search that is carried out. The 'extension' function determines the types of links and restricted link sequences that enter the search space. In the context of the current sentence alignment algorithm, these two functions generalize related concepts in decoders for SR and SMT in a unifying way. While the overall search algorithm implementation has several thousand lines of code, each list processing function implementation needs only tens of lines of code.

The 'recombine' function $\mathcal{R}(\cdot)$ takes a list of 'complete' hypotheses and keeps only the lowest scoring one for all identical states, i.e. for a monotone search we keep the lowest scoring hypothesis for each link end target position $i = i_{end}(l)$. 'Complete' states correspond to links whose partial score computation has finished covering all source and target positions for the latest link, i.e. whose link end position equals the current source position j: $j_{end}(l) = j$. The 'extension' function $\mathcal{E}(\cdot)$ takes a list of recombined hypotheses that correspond to the lowest-scoring partial link sequence ending at a grid-point (j, i) and extends them by a new link l' covering additional source and target positions, i.e. links for which $j_{beg}(l') = j + 1$. The extension function is used to control the links that enter the search space: either sub-sentential or sentence-level links. Similar list-processing functions are used for DP-based SR

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

search (Ney 1984) and phrase-based SMT decoding (Och *et al.* 2004; Tillmann 2006) algorithms. An algorithmic definition of the recombination function $\mathcal{R}(\cdot)$ and the extension function $\mathcal{E}(\cdot)$ is given in Section 4.1.

Algorithm 1 has three major parts: a *candidate pruning step* (lines 5–6), a *matching* step (lines 8–12), and a *scoring* step (lines 14–17). The candidate pruning step is linked to the extension function $\mathcal{E}(\cdot)$ in the matching step: the pruning candidate set $\Theta(S)$, which is defined in Section 5.1, restricts the possible links that are considered as extensions of a complete partial state. The search starts with a single initial state σ_0 in the list Γ (line 2):

$$\sigma_0 = [l_{ini}; 0.0]$$

where l_{ini} is an initial link that starts at the source document start: $l_{ini} = [-1; -1; -1; -1]$. In line 8, we compute the list Γ^{closed} of all complete hypotheses. For the sentence-level search types (Types I –III), links are conditioned to end at sentence boundaries. For the Type I sub-sentential alignment, a link may start and end anywhere *within* a sentence, but does not cross sentence boundaries. States that are not complete, i.e. for which $j_{beg}(l) \leq j < j_{end}(l)$, are left untouched in the current round of extension and are passed through to the scoring step. These states are kept in a separate list Γ^{open} (line 9). In line 12, complete states are extended one-by-one with additional links based on the candidate set $\Theta(S_m)$. The Type II search uses a target coverage vector $\mathcal{C}$. It is used to keep track of target sentences that have already been aligned. The use of the target coverage vector is similar to the use of a source word coverage vector in SMT decoders (Koehn, Och and Marcu 2003; Och and Ney 2004). Here we require that each target sentence is covered *at most* once (not *exactly* once), i.e. some target sentences are left uncovered. As target documents may consist of thousands of sentences, the permissible coverage vectors need to be restricted. For each source position j, we compute a restricted target coverage vector: a window of ± 10 target positions relative to the so-called coverage focus point is used to compute a restricted coverage vector of length $W = 20$ (computation of the focus point is explained in Section 5.1).

The scoring part of the algorithm is identical for all types of alignments and links. This simplifies the experiments in Section 6. While covering an additional source position j, the target coverage is computed proportionally. Based on the source and target coverage, an incremental matching score $\Delta_l(j) = -w^T \cdot f_j(c = 1, s_j, T)$ is computed (see Figure 4 and (8)). For all search types, the algorithm uses special source null-coverage links for uncovered source chunks. For a monotone search (Type I alignment), target null-coverage links are also needed in order to make sure that a global path that covers an entire document/sentence pair can be computed. Here many-to-many null coverage links are handled as well. The null-coverage score is based on a large null-link constant $\varrho(\cdot, \cdot) = 8.0$. Its value is largely compared to the average classifier-based score. As a result, the unified algorithm tries to cover all source and target positions with as few links as possible. Even though the algorithm computes the lowest scoring global link sequence l_1^N, each link l_i is scored separately based on the $p(c = 1|S, T)$ score, e.g. a link l on the global path is only selected if $p(c = 1|S, T) > \theta$, e.g. $\theta = 0.6$. The current two-stack implementation uses only

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

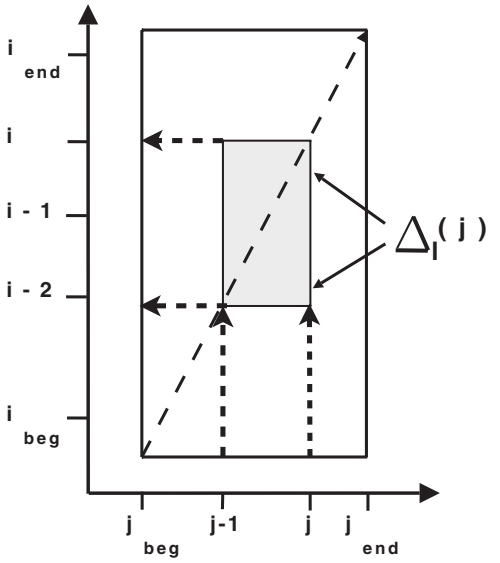


Fig. 4. The partial score computation is based on the link diagonal: the incremental target coverage is computed proportionally to the source coverage. In this example, the link target interval is twice as long as the link source interval. When expanding the source coverage by a single source position j, the partial feature computation is based on the two additional target positions, $i-1$ and i.

constant memory with respect to the source document length. In comparison, the standard implementation of a full Type I search needs a memory of $\mathcal{S} \cdot \mathcal{T}$. A full Type I sub-sentential fragment monotone search implementation would need $\mathcal{J} \cdot \mathcal{I}$ memory. In contrast, the current implementation needs a memory of $2 \cdot \mathcal{T}$ and $2 \cdot \mathcal{I}$, respectively (without the candidate pruning).

An important component of this memory efficient implementation is the so-called trace-back array. Complete states that pass the recombination step are written into this array. Each state contains the so-called back-pointer, which points to that trace-back array entry (not shown in the state definition in (10)). We can re-cover the partial link sequence corresponding to a state by following the sequence of back-pointers in the trace-back array. The trace-back array is implemented as a linear data structure (Ney 1984): A special merge step is used to remove all partial paths that cannot be reached from the states in the set of currently active states Γ. The merge step is important to ensure the constant memory implementation. Once the search is finished, we extract the back-pointer of the lowest scoring state from the final stack at $j = \mathcal{J}$. The link sequence is then extracted by running back the trace-back array (line 18). The trace-back array is especially efficient for the Type III search: for each sentence end position at most a single state is written into the trace-back array. Here the trace-back array can be exploited in a novel way to extract an overlapping link sequence. Instead of tracing-back the link sequence of the lowest scoring final state only (this corresponds to the segmentation of source documents into chunks), we extract all the links in the trace-back array, i.e. the lowest scoring link l for each source end position in the document (if it passes the

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

threshold $p(l) > \theta$). Empirically, this improves the utility of the extracted data as shown in Section 6. The handling of overlapping link sequences is similar to the construction of lattice alternatives in SR and statistical MT. The current approach is able to select link alternatives locally that might otherwise not appear in even a large N-best list. In addition, there is only a minimal computational overhead when extracting overlapping links.

4.1 List processing

This section explains how the recombination function $\mathcal{R}(\cdot)$ and extension function $\mathcal{E}(\cdot)$ are computed for various alignment and link types. The approach to recombination is explained in detail for the Type I alignment search, where states are defined in (10). We define a lexicographical order on the state set Γ: the states are sorted by the target position i. Within each equivalence class, the hypotheses are then sorted by increasing score. After sorting, states σ that are equivalent to each other, i.e. whose partial link sequences end at the same target position i occur at consecutive array positions. In that list, the lowest scoring states of each such equivalence class occur first. In a single run over the list, whenever we encounter the element of a new equivalence class, we know it is the lowest scoring one, and we just keep this state in the reduced list after recombination. We can formally define the partial hypotheses order as follows:

$$[i; d] < [i'; d'] \quad \textbf{iff} \quad (i < i') \tag{11}$$

The equality predicate is defined in terms of the lexicographical order predicate that simplifies the implementation:

$$h' \text{ eq } h \quad \textbf{iff} \quad \neg(h < h') \text{ and } \neg(h' < h) \tag{12}$$

Type II search states can be handled accordingly by assuming a lexicographical order on coverage sets. Here the coverage-based recombination is handled in a way identical to a standard phrase-based SMT decoder, where the source coverage vector is replaced with a target coverage vector. For the Type III search, we just keep the lowest scoring hypotheses across the entire list Γ. Here the space requirements for the recombination algorithm are restricted by the length N of the original array Γ.

The extension algorithm to compute $\mathcal{E}(\cdot)$ is illustrated in Algorithm 2. A list of hypotheses Γ is extended by handling each hypothesis $\sigma \in \Gamma$ individually and appending the resulting list: $\mathcal{E}(\Gamma) := \cup_{\sigma \in \Gamma} \mathcal{E}(\sigma)$. Algorithm 3 computes an auxiliary function. It generates successor links given the bitext ending position $[j_{end}, i_{end}]$ of the preceding link l. Each state $\sigma \in \Gamma$ is extended by $L = |\mathcal{L}|$ possible links, e.g. for a search with the 1-to-4 link set: $L = 9$ (including null links). For the monotone Type I alignment search, the L successor links start where the previous partial link sequence ends. For the Type II search, new sentence links are generated within a window of $W = 20$ target sentence positions. At each of these positions, one of the L different links can start. For the Type III alignment search, we search over a large number of candidates: there is no monotone alignment restriction and each of the L alignment types may start at any target sentence position. This search type can only be carried out in connection with the pruning described in Section 5.1, as

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Algorithm 2 Hypotheses List **Extension** Function. The function computes the successor states for an hypothesis with link l that ends in bitext position $[j_{end}, i_{end}]$. The auxiliary function $\mathcal{F}(\cdot)$ is defined in Algorithm 3. The link-type set $\mathcal{L}$ can be defined at run time. λ is a binary variable, where $\lambda = 0$ indicates a search for sentence-level and $\lambda = 1$ for sub-sentential links.

Input: *Complete* partial hypothesis σ with score d. Link-type set $\mathcal{L}$ and candidate set $\Theta(S_m)$ (m-th source sentence starts at $j_{end} + 1$).
Set $\mathcal{E} = \emptyset$
if (j_{end} is sentence boundary) **then**
  if (Sentence-level Type I) **then**
    $\mathcal{E} := \mathcal{F}(\lambda = 0, d, j_{end} + 1, i_{end} + 1)$
  else
    Use candidate set $\Theta(S_m)$.
    if (Sub-sentential Type I) **then**
      $\mathcal{E} := \bigcup_{l' \in \Theta} \mathcal{F}(\lambda = 1, d, j_{beg}(l'), i_{beg}(l'))$
    else if (Type III) **then**
      $\mathcal{E} := \bigcup_{l' \in \Theta} \mathcal{F}(\lambda = 0, d, j_{beg}(l'), i_{beg}(l'))$
else
  $\mathcal{E} := \mathcal{F}(\lambda = 1, d, j_{end} + 1, i_{end} + 1)$
Output: List of extended hypotheses $\mathcal{E}(\sigma)$.

there might be hundreds of thousands of candidates for a given source sentence. The pruning candidate set is denoted as $\Theta(S_m)$, where S_m is the m-th sentence in the source document. Here the candidate set $\Theta(S_m)$ is computed only once for each S_m. The same candidate set pruning is used for the Type I sub-sentential alignment search. Here the length of sub-sentential links is less than ten for both source and target chunks, i.e. the size of the extension set is $L = 100$. A link l ending at a sentence-internal position is only possible for the sub-sentential search Type I search. These partial sequences are extended monotonically as well. We use the notation $l \in \Theta$ to denote a link l whose bitext starting position coincides with the starting position of one of the candidate links. The Type II alignment search extension is handled similarly as the Type III extension and details are given in Section 5.1.

4.2 Complexity analysis

For the Type III alignment search, in analogy to the use of many-to-many links in Type I alignments, we still *segment* the source document into chunks. There is an exponential number of global paths to be considered as shown in Figure 5. Links that cover neighboring source chunks do not restrict each others' relative target document positions (cf. Figure 3). In particular, we allow the same target chunk to be aligned to several different source chunks ignoring the coverage restrictions of Type I or Type II search. The search is very effective in this case: The list of 'recombined' states collapses to just a single active state as the segmentation task

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Algorithm 3 Link extension function $\mathcal{F}(\cdot)$. The allowable link-type sets $\mathcal{L}$ are defined in Figure 2. The functions $u_S^{-1}(j)$ and $u_T^{-1}(i)$ yield the sentence-level position of document word-level source and target position (The indexing functions $u_S(\cdot)$ and $u_T(\cdot)$ are defined in Section 2.1). As there is no cost for transitions between links, the link score d is just copied into the link extensions.

1: **Input:** Extraction type λ, partial score d, and bitext start point $[j, i]$. Link-type set $\mathcal{L}$.
2: Set $\mathcal{E} = \emptyset$
3: **for** each link tuple $[m; n] \in \mathcal{L}$ **do**
4: **if** $(\lambda = 1)$ **then**
5: Set within sentence link $l = [j, i, j+m, i+n]$
6: **else**
7: Set word-level link $l = [j, i, u_S(j' + m), u_T(i' + n)]$
8: where $j' = u_S^{-1}(j)$; $i' = u_T^{-1}(i)$;
9: $\mathcal{E} := \mathcal{E} \cup [l; d]$
10: **Output:** Extension list $\mathcal{K}$.

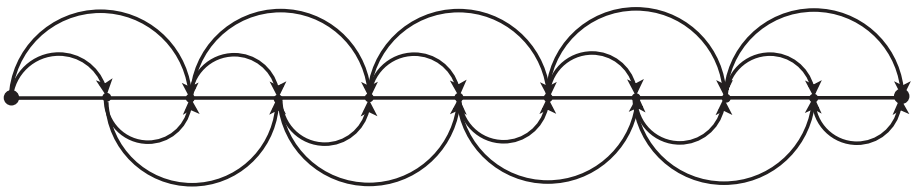


Fig. 5. Illustration of the number of source document segmentations for a Type III alignment with 2-to-1 links. At each source position two source chunks start (with the exception of path end positions). These are matched against at least a single target chunk and are shown as transition links. Here, there are at least $2^{(\mathcal{S}-2)/2}$ different segmentation paths where $\mathcal{S}$ is the number of source sentences.

focuses on the source chunk segmentation only. A possible disadvantage of the global distance definition as given in (3) is that the global path depends on the path length, and thus shorter paths with fewer links N are favored. Ney (1984) points out a similar disadvantage of a global path length criterion. In the context of sentence alignments, it might lead to locally wrong link boundary decisions. In Section 4, we have presented a simple solution to this problem by allowing links to overlap based on the trace-back array usage. This seems to mitigate the problem in the context of one-to-many links and Type III alignments as shown in Section 6.

The complexity of different search types is given in Table 1. The corpus parameters $\mathcal{S}$, $\mathcal{T}$, $\mathcal{J}$, and $\mathcal{I}$ have been defined in Section 2.1. In order to compute the search algorithm complexity for each search type, we count the number of state expansions, i.e. the number of times line 15 in Algorithm 1 is executed. We ignore the complexity for the list processing functions $\mathcal{R}(\cdot)$ and $\mathcal{E}(\cdot)$. These operations can be carried out efficiently, whereas the score computation involves expensive probability look-ups based on large data structures. In Table 1, the 'fan-out' column gives for each complete search state σ, the maximum number of successor states generated due

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Table 1. *Search complexity for different types along with search state definition, 'fan-out' (number of extensions per grid-point), maximum number of hypotheses in the single beam at position j (after recombination), and overall search complexity.* $\tilde{L}$ *is the average number of source words covered by a link.* $W = 20$ *is the size of the Type II re-ordering window. The complexity for the sub-sentential search assumes a monotone alignment across the entire document bitext*

Type	Type	State	Fan-out	Rec	Complexity
I	Sentence-level	$[l;d]$	L	$\mathcal{T}$	$\mathcal{S} \cdot \mathcal{T} \cdot L \cdot \tilde{L}$
	Sub-sentential	$[l;d]$	100	$\mathcal{I}$	$\mathcal{J} \cdot \mathcal{I} \cdot 100 \cdot \tilde{L}$
II	Sentence-level	$[l,C;d]$	$L \cdot W$	2^W	$\mathcal{S} \cdot 2^W \cdot L \cdot \tilde{L}$
III	Sentence-level	$[l;d]$	$L \cdot \mathcal{T}$	1	$\mathcal{S} \cdot \mathcal{T} \cdot L \cdot \tilde{L}$

to the extension function $\mathcal{E}(.)$. The 'Rec' column reports for each search type the maximum number of different hypotheses after the recombination has been carried out for a state list Γ. The complexity given in the last column of Table 1 is derived by multiplying the two preceding 'fan-out' and 'rec' columns. The factors $\mathcal{S}$ and $\mathcal{J}$ take into account that the recombination and extension operations are carried out at sentence boundaries (Types I–III), and possibly all word positions j (Type I sub-sentential alignment). Here, $\tilde{L}$ is the average source chunk length for the L different links. This factor is included as 'link-internal' states are extended deterministically in the scoring step in Algorithm 1. For the sub-sentential Type I search, the average source chunk length is $\tilde{L} = 5$ (since the maximum source chunk length is 10). We note that because of the length factor $\tilde{L}$, the complexity is increased more than linearly in the number of link types.

5 Efficient implementation and pruning

This section presents implementation details that make the sentence-level search efficient. Section 5.1 describes the computation of the beam-search candidate set $\Theta(S)$. Section 5.2 presents techniques to efficiently compute the Model-1-based scores in (5) and (6). Finally, Section 5.3 gives details on the computation of the incremental feature vector $f_j(\cdot)$ in (8).

5.1 Beam-search pruning

Except for the alignment Type I search, we cannot carry out a full search over the bilingual data. The link sequences l_1^N considered during search must be restricted appropriately. To do so, we modify previous work on extracting one-to-one sentence-level translation pairs from comparable data (Tillmann and Xu 2009). During the left-to-right run over the source word-level positions j, whenever we reach the starting position of a sentence S in the source document, we compute the following

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

candidate set of sentence-level 1-to-1-type links l_i:

$$\Theta(S) = \{l_i = (S, T_i) |\ T_i \text{ among the lowest scoring } N \text{ links with score } \varrho'(S, T_i)\ \} \tag{13}$$

where N is a pruning threshold (for our experiment, we pick $N = 25$) and $\varrho'(S, T_k)$ is the extraction score defined in (8) where only the lexical probabilities in two directions are used as features (as in Tillmann and Xu 2009). We also apply a score-based threshold: If $\hat{l}$ is the lowest scoring link for the source sentence S, the candidate set contains only those links l_i for which $\varrho'(l_i) \leq \varrho'(\hat{l}) + 2.0$. The candidate set is also restricted based on the sentence-length ratio introduced by Munteanu and Marcu (2005): for a sentence pair (S, T) to be considered parallel, the ratio of the lengths of the two sentences has to be smaller than two. In addition, at least half of the words in each sentence have to have a translation in the other sentence based on the Model-1 lexicon and a probability threshold of $5 \cdot 10^{-4}$. For the Arabic–English manually annotated training set used in Section 6, the true translation T for a source sentence S is in $\Theta(S)$ for 98 per cent of the source sentences (with all the target sentences T_i as candidates). This candidate set can be computed very efficiently: the early stopping criterion and caching techniques by Tillmann and Xu (2009) are applied appropriately.

For the Type III search, the links considered are restricted based on the candidate set $\Theta(S)$ as follows: Only those links l whose bitext start position is identical to one of the links l_i, i.e. $i_{beg}(l) = i_{beg}(l_i)$ and $j_{beg}(l) = j_{beg}(l_i)$ are considered, i.e. a many-to-many link is considered only if its lower left corner is part of the candidate set. The use of this candidate set in the extension of search hypotheses is shown in Algorithm 2. For the sub-sentential Type I alignment search, the candidate set $\Theta(S)$ restricts the number of sub-sentential links that are considered during search: the sub-sentential links l are covered entirely by one of the candidates l_i, i.e. $l \subseteq l_i$. For the Type II search, we compute the average of all link start positions $i_{beg}(l_i)$. This number defines the coverage focus point, which is used to restrict the permissible target coverage vectors in the search states $[l, \mathcal{C}; d]$. A state σ is extended with links l whose starting position $i_{beg}(l)$ lies within a window of ± 10 positions from that focus point. The target coverage vector $\mathcal{C}$ ensures that each target sentence is aligned to at most a single source sentence. The candidate pruning step is not applied for the Type I sentence-level search where a full search can be carried out.

The candidate pruning step is related to similar pruning steps in SMT decoders. For a given source segment, i.e. a word or a phrase, only a restricted number of translation candidates are considered in the decoder. Apart from the candidate pruning step, no additional pruning is carried out currently, i.e. there is no global pruning based on the partial scores in the current beam at position j. Due to the candidate set computation, the algorithm's memory usage depends on the average source sentence length in a document. Similar candidate pruning techniques have also been introduced for stack-based DP decoders in SR (Ortmanns, Ney and Eiden 1996).

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

5.2 Efficient lexicalized scoring

In order to make feasible the sentence-level processing pipeline described so-far in this paper, the computation of the Model-1 scoring features in (5) and (6) needs to be very efficient. That efficiency is based on the decomposition of the scoring functions into $I + J$ terms (I τ's and J σ's) where source and target terms are treated differently. While the scoring function computation is symmetrical, the processing is organized according the source language positions. This computational 'non-symmetry' is exploited as follows:

- **Array access for source term $\sigma(s, T)$:** The computation of the source term $\sigma(s, T)$ is based on translation probabilities $p(s|t)$. For each source word s, we can retrieve all target words t for which $p(s|t) > 0$ just **once**. Those words t along with their probabilities are then stored in an array of the size of the target vocabulary. Words t that do not have an entry in the lexicon have a 0 entry in that array. We keep a separate array for each source position. This way we reduce the probability access to a simple array look-up. Generating the efficient full array data structure requires less than 50 milliseconds per source sentence.
- **Caching of target term $\tau(t, S)$:** For each target word t that occurs in a candidate translation T, the value $\tau(t, S)$ can be *cached*: its value is independent of the other words in T. The same word t in different target sentences is processed with respect to the same source chunk S and $\tau(t, S)$ has to be computed only once. Compared with Tillmann and Xu (2009), the cache implementation to account for arbitrary links is extended as follows. We compute the caching function across sentence boundaries and store the following value for each document source position j and target word t:

$$c_t(j) = \sum_{k=0}^{j} p(t|s_k)$$

  i.e. we cache a sum of probabilities starting from source position 0. Based on this extended caching function, for any link l, we can compute all the target scores $\tau(t, S)$ in the sum $\sigma_{S \to T}(S, T)$ in (6) in constant time based on the score difference $c_t(j_{end}(l)) - c_t(j_{beg}(l))$. Since a source document may contain hundreds of thousands of words, this caching is carried out on the basis of source language window of 20 sentences.

The efficient computation of the candidate set $\Theta(S)$ also makes use of the caching and array access techniques presented in this section.

5.3 Partial feature vector computation

We select parallel chunk pairs based on the ME classifier defined in (7). It uses the following feature types: *coverage*, *fertility*, and *intersection*. These are computed on the basis of a Model-1 Viterbi alignment where a probability threshold of $\epsilon = 5 \cdot 10^{-4}$ is used to determine word alignment links. The classifier features are defined such

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

that they can be computed incrementally on a position-by-position basis. To do so, we process the bilingual data one source position at a time. When covering a single source position j, we compute the additional covered target positions incrementally (see Figure 4). Just for this incremental position set, we compute the feature values below: for the single source position, we compute the source lexical, fertility, coverage, and romanization features. For a single target position, we compute target lexical, fertility, and coverage features. If the single source position is mapped to several target positions, their feature values are added. In addition, we compute coverage, length, and intersection features for each source position. Once the algorithm has finished the incremental processing of a given link l, features have been computed that account for exactly $|S(l)| + |T(l)|$ link positions. We use the following eleven features:

Lexical Probability ($\mathbf{f_1}, \mathbf{f_2}$) : Two Model-1-based features are derived from the partial sums for the two directions in (5) and (6) : $\sigma(s_j, T)$ and $\tau(t_i, S)$.

Fertility ($\mathbf{f_3}, \mathbf{f_4}$) : We define the fertility of a source word s as the number of target words $t \in T$ for which $p(s|t) > \epsilon$. A high-fertility word indicates non-parallelism. Target word fertility is defined accordingly.

Consecutive Coverage ($\mathbf{f_5}, \mathbf{f_6}$) : A source word s is said to be covered if there is a target word $t \in T$ such that its probability is above a threshold $\epsilon : p(s|t) > \epsilon$. Consecutive coverage is the number of consecutively covered source word positions to the left of the current source position j. Target coverage is defined accordingly. A long run of covered positions indicates parallelism. In order to compute the coverage features, the search state definition in (10) is modified to keep track of the previous source and target word coverage. In the partial feature vector below, if $f_5 < 3$, we use $f_5 = 0$ instead. The feature f_6 is handled accordingly.

Coverage ($\mathbf{f_7}$) : The number of source and target positions covered by the Viterbi path inside a link is used as a feature. High coverage indicates parallelism.

Length ($\mathbf{f_8}, \mathbf{f_9}$) : We include the source and target chunk length ratio as ratio 'penalty' features: J/I and I/J. These features can be compared to the length penalty in regular SMT decoding (Koehn *et al.* 2003).

Romanization ($\mathbf{f_{10}}$) : For each source chunk word s, we compute its Levenshtein distance with respect to all target chunk words $t_i \in T(l)$ at the character level. For languages with non-Latin scripts (e.g. Russian and Arabic), the transliteration into Latin is done using an appropriate scheme, e.g. the Buckwalter scheme for Arabic. For the Spanish data, the string itself is used. The transliterated source string s is denoted by $tr(s)$. The following feature value f_{10} is computed as

$$f_{10} = \left[1.0 - \max_{1 \le i \le |T|} d(tr(s), t_i)\right]$$

Here, $d(t', t_i)$ is the Levenshtein distance between the transliterated source word t' and the i-th target word t_i.

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Table 2. *Corpus statistics for the three languages pairs on which the extraction algorithm is evaluated*

		Sentences		Words	
Language pair	Source	Source	Target	Source	Target
Russian–English	UN	2.47 M	2.38 M	51.91 M	55.22 M
	News	1.07 M	20.49 M	22.03 M	532.70 M
Spanish–English	Gigaword	21.36 M	66.45 M	643.38 M	1.504 B
Arabic–English	Gigaword	5.32 M	37.31 M	169.03 M	1.495 B

Intersection ($\mathbf{f_{11}}$) : The intersection feature is defined as the number of alignment points that occur on both Viterbi Model-1 alignments: source to target $S \rightarrow T$ and target-to-source $T \rightarrow S$.

Empirically, we find that normalizing the incremental feature values with respect to source chunk length J or target chunk length I improves classification accuracy. The sentence length normalization has been applied to all the features in our experiments. We define an auxiliary feature vector $\bar{f}_j$ as follows:

$$\bar{f}_j = (f_1, f_2, f'_3, f_4/I, f'_5, f'_6, f'_7, f'_8, f'_9, f'_{10}, f'_{11}) \tag{14}$$

where $f' = f/J$. The features f_1 and f_2 have already been normalized by definition. The target fertility feature has been normalized by target length I. This particular normalization has been found to be optimal with respect to F-score on the classifier training data. The auxiliary feature vector has been used to define the class-dependent feature vectors in Section 3 as follows: $f_j(c = 1, s_j, T) = (\bar{f}_j, \bar{n})$ and $f_j(c = 0, s_j, T) = (\bar{n}, \bar{f}_j)$, where $\bar{n}$ is a 11-dimensional 'null' vector with all entries set to 0.0. The advantage of this definition is that the 'negative' features can be ignored during search based on (8), as their feature values are zero.

6 Experiments

In this section, we report classification and extraction results for three language pairs: Russian–English, Spanish–English and Arabic–English.

6.1 Data sources and classifier training

The Russian data come from two sources: United Nations (UN) parallel documents and some news data. The Spanish–English and the Arabic–English data are based on the Gigaword data for these languages (Mendonca, Graff and DiPersio 2009; Parker *et al.* 2009). The corpus statistics are shown in Table 2. Here, the Russian–English UN data consist of 16,346 document pairs. For all the languages, the data are tokenized and number-classed (i.e. numbers are replaced with a single tag $num), and the bilingual matching features are computed on this pre-processed data. In addition, if the data does not have sentence-boundary annotation, we use

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Table 3. *Classifier evaluation in terms of precision/recall/F-score on the manually annotated data for the three languages pairs. The feature functions are defined in Section 5.3*

Pair	Features	Precision	Recall	F-score
Spanish–English	T2S, S2T	88.59	86.84	87.71
	+Src/Tgt Fertility	89.19	86.84	88.00
	+Src/Tgt Cons. Coverage	86.42	92.11	89.17
	+Coverage	90.97	86.18	88.51
	+Length	92.14	84.87	88.36
	+Romanization	92.41	88.16	90.24
	+Intersection	92.52	89.47	90.97
Arabic–English	T2S,S2T	91.30	96.00	93.59
	+Src/Tgt Fertility	91.30	96.00	93.59
	+Src/Tgt Cons. Coverage	91.71	94.86	93.26
	+Coverage	93.18	93.71	93.45
	+Length	93.18	93.71	93.45
	+Romanization	94.71	92.00	93.33
	+Intersection	93.22	94.29	93.75
Russian–English	T2S,S2T	93.78	98.49	96.08
	+Src/Tgt Fertility	93.78	98.49	96.08
	+Src/Tgt Cons. Coverage	94.15	96.98	95.54
	+Coverage	96.06	97.99	97.01
	+Length	95.57	97.49	96.52
	+Romanization	95.57	97.49	96.52
	+Intersection	96.08	98.49	97.27

punctuation information to split a text stream into sentences. In Table 3, we present classification results for the ME classifier described in Section 3 in terms of precision, recall, and F-score. The table contains the results for all three language pairs.The size of the annotated data for the Spanish–English, Russian–English, and Arabic–English data is 950, 1,030, and 1,600 sentence pairs, respectively. Roughly 80 per cent of the data sets are used to train the classifier and the rest are used for testing. Here the classifier selection threshold θ is set on the training data and lies in a range of $[0.6, 0.75]$ for the different language pairs. The table demonstrates the effect of using different feature sets. The simplest model uses just two lexical probability features, f_1, f_2 from Section 5.3. Each line adds an additional feature, i.e. the final classifier uses all the features, including the Viterbi alignment intersection feature. Including additional features does not always improve classification accuracy. The largest improvements in terms of F-score are reported for the Spanish–English data. In particular, the romanization feature improves accuracy by almost 2 per cent for this European language pair. Another important feature is the intersection feature. For the three language pairs, there is a 0.4 to 0.7 per cent improvement over the classifier results where this feature is not used. For the extraction experiments reported later in this section, we always use the classifier with the full feature set.

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

6.2 Translation experiments

We evaluate the performance of the unified search algorithm for different alignment and link types in terms of BLEU (Papineni *et al.* 2002) test set performance. Based on the unified alignment algorithm, chunk pairs are collected from the un-aligned, document-level aligned or comparable data, and this data are added to a baseline system. Using the ME-based extraction algorithm on data with different granularity and types of sentence-level alignments leads to an increased extraction accuracy and/or extraction yield. Only for the Russian–English UN data, there is an alignment at the document level. As the sentence alignment for these document pairs is largely monotone, a search for the alignment Types I and II is used on this data (alignment types have been defined in Section 2). Here, the largest document pair consists of up to 10,000 source and target sentences. For the Type II alignment search, the beam at a given word position j may contain tens of thousands of states. As the average sentence length is around thirty words for these source sentences, the overall search space reaches billions of states. Processing on such a document pair takes about 1 hour on a single processor. In comparison, for the Russian–English news data, no document boundaries are given and the data can only be processed at the sentence level. For each Russian source sentence, only those target sentences for which the publication date differs less than three days from the publication date of the source sentence, are considered as translation candidates. For the Spanish–English Gigaword data, we also match chunk pairs that have been published from the same news feed, e.g. from a specific news agency like AFP. In addition, the beam-search pruning in Section 5 is used to speed up the search. For this comparable data, even though there is no document alignment, the original sentence order is partially preserved and potentially good many-to-many links can be found. For this data, a sentence-level search for Type III alignments as well as a sub-sentential search for Type I alignments are carried out. Similar experiments are carried out for the comparable English–Spanish and Arabic–English data, which comes from the Gigaword corpora for the years 2003–2008. Again, no document-level alignment is provided for this data. The same three-day publication date filter and sentence-length filter are used. Using a publication date filter of five days increased the number of sentences only slightly.

For the Spanish–English data, already for the one-to-one link search, the search space is huge: link candidates are selected out of 1.08 trillion candidate pairs. There are about 20 million source sentences each of which may have hundreds of thousands of target sentence candidates after the date filter and length filter steps. Since L candidates are considered at each grid point, the overall candidate set of many-to-many links may reach several trillions. Processing the English–Spanish data takes at most two days on a cluster with 300 processors. Processing the Arabic–English data takes less then a single day on the same cluster, while processing the Russian–English UN data takes only a few hours where the un-pruned Type I search runs fastest.

Extraction results are presented for the Russian–English data in Table 4, for the Spanish–English data in Table 5, and for the Arabic–English data in Table 6. The

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Table 4. *BLEU scores for Russian–English phrase-based SMT systems trained on the baseline data plus some extracted data (UN or news)*

	Alignment type	Data	Link	Overlap	# Pairs	# Src	# Tgt	BLEU (%)
1	Baseline		-	-	1,470,604	15.03 M	17.56 M	7.34
2	**I**	UN	**1-1**	No	374,762	8.85 M	9.97 M	**14.22**
3			**1-4**	No	513,078	11.46 M	13.68 M	**14.87**
4			**4-4**	No	445,968	9.96 M	11.83 M	**14.75**
5			Frag	No	3,061,270	22.23 M	24.12 M	**16.09**
6	**II**		**1-1**	No	889,161	21.25 M	23.64 M	**14.86**
7			**1-4**	No	821,503	20.66 M	22.52 M	**14.56**
8			**4-4**	No	797,547	21.13 M	23.14 M	**14.77**
9	**III**		**1-1**	No	1,116,478	27.21 M	30.81 M	**15.20**
10			**1-4**	No	1,110,856	25.94 M	35.84 M	**14.93**
11				**Yes**	1,119,229	25.82 M	37.23 M	**15.16**
12	**I**	News	Frag	No	237,115	3.22 M	3.30 M	**12.90**
13	**III**		**1-1**	No	164,496	3.74 M	4.11 M	**12.69**
14			**1-4**	No	111,710	2.49 M	3.14 M	**12.70**
15				**Yes**	121,766	2.65 M	3.55 M	**12.87**

Table 5. *BLEU scores for various Spanish–English phrase-based SMT systems trained on the baseline data plus some extracted data (Gigaword)*

	Alignment type	Link	Overlap	# Pairs	# Src	# Tgt	BLEU (%)
1	Baseline	-	-	1,825,709	48.27 M	46.04 M	42.42
2	**I**	Frag	No	4,036,258	41.03 M	38.44 M	**45.77**
3	**III**	**1-1**	No	2,190,448	56.35 M	51.67 M	**46.14**
4		**1-4**	No	1,821,331	44.97 M	52.79 M	**45.40**
5			**Yes**	2,000,592	48.90 M	58.05 M	**45.71**

Table 6. *BLEU scores for various Arabic–English phrase-based SMT systems trained on the baseline data plus some extracted data (Gigaword)*

	Alignment type	Link	Overlap	# Pairs	# Src	# Tgt	BLEU (%)
1	Baseline	-	-	8,032,837	237.28 M	233.36 M	30.13
2	**I**	Frag	No	1,920,922	11.34 M	11.33 M	30.27
3	**III**	**1-1**	No	870,880	23.33 M	26.78 M	30.33
4		**1-4**	No	736,835	20.31 M	28.35 M	30.35
5			**Yes**	798,763	21.00 M	30.63 M	30.32

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Model-1 probability models used to compute the feature functions in Section 5.3 are trained on the baseline data as well. The Model-1 training has been carried out for two iterations, and the dictionaries are un-pruned and include function words. The Russian–English dictionaries include 97 and 73 million entries, and the Spanish–English dictionaries include 73 and 87 million entries (Source-to-English and English-to-Source, respectively). The use of large un-pruned dictionaries improved extraction recall in preliminary experiments. In the first line of each table we report the number of parallel sentences along with the number of source and target words for the baseline data. On the baseline data, we train a phrase-based system and report its lower cased BLEU score (Papineni *et al.* 2002) on the test sets: for the Russian–English and the Spanish–English data, we use manually collected web news data. The Russian–English data consist of 553 sentences with 20,602 words and a single reference translation. The Spanish–English data consist of 414 sentences with 13,474 words and a single reference translation as well. For the Arabic–English data, we use news wire broadcast data collected for the GALE (Olive *et al.* 2011) project. It consists of 683 sentences with 22,237 words and four reference translations. In terms of parallel data extraction, we have carried out experiments for a combination of different alignment and link types. For each experiment, we report the number of extracted chunk pairs as well as the number of source and target words in the extracted parallel data. The newly extracted data are added to the baseline parallel data (we have not tried to combine the data from different extraction runs). Including the extracted parallel data leads to a significant improvement over the baseline system in all cases. In addition, when searching for Type III sentence-level alignments, we allow many-to-many links to overlap. This novel concept leads to some small improvements in the BLEU score. In Section 4, the use of overlapping link sequences based on the trace-back array has been explained in more detail.

For the Russian–English results in Table 4, Types I and II alignments are computed for the UN data. Three different link types are extracted: one-to-one, one-to-many, and many-to-many (as shown in Figure 2) where up to four source sentences are aligned to single target sentence and *vice versa.* Here, the monotone Type I alignment search results are compared with the Type II search results. The Type II search increases the number of chunk pairs significantly. The additional extracted data improve the BLEU score by about 0.6 per cent only for **1-1** link types (line 2 *vs.* line 6). Even on the largely monotone UN data, the Type III alignment search actually outperforms the monotonic search types: the document pairs are several thousand sentences long, and some sentence re-ordering cannot be handled by the Type II re-ordering window. On this data, the Type I sub-sentential alignment search yields the best BLEU scores (see line 5). For the sub-sentential search, the allowable maximum length ratio has been increased to 4.0 and the coverage threshold has been reduced to 0.33. On the Russian–English news data (results in line 12–15), the sub-sentential search again yields the best extraction results. We carried out pairwise bootstrap re-sampling to determine whether some of the smaller BLEU differences for different search types are statistically significant. The extraction of sub-sentential fragment pairs improved the BLEU score significantly at the 95 per cent level (line 5 *vs.* line 9 and line 12 *vs.* line 13). In addition, the extraction of overlapping links

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

improved the BLEU score significantly at the 95 per cent level as well (line 10 *vs.* line 11 and line 14 *vs.* line 15). In addition, on the Russian–English news data, we trained an SMT system on the extracted data only. This lead to only a minor drop in translation quality, as the Russian–English test data were taken from the news domain as well.

For the Spanish–English Gigaword experiments in Table 5, the Type III alignment search with **1-1** links performs best (line 3), followed by the sub-sentential search (line 2). Training a phrase-based system on just the extracted data from line 3 leads to lower BLEU performance of 45.45 (down from 46.14). While still yielding a significant BLEU improvement over just using the baseline data, the Type III alignment search with many-to-many links performs worst on this data. Some improvements are obtained by allowing the many-to-many links to overlap (line 4 *vs.* line 5). In future, we will try to improve translation results by allowing links to overlap for all the link and alignment types, not just the Type III search with many-to-many links. The Arabic–English Gigaword results in Table 6 also show some small improvement over just using the baseline data. Here, already the baseline training data consist of 8 million parallel sentence pairs with more than 200 million words for both source and target language. Consequently, the overall BLEU gains from including the extracted data are small, and the difference between the alignment and link search types are small as well.

7 Comparison and future work

The paper presents a novel one-pass, two-stack beam-search algorithm for extracting parallel text chunk pairs. It unifies several related algorithms into a single framework, and is capable of searching over several trillion link candidates. Here the computation is carried out in parallel over 300 processors. The unified search extends previous work, namely Tillmann (2009). In that paper only one-to-one sentence-level links from comparable data are handled, i.e. a special case of the Type III alignment search in this paper. Tillmann (2009) presents results that show that avoiding the document-level filtering increases the number of extracted parallel sentence pairs for the Spanish–English Gigaword data. The additional data result in an increased BLEU score as well. The current paper focuses on comparing the effect of different alignment and link types. The beam-search extraction algorithm relies on simple word-based scoring features that can be computed incrementally. In future work, the state definition in (10) can be modified to account for coverage as well as Model-2 or HMM-type distortion model features. In general, the current algorithm is applicable to all extraction methods whose selection features can be computed incrementally. In order to investigate the effect of using incremental features only, the algorithm might also be evaluated in terms of segmentation accuracy on manually annotated data.

It is interesting to compare the size of the extraction classifier search space to the search space that is handled in regular SMT or SR decoding. As is shown in this paper, for a single source sentence the size of the target candidate set might be more than a 100,000 sentences. In comparison, a SR decoder might have to

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

deal with recognition vocabularies that have up to 100,000 entries. The number of feature vectors per input pattern is also of the same order of magnitude, i.e. tens of feature vectors. In the current beam-search algorithm, the link-internal matching is deterministic as shown in Figure 4, i.e. this component can not be compared with the word-internal time warping in SR decoders. In future, a 'fuzzy' match between source and target sentences could be carried out, e.g. when searching for chunk pairs with gaps. The way the current beam-search decoder computes an alignment score incrementally can be compared with standard phrase-based decoding in SMT. Here, incrementally generated features can obtain close to state-of-the-art performance, which supports their use in this paper. It is also interesting to compare the chunk pair extraction to the extraction of phrase pairs for SMT (Koehn *et al.* 2003). The results in Section 6 show that the lexical Model-1 scoring features defined in Section 3 attain close to the top classification performance. Those lexical scoring features are also commonly used to score phrase pairs in SMT decoding. Likewise, the use of the intersection feature improves classification performance: phrase pairs in SMT are extracted from intersecting word alignments as well. In addition, the paper introduces the following novel concept: In the context of extracting one-to-many sentence pairs on comparable data, the experimental results in Section 6 demonstrate that allowing pairs to overlap in a controlled way actually improves translation results. Again, this method can be related to standard models for phrase-based SMT (Koehn *et al.* 2003; Och *et al.* 2004), where the phrase pairs extracted from parallel sentences may overlap based on certain extraction heuristics. In future work, the current chunk-level segmentation algorithm might be improved by allowing overlapping link sequences, in particular when extracting sub-sentential chunk pairs (those could be compared to phrase pairs in SMT with particular long phrases). In addition, the unified algorithm could be extended to search for mixed Types I–III alignment types within a single document pair, which would result in an even more general algorithm.

Similar to Melamed (1999), the current paper handles the extraction task as a pattern recognition problem. Here our one-pass algorithm is conceptually simpler, since the search in that paper alternates between a generation and a recognition phase. A multi-pass beam-search algorithm (similar to Type I) is presented by Moore (2002). The author uses one-to-one links to bootstrap their lexicon model. Similar bootstrapping techniques might be used in the current work as well. Pike and Melamed (2004) present a two-pass algorithm that handles both parallel and comparable data, but it is tested on a small, artificial test set in terms of extraction accuracy. Deng *et al.* (2006) describes an approach for segmenting and simultaneously aligning chunks of source and target texts (including sub-sentential chunks). Their paper considers two search strategies: a monotone DP-based search, and an iterative binary search. The iterative binary search is called *divisive clustering* (DC) search, and it assumes the monotone search as a pre-processing step. It is similar to the Type II search as some non-monotone chunk alignment can be handled. In contrast, the Type II search is based on a coverage vector. Unlike the DC search, it guarantees to find the optimal global segmentation with an appropriately chosen re-ordering window. While Deng *et al.* (2006) do not demonstrate any results on

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

comparable data, the current paper focuses on a unified one-stage search algorithm. Brown *et al.* (1991) also state the use of a constant memory implementation using techniques from Brown *et al.* (1982), but no algorithmic details are given. Hewavitharana and Vogel (2011) compare several alignment approaches to extract parallel phrase pairs from sentences that are partially parallel (e.g. sentences marked with red broken lines in Figure 1). In future, we hope that reducing the amount of pre-filtering based on our novel pruning techniques will be especially important for extracting sub-sentential pairs.

8 Acknowledgment

We would like to thank our colleague Young-Suk Lee for suggesting the use of the intersection feature in Section 5.3 and for helpful discussions. We would also like to thank our colleague Yaser Al-Onaizan for his useful comments on our work.

References

Brown, P., Spohrer, J., Hochschild, P., and Baker, J. 1982. Partial traceback and dynamic programming. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 82)*, Paris, France, pp. 1629–32.

Brown, P. F., Lai, J. C., and Mercer, R. L. 1991. Aligning sentences in parallel corpora. In *Proceedings of ACL 91*, Berkeley, CA, pp. 169–76.

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

Chen, S. F. 1993. Aligning sentences in bilingual corpora using lexical information. In *Proceedings of ACL 93*, June 16–17, Columbus, OH, pp. 9–16.

Deng, Y., Kumar, S., and Byrne, W. 2006. Segmentation and alignment of parallel text for statistical machine translation. *Natural Language Engineering* **12**(4): 1–26.

Fung, P., and Cheung, P. 2004. Mining very-non-parallel corpora: parallel sentence and lexicon extraction via bootstrapping and EM. In *Proceedings of EMNLP 04*, July 25–26, Barcelona, Spain, pp. 57–63.

Hewavitharana, S., and Vogel, S. 2011. Extracting parallel phrases from comparable data. In *Proceedings of ACL Workshop on Building and Using Comparable Corpora*, June 24, Portland, OR, pp. 61–8.

Gale, W. A., and Church, K. W. 1991. A program for aligning sentences in bilingual corpora. In *Proceedings of ACL 91*, June 18–21, Berkeley, CA, pp. 177–84.

Koehn, P., Och, F. J., and Marcu, D. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 03*, May 27–June 1, Edmonton, Alberta, Canada, pp. 127–33.

Koehn, P. 2004. Pharaoh: a beam search decoder for phrase-based SMT models. In *Proceedings of AMTA 04*, September 28–October 2, Washington DC.

Ma, X. 2006. Champollion: a robust parallel text sentence aligner. In *Proceedings of LREC 06*, May 22–28, Genova, Italy, pp. 489–92.

Melamed, I. D. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics* **25**(1): 107–30.

Mendonca, A., Graff, D., and DiPersio, D. 2009. *Spanish Gigaword Corpus,* 2nd ed., LDC catalog no. 2009T21. Philadelphia, PA: LDC.

Moore, R. C. 2002. Fast and accurate sentence alignment of bilingual data. In *Proceedings of AMTA 05*, Tiburon, CA, pp. 135–44.

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press

Munteanu, D. S., and Marcu, D. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics* **31**(4): 477–504.

Munteanu, D. S., and Marcu, D. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of COLING/ACL 06*, July 17–21, Sydney, Australia, pp. 81–8.

Ney, H. 1984. The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **32**(2): 263–71.

Och, F.-J., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics* **30**(4): 417–50.

Och, F. J. *et al.* 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Joint HLT and NAACL Conference (HLT 04)*, May 2–7, Boston, MA, pp. 161–8.

Olive, J., Christianson, C., and McCary, J. (Editors). 2011. *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation.* New York: Springer.

Ortmanns, S., Ney, H., and Eiden, A. 1996. Language-model look-ahead for large vocabulary speech recognition. In *Proceedings of ICASSP 96*, May 7–9, Atlanta, GA, pp. 2095–8.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL 02*, July 7–12, Philadelphia, PA, pp. 311–18.

Parker, R., Graff, D., Kong, J., Chen, K., and Maeda, K. 2009. *English Gigaword Corpus,* 4th ed., LDC catalog no. 2009T13. Philadelphia, PA: LDC.

Pike, C., and Melamed, I. D. 2004. An automatic filter for non-parallel texts. In *The Comp. Volume of the Proceedings of ACL 04*, July 21–26, Barcelona, Spain, pp. 114–17.

Quirk, C., Udupa, R., and Menezes, A. 2007. Generative models of noisy translations with applications to parallel fragment extraction. In *Proceedings of the MT Summit XI*, September 10–14, Copenhagen, Demark, pp. 321–7.

Resnik, P., and Smith, N. 2003. The web as parallel corpus. *Computational Linguistics* **29**(3): 349–80.

Snover, M., Dorr, B, and Schwartz, R. 2008. Language and translation model adaptation using comparable corpora. In *Proceedings of EMNLP08*, October 25–27, Honolulu, HI, pp. 856–5.

Tillmann, C., and Xu, J.-M. 2009. A simple sentence-level extraction algorithm for comparable data. In *Proceedings of HLT/NAACL 09*, May 31–June 5, Boulder, CO, pp. 93–6.

Tillmann, C., and Zhang, T. 2007. A block bigram prediction model for statistical machine translation. *ACM-TSLP* **4**(6): 1–31 (July).

Tillmann, C. 2006. Efficient dynamic programming search algorithms for phrase-based SMT. In *Proceedings of the Workshop CHPSLP at HLT 06*, June 4–9, New York City, NY, pp. 9–16.

Tillmann, C. 2009. A beam-search extraction algorithm for comparable data. In *Proceedings of the ACL-IJCNLP 2009 Conference*, August 2–7, Suntec, Singapore, pp. 225–8.

Utiyama, M., and Isahara, H. 2003. Reliable measures for aligning Japanese–English news articles and sentences. In *Proceedings of ACL 03*, July 7–12, Sapporo, Japan, pp. 72–9.

Zhao, B., and Vogel, S. 2002. Adaptive parallel sentences mining from WebBilingualNewsCollection. In *IEEE International Conference on Data Mining (ICDM 2002)*, December 2–12, Maebashi City, Japan, pp. 745–8.

https://doi.org/10.1017/S135132491100026X Published online by Cambridge University Press