Bitext Maps and Alignment via Pattern Recognition

I. Dan Melamed*
West Group

*Texts that are available in two languages (*bitexts*) are becoming more and more plentiful, both in private data warehouses and on publicly accessible sites on the World Wide Web. As with other kinds of data, the value of bitexts largely depends on the efficacy of the available data mining tools. The first step in extracting useful information from bitexts is to find corresponding words and/or text segment boundaries in their two halves (*bitext maps*).*

This article advances the state of the art of bitext mapping by formulating the problem in terms of pattern recognition. From this point of view, the success of a bitext mapping algorithm hinges on how well it performs three tasks: signal generation, noise filtering, and search. The Smooth Injective Map Recognizer (SIMR) algorithm presented here integrates innovative approaches to each of these tasks. Objective evaluation has shown that SIMR's accuracy is consistently high for language pairs as diverse as French/English and Korean/English. If necessary, SIMR's bitext maps can be efficiently converted into segment alignments using the Geometric Segment Alignment (GSA) algorithm, which is also presented here.

SIMR has produced bitext maps for over 200 megabytes of French-English bitexts. GSA has converted these maps into alignments. Both the maps and the alignments are available from the Linguistic Data Consortium.[1]

1. Introduction

> Existing translations contain more solutions to more translation problems than any other existing resource (Isabelle 1992).

Although the above statement was made about translation problems faced by human translators, recent research (Brown et al. 1993; Melamed 1996b) suggests that it also applies to problems in machine translation. Texts that are available in two languages (**bitexts**) (Harris 1988) also play a pivotal role in various less automated applications. For example, bilingual lexicographers can use bitexts to discover new cross-language lexicalization patterns (Catizone, Russell, and Warwick 1993; Gale and Church 1991b); students of foreign languages can use one half of a bitext to practice their reading skills, referring to the other half for translation when they get stuck (Nerbonne et al. 1997). Bitexts are of little use, however, without an automatic method for matching corresponding text units in their two halves.

The bitext mapping problem can be formulated in terms of pattern recognition. From this point of view, the success of a bitext mapping algorithm hinges on three tasks: signal generation, noise filtering, and search. This article presents the Smooth Injective Map Recognizer (SIMR), a generic pattern recognition algorithm that is partic-

* 610 Opperman Drive, #D1-66F, Eagan, MN, 55123

[1] See `http://www.ldc.upenn.edu/ldc/catalog/html/text_html/hansfreng.html`

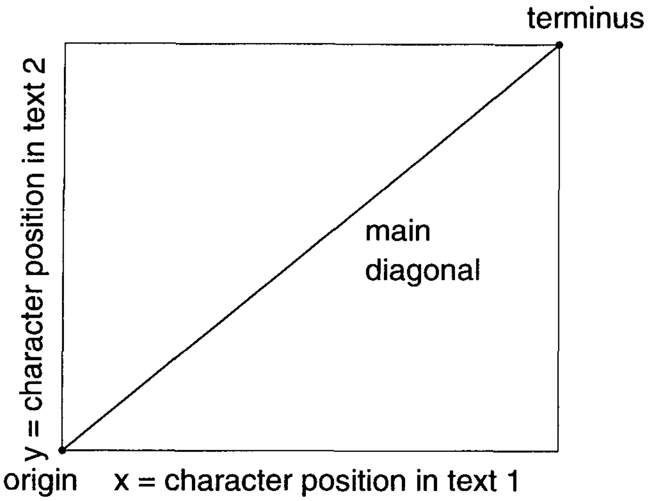
terminus
y = character position in text 2
main
diagonal
origin
x = character position in text 1

Figure 1
A bitext space.

ularly well suited to mapping bitext correspondence. SIMR demonstrates that, given effective signal generators and noise filters, it is possible to map bitext correspondence with high accuracy in linear space and time. If necessary, SIMR can be used with the Geometric Segment Alignment (GSA) algorithm, which uses segment boundary information to reduce general bitext maps to segment alignments. Evaluations on preexisting gold standards have shown that SIMR's bitext maps and GSA's alignments are more accurate than those of comparable algorithms in the literature.

The article begins with a geometric interpretation of the bitext mapping problem and a discussion of previous work. SIMR is detailed in Section 4 and evaluated in Section 6. Section 7 discusses the formal relationship between bitext maps and segment alignments. The GSA algorithm for converting from the former to the latter is presented in Section 7 and evaluated in Section 8.

2. Bitext Geometry

Each bitext defines a rectangular **bitext space**, as illustrated in Figure 1. The lower left corner of the rectangle is the **origin** of the bitext space and represents the two texts' beginnings. The upper right corner is the **terminus** and represents the texts' ends. The line between the origin and the terminus is the **main diagonal**. The slope of the main diagonal is the **bitext slope**.

Each bitext space is spanned by a pair of **axes**. The lengths of the axes are the lengths of the two component texts. The axes of a bitext space are measured in characters, because text lengths measured in characters correlate better than text lengths measured in tokens (Gale and Church 1991a). This correlation is important for geometric bitext mapping heuristics, such as those described in Section 4.4. Although the axes are measured in characters, I will argue that word tokens are the optimum level of analysis for bitext mapping. By convention, each token is assigned the position of its median character.

Each bitext space contains a number of **true points of correspondence (TPCs)**, other than the origin and the terminus. TPCs exist both at the coordinates of matching text units and at the coordinates of matching text unit boundaries. If a token at position p on the x-axis and a token at position q on the y-axis are translations of each other, then the coordinate (p, q) in the bitext space is a TPC. If a sentence on the x-axis ends at character r and the corresponding sentence on the y-axis ends at character s, then the coordinate $(r + .5, s + .5)$ is a TPC. The .5 is added because it is the inter-sentence boundaries that correspond, rather than the last characters of the sentences. Similarly, TPCs arise from corresponding boundaries between paragraphs, chapters, list items, etc. Groups of TPCs with a roughly linear arrangement in the bitext space are called **chains**.

Bitext maps are injective (1-to-1) partial functions in bitext spaces. A complete set of TPCs for a particular bitext is the **true bitext map (TBM)**. The purpose of a **bitext mapping algorithm** is to produce bitext maps that are the best possible approximations of each bitext's TBM.

3. Previous Work

Early bitext mapping algorithms focused on finding corresponding sentences (Debili and Sammouda 1992; Kay and Röscheisen 1993). Although sentence maps are too coarse for some bitext applications (Melamed 1996a; Macklovitch 1996), sentences were a relatively easy starting point, because their order rarely changes during translation. Therefore, most sentence mapping algorithms ignore the possibility of crossing correspondences and aim to produce only an **alignment**. Given parallel texts U and V, an alignment is a segmentation of U and V into n segments each, so that for each $i, 1 \leq i \leq n$, u_i and v_i are mutual translations. An **aligned segment pair** a_i is an ordered pair (u_i, v_i). Thus, an alignment A can also be defined as a sequence of aligned segments: $A \equiv \langle a_1, \ldots, a_n \rangle$. In 1991, two teams of researchers independently discovered that sentences from bitexts involving clean translations can be aligned with high accuracy just by matching sentence sequences with similar lengths (Brown, Lai, and Mercer 1991; Gale and Church 1991a). Both teams approached the alignment problem via maximum-likelihood estimation, but using different models.

Brown, Lai, and Mercer (1991) formulated the problem as a hidden Markov model (HMM), based on a two-stage generative process. Stage one generated some number of aligned segment pairs; stage two decided how many segments from each half of the bitext to put in each aligned segment pair. Brown, Lai, and Mercer (1991) took advantage of various lexical "anchors" in the bitext that they were experimenting with. These anchors were also generated by the HMM, according to their respective probability functions. All the hidden variables were estimated using the EM algorithm (Dempster, Laird, and Rubin 1977).

Gale and Church (1991a) began with a less structured model and proceeded to estimate its parameters through a series of approximations. Given the set $\mathcal{A}$ of all possible alignments, the maximum-likelihood alignment is

$$A_{max} = \arg\max_{A \in \mathcal{A}} \Pr(A|U, V). \tag{1}$$

Gale and Church first assumed that the probability of any aligned segment pair is independent of any other segment pair:

$$A_{max} = \arg\max_{A \in \mathcal{A}} \prod_{i=1}^{|A|} \Pr(a_i|u_i, v_i). \tag{2}$$

Next, they assumed that the only feature of u_i and v_i that influences the probability of

Table 1
Alignment algorithms that don't look at the words can fumble in bitext regions like this vote record. Source: Chen (1996)

English	French
⋮	⋮
Mr. McInnis?	M. McInnis?
Yes.	Oui.
Mr. Saunders?	M. Saunders?
No.	Non.
Mr. Cossitt?	M. Cossitt?
Yes.	Oui.
⋮	⋮

their alignment is a function $d(u_i, v_i)$ of the difference in their lengths, in characters:

$$A_{max} = \arg\max_{A \in \mathcal{A}} \prod_{i=1}^{|A|} \Pr(a_i | d(u_i, v_i)). \tag{3}$$

By Bayes' rule,

$$A_{max} = \arg\max_{A \in \mathcal{A}} \prod_{i=1}^{|A|} \frac{\Pr(d(u_i, v_i)|a_i) \Pr(a_i)}{\Pr(d(u_i, v_i))}. \tag{4}$$

Ignoring the normalizing constant $\Pr(d(u_i, v_i))$ and taking the logarithm, Gale and Church arrived at

$$A_{max} = \arg\max_{A \in \mathcal{A}} \sum_{i=1}^{|A|} \log \Pr(d(u_i, v_i)|a_i) \Pr(a_i). \tag{5}$$

Gale and Church empirically estimated the distributions $\Pr(d(u_i, v_i)|a_i)$ and $\Pr(a_i)$ from a hand-aligned training bitext and then used dynamic programming to solve Equation 5.

The length-based alignment algorithms work remarkably well on language pairs like French/English and German/English, considering how little information they use. However, length correlations are not as high when either of the languages involved does not use a phonetically based alphabet (e.g., Chinese). Even in language pairs where the length correlation is high, length-based algorithms can fumble in bitext regions that contain many segments of similar length, like the vote record in Table 1. The only way to ensure a correct alignment in such cases is to look at the words. For this reason, Chen (1996) added a statistical translation model to the Brown, Lai, and Mercer alignment algorithm, and Wu (1994) added a translation lexicon to the Gale and Church alignment algorithm.

A translation lexicon T can be represented as a sequence of t entries, where each entry is a pair of words: $T \equiv \langle (x_1, y_1), \ldots, (x_t, y_t) \rangle$. Roughly speaking, Wu (1994) extended Gale and Church's (1991a) method with a matching function $m(u, v, j)$, which was equal to one whenever $x_j \in u$ and $y_j \in v$ for lexicon entry (x_j, y_j), and zero otherwise. The information in the matching function was then used along with the information in $d(u_i, v_i)$ to condition the probability of alignments in Equation 3:

$$A_{max} = \arg\max_{A \in \mathcal{A}} \prod_{i=1}^{|A|} \Pr(a_i | d(u_i, v_i); m(u_i, v_i, 1), \ldots, m(u_i, v_i, t)). \tag{6}$$

From this point, Wu proceeded along the lines of Equations 4 and 5 and the dynamic programming solution.

Another interesting approach is possible when part-of-speech taggers are available for both languages. The insight that parts of speech are usually preserved in translation enabled Papageorgiou, Cranias, and Piperidis (1994) to design an alignment algorithm that maximizes the number of matching parts of speech in aligned segments. It is difficult to compare this algorithm's performance to that of other algorithms in the literature, because results were only reported for a relatively easy bitext. On this bitext, the algorithm's performance was nearly perfect. A translation model between parts of speech would not help on bitext regions like the one in Table 1.

The alignment algorithms described above work nearly perfectly given clean bitexts that have easily detectable sentence boundaries. However bitext mapping at the sentence level is not an option for many bitexts (Church 1993). Sentences are often difficult to detect, especially when punctuation is missing due to OCR errors. More importantly, bitexts often contain lists, tables, titles, footnotes, citations and/or markup codes that foil sentence alignment methods. Church's solution was to map bitext correspondence at the level of the smallest text units—characters. Characters match across languages to the extent that they participate in orthographic cognates—words with similar meanings and spellings in different languages. Since there are far more characters than sentences in any bitext, the quadratic computational complexity of this approach presented an efficiency problem. Church showed how to use a high-band filter to find a rough bitext map quickly.

Church's rough bitext maps were intended for input into Dagan, Church, and Gale's (1993) slower algorithm for refinement. Dagan, Church, and Gale used the rough bitext map to define a distance-based model of co-occurrence. Then they adapted Brown et al.'s (1993) statistical translation Model 2 to work with this model of co-occurrence.[2] The information in the translation model was more reliable than character-level cognate information, so it produced a higher signal-to-noise ratio in the bitext space. Therefore, Dagan, Church, and Gale were able to filter out many of the imperfections of the initial bitext map.

A limitation of Church's method, and therefore also of Dagan, Church, and Gale's method, is that orthographic cognates exist only among languages with similar alphabets (Church et al. 1993). Fung investigated ways to make these methods useful when cognates cannot be found. First, working with Church, she introduced the K-Vec algorithm (Fung and Church 1994), which used a rough model of co-occurrence to bootstrap a small translation lexicon. The translation lexicon indicated points of correspondence in the bitext map, much the same way as matching character n-grams. These points of correspondence could then be further refined using the methods previously developed by Church (1993) and Dagan, Church, and Gale (1993). Later, Fung and McKeown (1994) improved on K-Vec by employing relative position offsets, instead of a fixed model of co-occurrence. This strategy made the algorithm more robust for noisier bitexts.

4. The Smooth Injective Map Recognizer (SIMR)

4.1 Overview

SIMR borrows several insights from previous work. Like the algorithms of Gale and Church (1991a) and Brown, Lai, and Mercer (1991), SIMR exploits the cor-

[2] See Melamed (1998a) for a general discussion of models of co-occurrence.

relation between the lengths of mutual translations. Like char_align (Church 1993), SIMR infers bitext maps from likely points of correspondence between the two texts, points that are plotted in a two-dimensional space of possibilities. Unlike previous methods, SIMR greedily searches for only a small chain of correspondence points at a time.

The search begins in a small search rectangle in the bitext space, whose diagonal is parallel to the main diagonal. The search for each chain alternates between a generation phase and a recognition phase. In the generation phase, SIMR generates candidate points of correspondence within the search rectangle that satisfy the supplied matching predicate, as explained in Section 4.2. In the recognition phase, SIMR invokes the chain recognition heuristic to select the most likely chain of *true* points of correspondence (TPCs) among the generated points. The most likely chain of TPCs is the set of points whose geometric arrangement most resembles the typical arrangement of TPCs. The parameters of the chain recognition heuristic are optimized on a small training bitext. If no suitable chains are found, the search rectangle is proportionally expanded by the minimum possible amount and the generation-recognition cycle is repeated. The rectangle keeps expanding until at least one acceptable chain is found. If more than one acceptable chain is found in the same cycle, SIMR accepts the chain whose points are least dispersed around its least-squares line. Each time SIMR accepts a chain, it moves the search rectangle to another region of the bitext space to search for the next chain.

SIMR employs a simple heuristic to select regions of the bitext space to search. To a first approximation, true bitext maps are monotonically increasing functions. This means that if SIMR accepts one chain, it should look for others either above and to the right or below and to the left of the one it has just found. All SIMR needs is a place to start the trace, and a good place to start is at the beginning. Since the origin of the bitext space is always a TPC, the first search rectangle is anchored at the origin. Subsequent search rectangles are anchored at the top right corner of the previously found chain, as shown in Figure 2.

The expanding rectangle search strategy makes SIMR robust in the face of TBM discontinuities. Figure 2 shows a segment of the TBM that contains a vertical gap (an omission in the text on the x-axis). As the search rectangle grows, it will eventually intersect with the TBM, even if the discontinuity is quite large (Melamed 1996a). The noise filter described in Section 4.3 reduces the chances that SIMR will be led astray by false points of correspondence.

4.2 Point Generation

Before SIMR can decide where to generate candidate points of correspondence, it must be told which pairs of words have coordinates within the boundaries of the current search rectangle. The mapping from tokens to axis positions is performed by a language-specific **axis generator** (Melamed 1998b). SIMR calls one of its matching predicates on each pair of tokens whose coordinate falls within the search rectangle. A **matching predicate** is a heuristic for deciding whether two given tokens might be mutual translations. Two kinds of information that a matching predicate can rely on most often are cognates and translation lexicons.

Two words are **orthographic cognates** if they have the same meaning and similar spellings. Similarity of spelling can be measured in more or less complicated ways. The first published attempt to exploit cognates for bitext mapping purposes (Simard, Foster, and Isabelle 1992) deemed two alphabetic tokens cognates if their first four characters were identical. This criterion proved surprisingly effective, given its simplicity. However, like all heuristics, it produced some false positives and some false

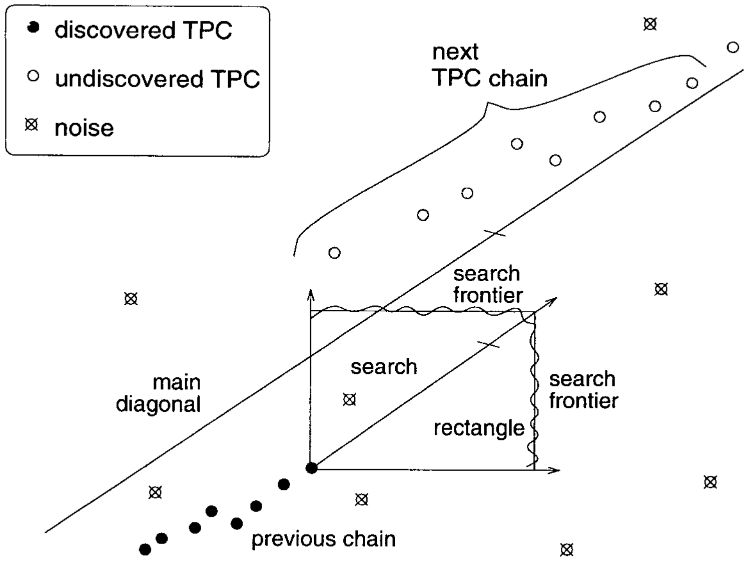

Figure 2
SIMR's "expanding rectangle" search strategy. The search rectangle is anchored at the top right corner of the previously accepted chain. Its diagonal remains parallel to the main diagonal.

negatives. An example of a false negative is the word pair *government* and *gouvernement*. The false positives were often words with a big difference in length, like *conseil* and *conservative*. These examples suggest that a more accurate cognate criterion can be driven by approximate string matching. For example, McEnery and Oakes (1995) threshold the Dice coefficient of matching character bigrams in each pair of candidate cognates. The matching predicates in SIMR's current implementation threshold the Longest Common Subsequence Ratio (LCSR).

The LCSR of two tokens is the ratio of the length of their longest (not necessarily contiguous) common subsequence (LCS) and the length of the longer token. In symbols,

$$LCSR(A,B) = \frac{length[LCS(A,B)]}{max[length(A), length(B)]} . \tag{7}$$

For example, *gouvernement*, which is 12 characters long, has 10 characters that appear in the same order in *government*. So, the LCSR for these two words is 10/12. On the other hand, the LCSR for *conseil* and *conservative* is only 6/12. A simple dynamic programming algorithm (Bellman 1957) can compute the LCS in $O(n^2)$. A rather more complicated algorithm can compute it in $O(n \log\log n)$ time on average (Hunt and Szymanski 1977).

When dealing with language pairs that have different alphabets, the matching predicate can employ **phonetic cognates**. When language L1 borrows a word from language L2, the word is usually written in L1 similarly to the way it sounds in L2. Thus, French and Russian /pɔrtmənə/ are cognates, as are English /sIstəm/ and Japanese /šisutemu/. For many languages, it is not difficult to construct an approximate mapping from the orthography to its underlying phonological form. Given such a mapping for L1 and L2, it is possible to identify cognates despite incomparable orthographies.

Knight and Graehl (1997) have shown that it is possible to find phonetic cognates even between languages whose writing systems are as different as those of English and Japanese. They have built a weighted finite-state automaton (WFSA), based on empirically estimated probability distributions, for back-transliterating English loan words written in katakana into their original English form. The WFSA efficiently represents a large number of transliteration probabilities between words written in the katakana and Roman alphabets. Standard finite-state techniques can efficiently find the most likely path through the WFSA from a Japanese word written in katakana to an English word. The weight of the most likely path is an estimate of the probability that the former is a transliteration of the latter. Thresholding this probability would lead to a phonetic cognate matching predicate for English/Japanese bitexts. The threshold would need to be optimized together with SIMR's other parameters, the same way the LCSR threshold is currently optimized (see Section 5).

Cognates are more common in bitexts from more similar language pairs, and from text genres where more word borrowing occurs, such as technical texts. In the nontechnical Canadian Hansards (parliamentary debate transcripts published in English and in French), an LCSR cutoff of .58 finds cognates for roughly one quarter of all text tokens. Even distantly related languages like English and Czech will share a large number of orthographic cognates in the form of proper nouns, numerals, and punctuation. When one or both of the languages involved is written in pictographs, cognates can still be found among punctuation and numerals. However, these kinds of cognates are usually too sparse to build an accurate bitext map from.

When the matching predicate cannot generate enough candidate correspondence points based on cognates, its signal can be strengthened by a **seed translation lexicon**—a simple list of word pairs that are believed to be mutual translations. Seed translation lexicons can be extracted from machine-readable bilingual dictionaries (MRBDs) in the rare cases where MRBDs are available. In other cases, they can be constructed automatically or semiautomatically using any of several published methods (Fung and Church 1994; Fung 1995; Melamed 1996b; Resnik & Melamed 1997).[3] A matching predicate based on a seed translation lexicon deems two candidate tokens to be mutual translations if the token pair appears in the lexicon. Since the matching predicate need not be perfectly accurate, the seed translation lexicons need not be perfectly accurate either.

All the matching predicates described above can be fine-tuned with stop lists for one or both languages. For example, closed-class words are unlikely to have cognates. Indeed, French/English words like *a, an, on,* and *par* often produce spurious points of correspondence. The same problem is caused by *faux amis* ("false friends") (Macklovitch 1996). These are words with similar spellings but *different* meanings in different languages. For example, the French word *librarie* means 'bookstore,' not 'library,' and *actuel* means 'current,' not 'actual.' A matching predicate can use a list of closed-class words and/or a list of pairs of *faux amis* to filter out spurious matches.

[3] Most published methods for automatically constructing translation lexicons require a preexisting bitext map, which seems to render them useless for the purposes of bitext mapping algorithms. Fortunately, only one seed translation lexicon is required for each language pair, or at worst for each sublanguage. If we expect to map many bitexts in the same language pair, then it becomes feasible to spend a few hours creating one bitext map by hand. Melamed (1996c) explains how to do so quickly and efficiently. Better yet, Fung (1995) shows how it may be possible to extract a small translation lexicon and a rough bitext map simultaneously.

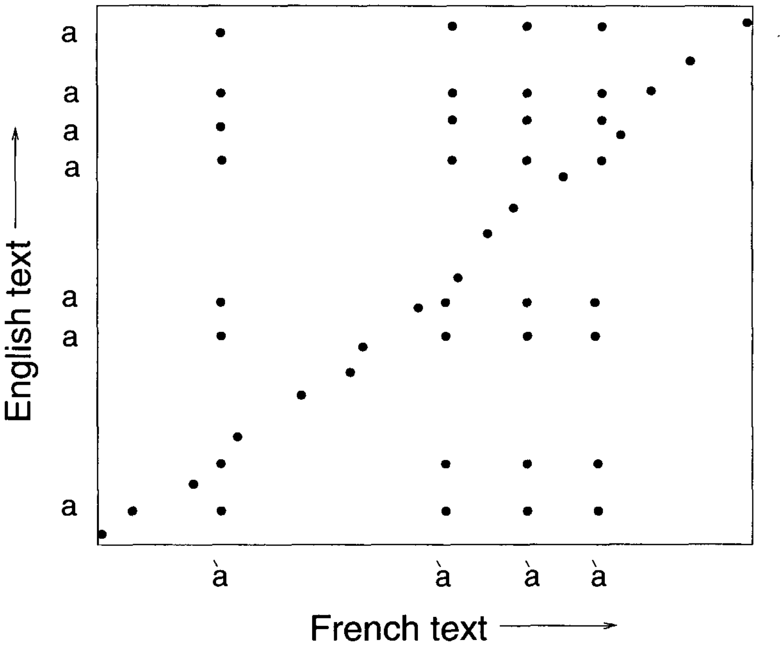

Figure 3
Frequent word types cause false points of correspondence that line up in rows and columns.

4.3 Noise Filter

Inspection of several bitext spaces has revealed a common noise pattern, illustrated in Figure 3. It consists of correspondence points that line up in rows or columns associated with frequent word types. Word types like the English article *a* can produce one or more correspondence points for almost every sentence in the opposite text. Only one point of correspondence in each row and column can be correct; the rest are noise. It is difficult to measure exactly how much noise is generated by frequent tokens, and the proportion is different for every bitext. Informal inspection of some bitext spaces indicated that frequent tokens are often responsible for the lion's share of the noise. Reducing this source of noise makes it much easier for SIMR to stay on track.

Other bitext mapping algorithms mitigate this source of noise either by assigning lower weights to correspondence points associated with frequent word types (Church 1993) or by deleting frequent word types from the bitext altogether (Dagan, Church, and Gale 1993). However, a word type that is relatively frequent overall can be rare in some parts of the text. In those parts, the word type can provide valuable clues to correspondence. On the other hand, many tokens of a relatively rare type can be concentrated in a short segment of the text, resulting in many false correspondence points. The varying concentration of identical tokens suggests that more localized noise filters would be more effective. SIMR's localized search strategy provides a vehicle for a localized noise filter.

The filter is based on the **maximum point ambiguity level** parameter. For each point $p = (x, y)$, let X be the number of points in column x within the search rectangle, and let Y be the number of points in row y within the search rectangle. The ambiguity level of p is defined as X + Y − 2. In particular, if p is the only point in its row and in its column, then its ambiguity level is zero. The chain recognition heuristic ignores

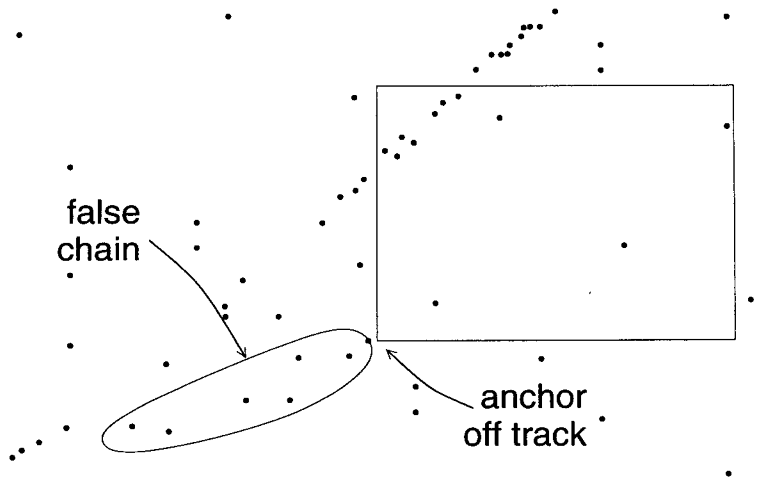

Figure 4
SIMR's noise filter makes an important contribution to the signal-to-noise ratio in the bitext space. Even if one chain of false points of correspondence slips by the chain recognition heuristic, the expanding rectangle is likely to find its way back to the TBM trace before the chain recognition heuristic accepts another chain.

points whose ambiguity level is too high. What makes this a localized filter is that only points within the search rectangle count toward each other's ambiguity level. The ambiguity level of a given point can change when the search rectangle expands or moves.

The noise filter ensures that false points of correspondence are relatively sparse, as illustrated in Figure 4. Even if one chain of false points of correspondence slips by the chain recognition heuristic, the expanding rectangle is likely to find its way back to the TBM trace before the chain recognition heuristic accepts another chain. If the matching predicate generates a reasonably strong signal then the signal-to-noise ratio will be high and SIMR is not likely to get lost, even though it is a greedy algorithm with no ability to look ahead.

4.4 Point Selection

After noise filtering, most TPC chains conform to the pattern illustrated in Figure 5. The pattern can be characterized by three properties:

- **Injectivity:** No two points in a chain of TPCs can have the same x- or y-coordinates.
- **Linearity**: TPCs tend to line up straight. Recall that sets of points with a roughly linear arrangement are called **chains**.
- **Low Variance of Slope**: The slope of a TPC chain is rarely much different from the bitext slope.

SIMR exploits these properties to decide which chains might be TPC chains. First, chains that lack the injectivity property are rejected outright. The remaining chains are filtered using two threshold parameters: **maximum point dispersal** and **maximum angle deviation**. The linearity of each chain is measured as the root mean squared

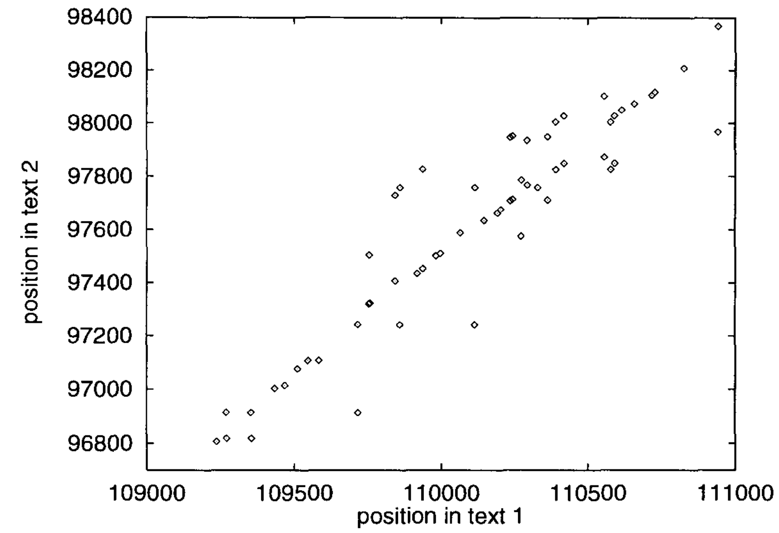

Figure 5
Typical pattern of candidate points of correspondence in a bitext space, after noise filtering. The true points of correspondence trace the true bitext map parallel to the main diagonal.

distance of the chain's points from the chain's least-squares line. If this distance exceeds the maximum point dispersal threshold, the chain is rejected. The angle of each chain's least-squares line is compared to the arctangent of the bitext slope. If the difference exceeds the maximum angle deviation threshold, the chain is rejected.

4.5 Reduction of the Search Space

In a search rectangle containing n points, there are 2^n possible chains—too many to search by brute force. The properties of TPCs listed above provide two ways to constrain the search.

The Linearity property leads to a constraint on the chain size. Chains of only a few points are unreliable, because they often line up straight by coincidence. Chains that are too big will span too long a segment of the TBM to be well approximated by a line. SIMR uses a fixed chain size k, $6 \leq k \leq 11$. The exact value of k is optimized together with the other parameters, as described in Section 5. Fixing the chain size at k reduces the number of candidate chains to $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.

For typical values of n and k, $\binom{n}{k}$ can still reach into the millions. The Low Variance of Slope property suggests another constraint: SIMR should consider only chains that are roughly parallel to the main diagonal. Two lines are parallel if the perpendicular displacement between them is constant. So, chains that are roughly parallel to the main diagonal will consist of points that all have roughly the same displacement from the main diagonal.[4] Points with similar displacement can be grouped together by sorting, as illustrated in Figure 6. Then, chains that are most parallel to the main diagonal will be contiguous subsequences of the sorted point sequence. In a region of the bitext

[4] Displacement can be negative.

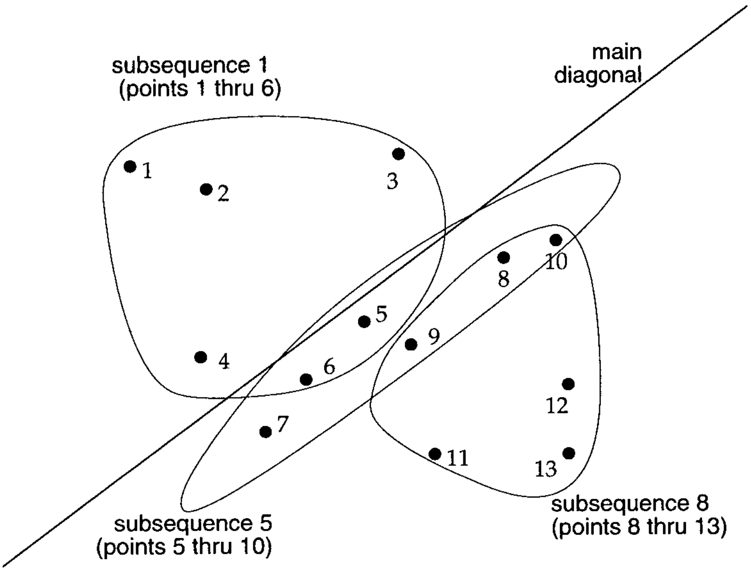

Figure 6
The chain recognition heuristic exploits the Low Variance of Slope property of TPC chains. The candidate points of correspondence are numbered according to their displacement from the main diagonal. The chain most parallel to the main diagonal is always one of the contiguous subsequences of this ordering. For a fixed chain size of 6, there are $13 - 6 + 1 = 8$ contiguous subsequences in this region of 13 points. Of these 8, the fifth subsequence is the best chain.

space containing n points, there will be only $n - k + 1$ such subsequences of length k. The most computationally expensive step in the chain recognition process is the insertion of candidate points into the sorted point sequence.

4.6 Enhancements

The following subsections describe two of the more interesting enhancements in the current SIMR implementation.

4.6.1 Overlapping Chains. SIMR's fixed chain size imposes a rather arbitrary fragmentation on the TBM trace. Each chain starts at the top-right corner of the previously found chain, but these chain boundaries are independent of discontinuities or angle variations in the TBM trace. Therefore, SIMR is likely to miss TPCs wherever the TBM is not linear. One way to make SIMR more robust is to start the search rectangle just above the *lowest* point of the previously found chain, instead of just above the highest point. If the chain size is fixed at k, then each linear stretch of s TPCs will result in $s - k + 1$ overlapping chains.

Unfortunately, this solution introduces another problem: Two overlapping chains can be inconsistent. The injective property of TBMs implies that whenever two (interpolated) chains overlap in the x or y dimensions, but are not identical in the region of overlap, then one of the chains must be wrong. To resolve such conflicts, SIMR employs a postprocessing algorithm to eliminate conflicting chains one at a time, until all remaining chains are pairwise consistent. The conflict resolution algorithm is based on the heuristic that chains that conflict with a larger number of other chains are more likely to be wrong. The algorithm sorts all chains with respect to how many other chains they conflict with, and eliminates them in this sort order, one at a time, until no

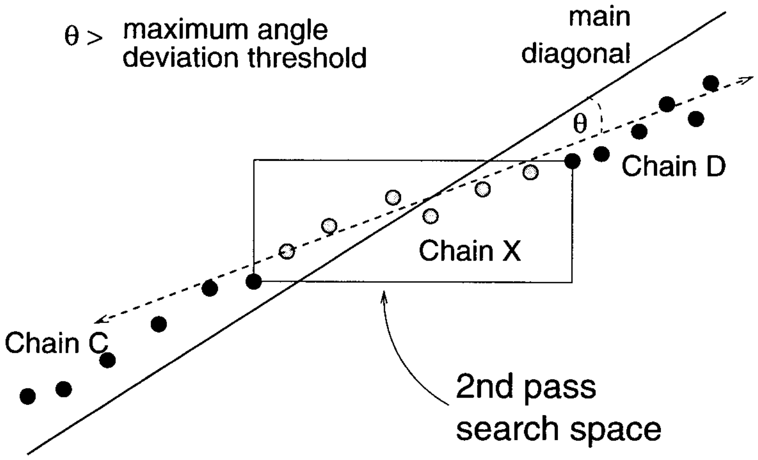

Figure 7
Chain X is perfectly valid, even though it has a highly deviant slope. Such chains can be recovered by re-searching regions between accepted chains. The slope of the *local* main diagonal can be quite different from the slope of the *global* main diagonal.

conflicts remain. Whenever two or more chains are tied in the sort order, the conflict resolution algorithm eliminates all but the chain with the least point dispersal.

4.6.2 Additional Search Passes. To ensure that SIMR rejects spurious chains, the maximum angle deviation threshold must be set low. However, like any heuristic filter, this one will reject some perfectly valid candidates. If a more precise bitext map is desired, some of these valid chains can be recovered during an extra sweep through the bitext space. Since bitext maps are mostly injective, valid chains that are rejected by the angle deviation filter usually occur between two accepted chains, as shown in Figure 7. If Chains C and D are accepted as valid, then the slope of the TBM between the end of Chain C and the start of Chain D must be much closer to the slope of Chain X than to the slope of the main diagonal. Chain X should be accepted. During a second pass through the bitext space, SIMR searches for sandwiched chains in any space between two accepted chains that is large enough to accommodate another chain. This subspace of the bitext space will have its own main diagonal. The slope of this *local* main diagonal can be quite different from the slope of the *global* main diagonal.

An additional search through the bitext space also enables SIMR to recover chains that were missed because of an inversion in the translation. Nonmonotonic TBM segments result in a characteristic map pattern, as a consequence of the injectivity of bitext maps. SIMR has no problem with small nonmonotonic segments inside chains. However, the expanding rectangle search strategy can miss larger nonmonotonic segments that do not fit inside one chain. In Figure 8, the vertical range of segment j corresponds to a vertical gap in SIMR's first-pass map. The horizontal range of segment j corresponds to a horizontal gap in SIMR's first-pass map. Similarly, any nonmonotonic segment of the TBM will occupy the intersection of a vertical gap and a horizontal gap in the monotonic first-pass map. Furthermore, switched segments are usually adjacent and relatively short. Therefore, to recover nonmonotonic segments of the TBM, SIMR needs only to search gap intersections that are close to the first-pass map. There are usually very few such intersections that are large enough to accommodate new chains,

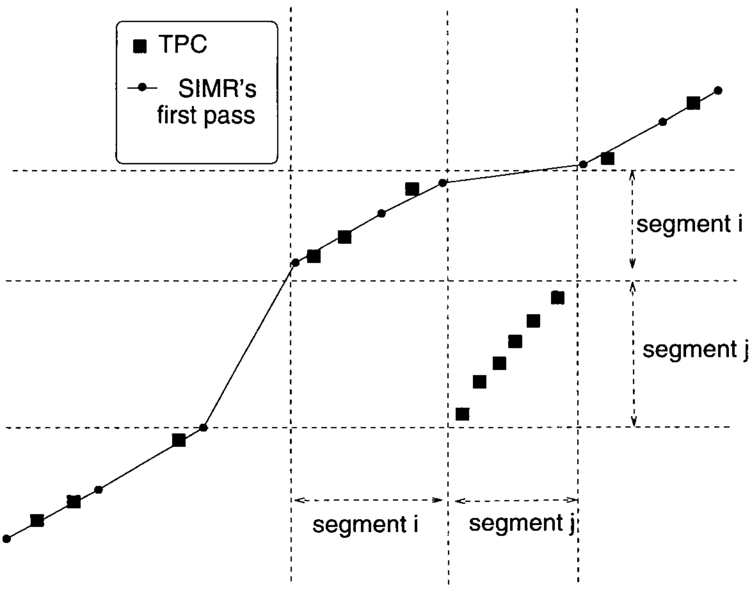

Figure 8
Segments i and j switched places during translation. Any nonmonotonic segment of the TBM will occupy the intersection of a vertical gap and a horizontal gap in the monotonic first-pass map. These larger nonmonotonic segments can be recovered during a second sweep through the bitext space.

so the second-pass search requires only a small fraction of the computational effort of the first pass.

5. Parameter Optimization

SIMR's parameters—the fixed chain size; the LCSR threshold used in the matching predicate; and the thresholds for maximum point dispersal, maximum angle deviation, and maximum point ambiguity—interact in complicated ways. Ideally, SIMR should be reparameterized so that its parameters are pairwise independent. Then it may be possible to optimize the parameters analytically, or at least in a probabilistic framework. For now, the easiest way to optimize these parameters is via simulated annealing (Vidal 1993), a simple general framework for optimizing highly interdependent parameter sets.

Simulated annealing requires an objective function to optimize. The objective function for bitext mapping should measure the difference between the TBM and the interpolated bitext maps produced with the current parameter set. In geometric terms, the difference is a distance. The TBM consists of a set of TPCs. The distance between a bitext map and each TPC can be defined in a number of ways. The simplest metrics are the horizontal distance or the vertical distance, but these metrics measure the error with respect to only one language or the other. A more robust average is the distance perpendicular to the main diagonal. In order to penalize large errors more heavily, root mean squared (RMS) distance, rather than mean distance, should be minimized.

There is a slight complication in the computation of distances between two partial functions, in that linear interpolation is not well-defined for nonmonotonic sets of points. It would be incorrect to simply connect the dots left to right, because the result-

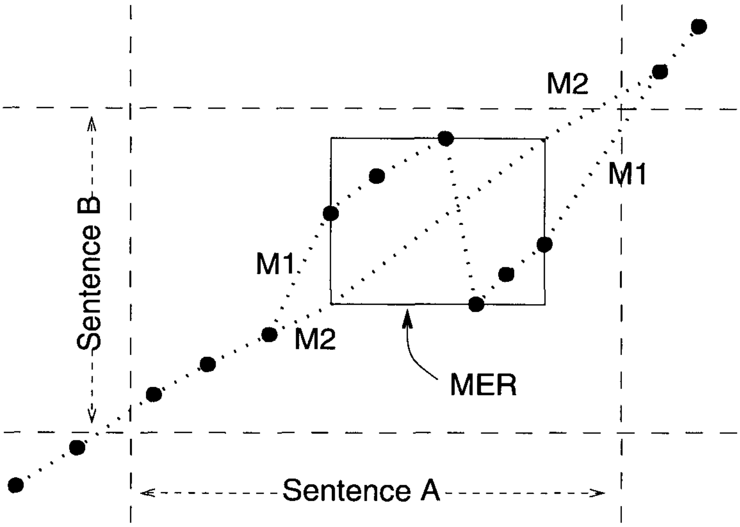

Figure 9
Two text segments at the end of Sentence A were switched during translation, resulting in a nonmonotonic segment. To interpolate injective bitext maps, nonmonotonic segments must be encapsulated in Minimum Enclosing Rectangles (MERs). A unique bitext map can then be interpolated by using the lower left and upper right corners of the MER (map M2), instead of using the nonmonotonic correspondence points (function M1).

Table 2
SIMR accuracy on training bitexts for three language pairs.

Language Pair	Number of Training TPCs	Training Genre	RMS Error in Characters
French / English	598	marketing report	6.6
Spanish / English	562	software manuals	5.5
Korean / English	615	military manuals	3.9

ing function may not be injective. To interpolate injective bitext maps, nonmonotonic segments must be encapsulated in Minimum Enclosing Rectangles (MERs), as shown in Figure 9. A unique bitext map results from interpolating between the lower left and upper right corners of the MER, instead of using the nonmonotonic correspondence points.

6. Evaluation of SIMR

SIMR's parameters were optimized by simulated annealing, as described in the previous section. A separate optimization was performed on separate training bitexts for each of three language pairs. SIMR was then evaluated on previously unseen test bitexts in the three language pairs. The objective function for optimization and the evaluation metric were the root mean squared distance, in characters, between each TPC and the interpolated bitext map produced by SIMR, where the distance was measured perpendicular to the main diagonal. Tables 2 and 3 report SIMR's errors on the training and test bitexts, respectively.

The TBM samples used for training and testing were derived from segment alignments. All the bitexts had been manually aligned by bilingual annotators (Melamed

Table 3
SIMR error estimates on different text genres in three language pairs.

Language Pair	Bitext or Genre	Number of Test TPCs	RMS Error in Characters
French / English	parliamentary debates	7,123	5.7
	CITI technical reports	365, 305, 176	4.4, 2.6, 9.9
	other technical reports	561, 1,393	21, 14
	court transcripts	1,377	3.9
	U.N. annual report	2,049	12
	I.L.O. report	7,129	6.4
Spanish / English	software manuals	376, 151, 100, 349	4.6, 0.67, 5.2, 4.7
Korean / English	military manuals	40, 88, 186, 299	2.6, 7.1, 25, 7.8
	military messages	192	0.53

1997). The alignments were converted into sets of coordinates in the bitext space by pairing the character positions at the ends of aligned segment pairs. This TBM sampling method artificially reduced the error estimates. Most of the aligned segments were sentences, which ended with a period. Whenever SIMR matched the periods correctly, the interpolated bitext map was pulled close to the TPC, even though it may have been much farther off in the middle of the sentence. Thus, the results in Table 3 should be considered only relative to each other and to other results obtained under the same experimental conditions. It would be impressive indeed if any bitext mapping algorithm's actual RMS error were less than 1 character on bitexts involving languages with different word order, such as English/Korean.

The matching predicates for French/English and Spanish/English relied on an LCSR threshold to find cognates. The Korean text contained some Roman character strings, so the matching predicate for Korean/English generated candidate points of correspondence whenever one of these strings coordinated in the search rectangle with an identical string in the English half of the bitext. A seed translation lexicon was also used to strengthen the Korean/English signal. In addition, English, French, Spanish and Korean stop lists were used to prevent matches of closed-class words. The translation lexicon and stop lists had been previously developed independently of the training and test bitexts.

The French/English part of the evaluation was performed on bitexts from the publicly available *corpus de bi-texte anglais-français (BAF)* (Simard and Plamondon 1996). SIMR's error distribution on the "parliamentary debates" bitext in this collection is given in Table 4. This distribution can be compared to the error distributions reported for the same test set by Dagan, Church, and Gale (1993), who reported parts of their error distribution in words, rather than in characters: "In 55% of the cases, there is no error in `word_align`'s output (distance of 0), in 73% the distance from the correct alignment is at most 1, and in 84% the distance is at most 3" (Dagan, Church, and Gale 1993, 7). These distances were measured horizontally from the bitext map rather than perpendicularly to the main diagonal. Given the bitext slope for that bitext and a conservative estimate of 6 characters per word (including the space between words), each horizontal word of error corresponds to just over 4 characters of error perpendicular to the main diagonal. Thus, Dagan, Church, and Gale's "no error" is the same as

Table 4
SIMR's error distribution on the French/English "parliamentary debates" bitext. Errors were measured perpendicular to the main diagonal.

Number of Test Points	Error Range in Characters	Fraction of Test Points
1	-101	.0001
2	-80 to -70	.0003
1	-70 to -60	.0001
5	-60 to -50	.0007
4	-50 to -40	.0006
6	-40 to -30	.0008
9	-30 to -20	.0013
29	-20 to -10	.0041
3,057	-10 to 0	.4292
3,902	0 to 10	.5478
43	10 to 20	.0060
28	20 to 30	.0039
17	30 to 40	.0024
5	40 to 50	.0007
8	50 to 60	.0011
1	60 to 70	.0001
1	70 to 80	.0001
1	80 to 90	.0001
1	90 to 100	.0001
1	110 to 120	.0001
1	185	.0001
7,123	-101 to 185	1.000

Table 5
Comparison of error distributions for SIMR and `word_align` on the parliamentary debates bitext.

Algorithm	Error of at Most 2 Characters	Error of at Most 6 Characters	Error of at Most 14 Characters
`word_align`	55%	73%	84%
SIMR	93%	97%	98%

2 characters of error or less, i.e., less than half a word. One word of error is the same as an error of up to 6 characters and 3 words are equivalent to $4 \cdot 3\frac{1}{2} = 14$ characters. On this basis, Table 5 compares the accuracy of SIMR and `word_align`.[5]

Another interesting comparison is in terms of maximum error. Certain applications of bitext maps, such as the one described by Melamed (1996a), can tolerate many small errors but no large ones. As shown in Table 4, SIMR's bitext map was never off by more than 185 characters from any of the 7,123 segment boundaries. 185 characters is about 1.5 times the length of an average sentence (Melamed 1996a). The input to `word_align` is the output of `char_align` and Dagan, Church, and Gale (1993) have reported that `word_align` cannot escape from `char_align`'s worst errors. An independent implementation of `char_align` (Michel Simard, personal communication) erred by more than one thousand characters on the same bitext.

[5] Error measurements at the character level are less susceptible to random variation than measurements at the word level. Character-level measurements also have the advantage of being universally applicable to all languages, including those in which words are difficult to identify automatically.

The Spanish/English and Korean/English bitexts were hand-aligned when SIMR was being ported to these language pairs.[6] The Spanish/English bitexts were drawn from the Sun Solaris AnswerBooks and hand-aligned by Philip Resnik. The Korean/ English bitexts were provided by MIT's Lincoln Laboratories and hand-aligned by Young-Suk Lee. Table 3 shows that SIMR's performance on Spanish/English and Korean/English bitexts is no worse than its performance on French/English bitexts.

The results in Table 3 were obtained using a version of SIMR that included all the enhancements described in Section 4.6. It is interesting to consider the degree to which each enhancement improves performance. I remapped the French/English bitexts listed in Table 3 with two stripped-down versions of SIMR. One version was basic SIMR without any enhancements. The other version incorporated overlapping chains, but performed only one search pass. The deterioration in performance varied widely. For example, on the parliamentary debates bitext, the RMS error rose from 5.7 to 16 when only one search pass was allowed, but rose only another 2 points to 18 using non-overlapping chains. In contrast, on the U.N. annual report bitext, the extra search passes made no difference at all but non-overlapping chains increased the RMS error from 12 to 40. For most of the other bitexts, each enhancement reduced the RMS error by a few characters, compared to the basic version. However, the improvement was not universal: the RMS error of the basic SIMR was 19 for the "other technical report" on which the enhanced SIMR scored 21. The expected value of the enhancements is difficult to predict, because each enhancement is aimed at solving a particular pattern recognition problem, and each problem may or may not occur in a given bitext. The relationship between geometric patterns in TPC chains and syntactic properties of bitexts is a ripe research topic.

7. Alignment

SIMR has no idea that words are often used to make sentences. It just outputs a series of corresponding token positions, leaving users free to draw their own conclusions about how the texts' larger units correspond. However, many existing translators' tools and machine translation strategies depend on aligned sentences or other aligned text segments. What can SIMR do for them? Formally, an alignment is a correspondence relation that does not permit crossing correspondences. The rest of this article presents the Geometric Segment Alignment (GSA) algorithm, which uses segment boundary information to reduce the correspondence relation in SIMR's output to a segment alignment. The GSA algorithm can be applied equally well to sentences, paragraphs, lists of items, or any other text units for which boundary information is available.

7.1 Correspondence is Richer than Alignment

A set of correspondence points, supplemented with segment boundary information, expresses **segment correspondence**, which is a richer representation than segment alignment. Figure 10 illustrates how segment boundaries form a grid over the bitext space. Each cell in the grid represents the intersection of two segments, one from each half of the bitext. A point of correspondence inside cell (X,y) indicates that some token in segment X corresponds with some token in segment y; i.e., segments X and y correspond. For example, Figure 10 indicates that segment e corresponds with segments G and H.

In contrast to a correspondence relation, "an **alignment** is a segmentation of the

[6] The porting method is detailed elsewhere (Melamed 1996c, 1997, 1998b).

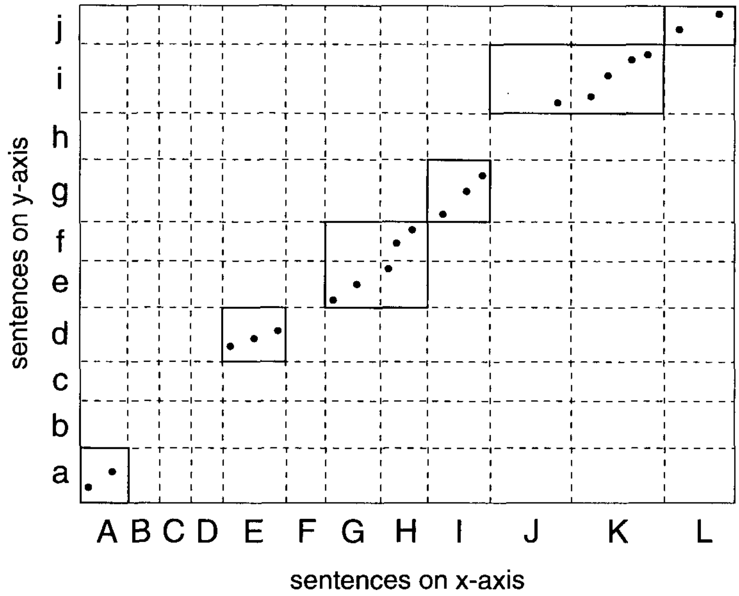

Figure 10
Segment boundaries form a grid over the bitext space. Each cell in the grid represents the product of two segments, one from each half of the bitext. A point of correspondence inside cell (X,y) indicates that some token in segment X corresponds with some token in segment y; i.e., the segments X and y correspond. So, for example, segment E corresponds with segment d. The aligned blocks are outlined with solid lines.

two texts such that the *n*th segment of one text is the translation of the *n*th segment of the other" (Simard, Foster, and Isabelle 1992, 68). For example, given the token correspondences in Figure 10, the segment ⟨G,H⟩ should be aligned with the segment ⟨e,f⟩. If segments $\langle X_1, \ldots, X_n \rangle$ align with segments $\langle y_1, \ldots, y_n \rangle$, then $(\langle X_1, \ldots, X_n \rangle, \langle y_1, \ldots, y_n \rangle)$ is an **aligned block**. In geometric terms, aligned blocks are rectangular regions of the bitext space, such that the sides of the rectangles coincide with segment boundaries, and such that no two rectangles overlap either vertically or horizontally. The aligned blocks in Figure 10 are outlined with solid lines.

SIMR's initial output has more expressive power than the alignment that can be derived from it. One illustration of this difference is that segment correspondence can represent order inversions, but segment alignment cannot. Inversions occur surprisingly often in real bitexts, even for sentence-size segments (Church 1993). Figure 10 provides another illustration. If, instead of the point in cell (H,e), there was a point in cell (G,f), the correct alignment for that region would still be (⟨G,H⟩, ⟨e,f⟩). If there were points of correspondence in both (H,e) and (G,f), the correct alignment would still be the same. Yet, the three cases are clearly different. If a lexicographer wanted to see a word in segment G in its bilingual context, it would be useful to know whether segment f is relevant.

7.2 The Geometric Segment Alignment (GSA) Algorithm

Given a sequence of segment boundaries for each half of a bitext, the **Geometric Segment Alignment (GSA)** algorithm reduces sets of correspondence points to segment alignments. The algorithm's first step is to perform a transitive closure over the input correspondence relation. For instance, if the input contains (G,e), (H,e), and (H,f), then GSA adds the pairing (G,f). Next, GSA forces all segments to be contiguous: If segment Y corresponds with segments x and z, but not y, the pairing (Y,y) is added. In geometric terms, these two operations arrange all cells that contain points of correspondence into nonoverlapping rectangles, while adding as few cells as possible. The result is an alignment relation.

A complete set of TPCs, together with appropriate boundary information, guarantees a perfect alignment. Alas, the points of correspondence postulated by SIMR are neither complete nor noise-free. SIMR makes errors of omission and errors of commission. Fortunately, the noise in SIMR's output causes alignment errors in predictable ways. GSA employs several backing-off heuristics to reduce the number of errors.

Typical errors of commission are stray points of correspondence like the one in cell (H,e) in Figure 10. This point indicates that ⟨G,H⟩ and ⟨e,f⟩ should form a 2×2 aligned block, whereas the lengths of the component segments suggest that a pair of 1×1 blocks is more likely. In a separate development bitext, I have found that SIMR is usually wrong in these cases. To reduce such errors, GSA asks Gale & Church's length-based alignment algorithm (Gale and Church 1991a; Michel Simard, personal communication) for a second opinion on any aligned block that is not 1×1. Whenever the length-based algorithm prefers a more fine-grained alignment, its judgement overrules SIMR's.

Typical errors of omission are illustrated in Figure 10 by the complete absence of correspondence points between segments ⟨B,C,D⟩ and ⟨b,c⟩. This empty block of segments is sandwiched between aligned blocks. It is highly likely that at least some of these segments are mutual translations, despite SIMR's failure to find any points of correspondence between them. Therefore, GSA treats all sandwiched empty blocks as aligned blocks. If an empty block is not 1×1, GSA realigns it using Gale and Church's length-based algorithm, just as it would realign any other many-to-many aligned block.

The most problematic cases involve an error of omission adjacent to an error of commission, as in blocks (⟨⟩, ⟨h⟩) and (⟨J,K⟩, ⟨i⟩). If the point in cell (J,i) should really be in cell (J,h), then realignment inside the erroneous blocks would not solve the problem. A naive solution is to merge these blocks and then to realign them using a length-based method. Unfortunately, this kind of alignment pattern, i.e., 0×1 followed by 2×1, is surprisingly often correct. Length-based methods assign low probabilities to such pattern sequences and usually get them wrong. Therefore, GSA also considers the confidence level with which the length-based alignment algorithm reports its realignment. If this confidence level is sufficiently high, GSA accepts the length-based realignment; otherwise, the alignment indicated by SIMR's points of correspondence is retained. The minimum confidence at which GSA trusts the length-based realignment is a GSA parameter, which has been optimized on a separate development bitext.

8. Evaluation of GSA

GSA processed two bitext maps produced by SIMR using two different matching predicates. The first matching predicate relied only on cognates that pass a certain LCSR threshold, as described in Section 4.2. The second matching predicate was like the first, except that it also generated a point of correspondence whenever the input token pair appeared as an entry in a translation lexicon. The translation lexicon was automatically extracted from an MRBD (Cousin et al. 1991).

Bitexts involving millions of segments are becoming more and more common. Before comparing bitext alignment algorithms in terms of accuracy, it is important

Table 6
Comparison of bitext alignment algorithms' accuracy. One error is counted for each aligned block in the reference alignment that is missing from the test alignment.

Bitext	Algorithm	Errors, Given Aligned Paragraphs	%	Errors, Not Given Aligned Paragraphs	%
"easy"	Gale and Church (1991a)	not available		128	1.8
Hansard	Simard, Foster, and Isabelle (1992)	114	1.6	171	2.4
(n = 7,123)	SIMR/GSA	104	1.5	115	1.6
	SIMR/GSA with MRBD	80	1.1	90	1.3
"hard"	Gale and Church (1991a)	not available		80	3.0
Hansard	Simard, Foster, and Isabelle (1992)	50	1.9	102	3.8
(n = 2,693)	SIMR/GSA	50	1.9	61	2.3
	SIMR/GSA with MRBD	45	1.7	48	1.8

to compare their asymptotic running times. In order to run a quadratic-time alignment algorithm in a reasonable amount of time on a large bitext, the bitext must be presegmented into a set of smaller bitexts. When a bitext contains no easily recognizable "anchors," such as paragraphs or sections, this first-pass alignment must be done manually.

Given a reasonably good bitext map, GSA's expected running time is linear in the number of input segment boundaries. In all the bitexts on which GSA was trained and tested, the points of correspondence in SIMR's output were sufficiently dense and accurate that GSA backed off to a quadratic-time alignment algorithm only for very small aligned blocks. For example, when the seed translation lexicon was used in SIMR's matching predicate, the largest aligned block that needed to be realigned was 5×5 segments. Without the seed translation lexicon, the largest realigned block was 7×7 segments. Thus, GSA can obviate the need to manually prealign large bitexts.

Table 6 compares GSA's accuracy on the "easy" and "hard" French/English bitexts with the accuracy of two other alignment algorithms, as reported by Simard, Foster, and Isabelle (1992). The error metric counts one error for each aligned block in the reference alignment that is missing from the test alignment. To account for the possibility of modularizing the overall alignment task into paragraph alignment followed by sentence alignment, Simard, Foster, and Isabelle (1992) have reported the accuracy of their sentence alignment algorithm when a perfect alignment at the paragraph level is given. SIMR/GSA was also tested in this manner, to enable the second set of comparisons in Table 6.

Due to the scarcity of hand-aligned training bitexts at my disposal, GSA's backing-off heuristics are somewhat ad hoc. Even so, GSA performs at least as well as, and usually better than, other alignment algorithms for which comparable results have been published. Chen (1996) has also published a quantitative evaluation of his alignment algorithm on these reference bitexts, but his evaluation was done post hoc. Since the results in this article are based on a gold standard, they are not comparable to Chen's results. Among other reasons, error rates based on a gold standard are sometimes inflated by errors in the gold standard and this was indeed the case for the gold standard used here (see Melamed [1996a]). It is also an open question whether GSA performs better than the algorithm proposed by Wu (1994). The two algorithms have not yet been evaluated on the same test data. For now, I can offer only a theoretical reason why SIMR+GSA should be more accurate than the algorithms of Chen and Wu: Bitext maps lead to alignment more directly than a translation model (Chen 1996) or a translation lexicon (Wu 1994), because both segment alignments and bitext maps are relations between token instances, rather than between token types.

More important than GSA's current accuracy is GSA's potential accuracy. With a bigger development bitext, more effective backing-off heuristics can be developed. Better input can also make a difference: GSA's accuracy will improve in lockstep with SIMR's accuracy.

9. Conclusion

The Smooth Injective Map Recognizer (SIMR) is based on innovative approaches to each of the three main components of a bitext mapping algorithm: signal generation, noise filtering, and search. The advances in signal generation stemmed from the use of word-based matching predicates. When word-pair coordinates are plotted in a Cartesian bitext space, the geometric heuristics of existing sentence alignment algorithms can be exploited just as easily and to a greater extent at the word level. The cognate heuristic of character-based bitext mapping algorithms also works better at the word level, because cognateness can be defined more precisely in terms of words, e.g., using the Longest Common Subsequence Ratio. Most importantly, matching heuristics based on existing translation lexicons can be defined only at the word level. When neither cognates nor sentence boundaries can be found, we can still map bitexts in any pair of languages using a small hand-constructed translation lexicon. To complement word-based matching predicates, I have proposed localized noise filtering. Localized noise filters are more accurate than global ones because they are sensitive to local variations in noise distributions. The combination of a strong signal and an accurate noise filter enables localized search heuristics. Localized search heuristics can directly exploit the geometric tendencies of TPC chains in order to search the bitext space in linear space and time. This level of efficiency is particularly important for large bitexts.

SIMR also advances the state of the art of bitext mapping on several other criteria. Evaluation on preexisting gold standards has shown that SIMR can map bitexts with high accuracy in a variety of language pairs and text genres, without getting lost. SIMR is robust in the face of translation irregularities like omissions and allows crossing correspondences to account for word-order differences. SIMR encapsulates its language-specific heuristics, so that it can be ported to any language pair with a minimal effort (Melamed 1997). These features make SIMR one of the most widely applicable bitext mapping algorithms published to date.

For applications that require it, SIMR's bitext maps can be efficiently reduced to segment alignments, using the Geometric Segment Alignment (GSA) algorithm presented here. Admittedly, GSA is only useful when a good bitext map is available. In such cases, there are three reasons to favor GSA over other options for alignment: One, it is simply more accurate. Two, its expected running time is linear in the size of the bitext. Therefore, three, it is not necessary to manually prealign large bitexts before input to GSA.

There are numerous ways to improve on the methods presented here. If SIMR can be reparameterized so that its parameters are pairwise independent, then it may be possible to optimize these parameters analytically, or at least within a well-founded probabilistic framework. Likewise, the parameters in GSA's backing-off heuristics and the heuristics themselves were partially dictated by the scarcity of suitable training data at the time that GSA was being developed. All of this is to say that the details of the current implementations of SIMR and GSA are less important than the general approach to bitext mapping advocated here.

Acknowledgments

This research began while I was a visitor at the Centre d'Innovation en Technologies de l'Information in Laval, Canada. I am indebted to Pierre Isabelle for informing me that the bitext mapping problem was far from being solved. Thanks are due to everyone at CITI for letting me use their software. SIMR was ported to Spanish/English while I was visiting Sun MicroSystems Laboratories. Thanks to Gary Adams, Cookie Callahan, Bob Kuhns, and Philip Resnik for their help with that project. Thanks also to Philip Resnik for writing the Spanish tokenizer, and for hand-aligning the Spanish/English training bitexts. Porting SIMR to Korean/English would not have been possible without Young-Suk Lee of MIT's Lincoln Laboratories, who provided the seed translation lexicon, and aligned all the training and test bitexts. This paper has benefited tremendously from the insights and comments of Stan Chen, Ken Church, Mike Collins, Ido Dagan, Jason Eisner, George Foster, Pierre Isabelle, Elliott Macklovitch, Mitch Marcus, Adwait Ratnaparkhi, Michel Simard, Eero Simoncelli, Matthew Stone, Lyle Ungar, Bonnie Webber, and four anonymous reviewers. The majority of this work was done at the Department of Computer and Information Science of the University of Pennsylvania, where it was supported by an equipment grant from Sun MicroSystems and partially funded by ARO grant DAAL03-89-C0031 PRIME and by ARPA grants N00014-90-J-1863 and N66001-94C-6043.

References

Bellman, Richard. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Brown, Peter F., Stephen Della Pietra, Vincent Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics, 19*(2): 263–311.

Brown, Peter F., Jennifer C. Lai, and Robert L. Mercer. 1991. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting*, pages 169–176, Berkeley, CA. Association for Computational Linguistics.

Catizone, Roberta, Graham Russell, and Susan Warwick. 1993. Deriving translation data from bilingual texts. In *Proceedings of the First International Lexical Acquisition Workshop*, Detroit, MI.

Chen, Stanley. 1996. *Building Probabilistic Models for Natural Language*. Ph.D. dissertation, Harvard University, Cambridge, MA.

Church, Kenneth W. 1993. Char_align: A program for aligning parallel texts at the character level. In *Proceedings of the 31st Annual Meeting*, pages 1–8, Columbus, OH. Association for Computational Linguistics.

Church, Kenneth W., Ido Dagan, William Gale, Pascale Fung, J. Helfman, and B. Satish. 1993. Aligning parallel texts: Do methods developed for English-French generalize to Asian languages? In *Proceedings of PacfoCol'93*. Taipei, Taiwan.

Cousin, Pierre-Henri, Lorna Sinclair, Jean-François Allain, and Catherine E. Love. 1991. *The Collins Paperback French Dictionary*. Harper Collins Publishers, Glasgow, Scotland.

Dagan, Ido, Kenneth W. Church, and William Gale. 1993. Robust word alignment for machine aided translation. In *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives*, pages 1–8, Columbus, OH.

Debili, Fathi and Elyès Sammouda. 1992. Appariement des phrases de textes bilingues. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 517–538, Nantes, France.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, 34(B)*: 1–38.

Fung, Pascale. 1995. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *Proceedings of the 33rd Annual Meeting*, pages 236–243, Boston, MA. Association for Computational Linguistics.

Fung, Pascale and Kenneth W. Church. 1994. K-vec: A new approach for aligning parallel texts. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 1,096–1,102, Kyoto, Japan.

Fung, Pascale and Kathleen McKeown. 1994. Aligning noisy parallel corpora across language groups: Word pair feature matching by dynamic time warping. In *Proceedings of the Conference of the Association for Machine Translation in the*

Americas, pages 81–88, Columbia, MD.

Gale, William and Kenneth W. Church. 1991a. A program for aligning sentences in bilingual corpora. In *Proceedings of the 29th Annual Meeting*, pages 177–184, Berkeley, CA. Association for Computational Linguistics.

Gale, William and Kenneth W. Church. 1991b. Identifying word correspondences in parallel texts. In *Proceedings of the DARPA SNL Workshop*, pages 152–157.

Harris, Brian. 1988. Bi-text, a new concept in translation theory. *Language Monthly*, 54: 8–10.

J. W. Hunt and T. G. Szymanski. 1977. A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20(5): 350–353.

Isabelle, Pierre. 1992. Bi-textual aids for translators. In *Proceedings of the 8th Annual Conference of the UW Centre for the New OED and Text Research*, pages 1–15, Waterloo, Canada.

Kay, Martin and Martin Röscheisen. 1993. Text-translation alignment. *Computational Linguistics*, 19(1): 121–142.

Knight, Kevin and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the 35th Annual Meeting*, pages 128–135, Madrid, Spain. Association for Computational Linguistics.

Macklovitch, Elliott. 1996. Peut-on vérifier automatiquement la cohérence terminologique? *META* 41(3).

McEnery, Tony and Michael Oakes. 1995. Cognate extraction in the CRATER project: Methods and assessment. In *Proceedings of From Texts to Tags: Issues in Multilingual Language Analysis*, pages 77–86, Dublin, Ireland.

Melamed, I. Dan. 1996a. Automatic detection of omissions in translations. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 764–769, Copenhagen, Denmark.

Melamed, I. Dan. 1996b. Automatic construction of clean broad-coverage translation lexicons. In *Proceedings of the 2nd Conference of the Association for Machine Translation in the Americas*, pages 125–134, Montreal, Canada.

Melamed, I. Dan. 1996c. Porting SIMR to new language pairs. Institute for Research in Cognitive Science Technical Report 96-26, University of Pennsylvania, Philadelphia, PA.

Melamed, I. Dan. 1997. A portable algorithm for mapping bitext correspondence. In *Proceedings of the 35th Annual Meeting*, pages 305–312, Madrid, Spain. Association for Computational Linguistics.

Melamed, I. Dan. 1998a. Models of co-occurrence. Institute for Research in Cognitive Science Technical Report 98-05, University of Pennsylvania, Philadelphia, PA.

Melamed, I. Dan. 1998b. *Empirical Methods for Exploiting Parallel Texts*. Ph.D. dissertation, University of Pennsylvania, Philadelphia, PA.

Nerbonne, John, Lauri Karttunen, Elena Paskaleva, Gabor Proszeky, and Tiit Roosmaa. 1997. Reading more into foreign languages. In *Proceedings of the 5th ACL Conference on Applied Natural Language Processing*, pages 135–138, Washington, DC.

Papageorgiou, Harris, Lambros Cranias, and Stelios Piperidis. 1994. Automatic alignment in parallel corpora. In *Proceedings of the 32nd Annual Meeting (Student Session)*, pages 334–336, Las Cruces, NM. Association for Computational Linguistics.

Resnik, Philip and I. Dan Melamed. 1997. Semi-automatic acquisition of domain-specific translation lexicons. In *Proceedings of the 5th ACL Conference on Applied Natural Language Processing*, pages 340–347, Washington, DC.

Simard, Michel, George F. Foster, and Pierre Isabelle. 1992. Using cognates to align sentences in bilingual corpora. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 67–81. Montreal, Canada.

Simard, Michel and Pierre Plamondon. 1996. Bilingual sentence alignment: Balancing robustness and accuracy. In *Proceedings of the 2nd Conference of the Association for Machine Translation in the Americas*, pages 135–144, Montreal, Canada.

René V. V. Vidal, editor. 1993. *Applied Simulated Annealing*. Springer-Verlag, Heidelberg, Germany.

Wu, Dekai. 1994. Aligning a parallel English-Chinese corpus statistically with lexical criteria. In *Proceedings of the 32nd Annual Meeting*, pages 80–87, Las Cruces, NM. Association for Computational Linguistics.