The Effects of Word Order and Segmentation on Translation Retrieval Performance

Timothy Baldwin and **Hozumi Tanaka**
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552 JAPAN
{tim,tanaka}@cl.cs.titech.ac.jp

Abstract

This research looks at the effects of word order and segmentation on translation retrieval performance for an experimental Japanese-English translation memory system. We implement a number of both bag-of-words and word order-sensitive similarity metrics, and test each over character-based and word-based indexing. The translation retrieval performance of each system configuration is evaluated empirically through the notion of word edit distance between translation candidate outputs and the model translation. Our results indicate that character-based indexing is consistently superior to word-based indexing, suggesting that segmentation is an unnecessary luxury in the given domain. Word order-sensitive approaches are demonstrated to generally outperform bag-of-words methods, with source language segment-level edit distance proving the most effective similarity metric.

1 Introduction

Translation memories (TM's) are a well-established technology within the human and machine translation fraternities, due to the high translation precision they afford. Essentially, TM's are a list of **translation records** (source language strings paired with a unique target language translation), which the TM system accesses in suggesting a list of target language **translation candidates** which may be helpful to the translator in translating a given source language input.[1]

Naturally, TM systems have no way of accessing the target language equivalent of the source language input, and hence the list of *target language* translation candidates is determined based on *source language* similarity between the current input and translation examples within the TM, with translation equivalent(s) of maximally similar source language string(s) given as the translation candidate(s). This is based on the assumption that structural and semantic similarities between target language translations will be reflected in the original source language equivalents.

One reason for the popularity of TM's is the low operational burden they pose to the user, in that translation pairs are largely acquired automatically from observation of the incremental translation process, and translation candidates can be produced on demand almost instantaneously. To support this low overhead, TM systems must allow fast access into the potentially large-scale TM, but at the same time be able to predict translation similarity with high accuracy. Here, there is clearly a trade-off between **access/retrieval speed** and **predictive accuracy** of the retrieval mechanism. Traditionally, research on TM retrieval methods has focused on speed, with little cross-evaluation of the accuracy of different methods. We prefer to focus on accuracy, and present empirical data evidencing the relative predictive potential of different similarity metrics over different parameterisations.

In this paper, we focus on comparison of different retrieval algorithms for non-segmenting languages, based around a TM system from Japanese to English. Non-segmenting languages are those which do not involve delimiters (e.g. spaces) between words, and include Japanese, Chinese and Thai. We are particularly interested in the part the orthogonal parameters of segmentation and word order play in the speed/accuracy trade-off. That is, by doing away with segmentation in relying solely on character-level comparison (**character-based indexing**), do we significantly degrade match performance, as compared to word-level comparison (**word-based indexing**)? Similarly, by ignoring word order and treating each source language string as a "bag of words", do we genuinely lose out over word order-sensitive approaches? The main objective of this research is thus to determine whether the computational overhead associated with more stringent approaches (i.e. word-based indexing and word order-sensitive approaches) is commensurate with the performance gains they offer.

To preempt what follows, the major contributions of this research are: (a) empirical evaluation of different comparison methods over actual Japanese-English TM data, focusing on four orthogonal retrieval paradigms; (b) the finding that, over the target data, character-based indexing is consistently superior to word-based indexing in identifying the translation candidate most similar to the optimal translation for a given input; and (c) empirical verification of the supremacy of word order-sensitive exhaustive string comparison methods over boolean match methods.

In the following sections we discuss the effects

[1] See Planas (1998) for a thorough review of commercial TM systems.

of segmentation and word order (§ 2) and present a number of both bag-of-words and word order-sensitive similarity metrics (§ 3), before going on to evaluate the different methods with character-based and word-based indexing (§ 4). We then conclude the paper in Section 5.

2 Segmentation and word order

Using **segmentation** to divide strings into component words or morphemes has the obvious advantage of clustering characters into semantic units, which in the case of ideogram-based languages such as Japanese (in the form of kanji characters) and Chinese, generally disambiguates character meaning. The kanji character '見', for example, can be used to mean any of "to discern/discriminate", "to speak/argue" and "a valve", but word context easily resolves such ambiguity. In this sense, our intuition is that segmented strings should produce better results than non-segmented strings.

Looking to past research on similarity metrics for TM systems, almost all systems involving Japanese as the source language rely on segmentation (e.g. (Nakamura, 1989; Sumita and Tsutsumi, 1991; Kitamura and Yamamoto, 1996; Tanaka, 1997)), with Sato (1992) and Sato and Kawase (1994) providing rare instances of character-based systems.

By avoiding the need to segment text, we: (a) alleviate computational overhead; (b) avoid the need to commit ourselves to a particular analysis type in the case of ambiguity; (c) avoid the issue of how to deal with unknown words; (d) avoid the need for stemming/lemmatisation; and (e) to a large extent get around problems related to the normalisation of lexical alternation (see Baldwin and Tanaka (1999) for a discussion of problems related to lexical alternation in Japanese). Additionally, we can use the commonly ambiguous nature of individual kanji characters to our advantage, in modelling semantic similarity between related words with character overlap. With word-based indexing, this would only be possible with the aid of a thesaurus.

Similarly for **word order**, we would expect that translation records that preserve the word (segment) order observed in the input string would provide closer-matching translations than translation records containing those same segments in a different order. Naturally, enforcing preservation of word order is going to place a significant burden on the matching mechanism, in that a number of different substring match schemata are inevitably going to be produced between any two strings, each of which must be considered on its own merits.

To the authors' knowledge, there is no TM system operating from Japanese that does not rely on word/segment/character order to some degree. Tanaka (1997) uses pivotal content words identified by the user to search through the TM and locate translation records which contain those same content words in the same order and preferably the same segment distance apart. Nakamura (1989) similarly gives preference to translation records in which the content words contained in the original input occur in the same linear order, although there is the scope to back off to translation records which do not preserve the original word order. Sumita and Tsutsumi (1991) take the opposite tack in iteratively filtering out NPs and adverbs to leave only functional words and matrix-level predicates, and find translation records which contain those same key words in the same ordering, preferably with the same segment types between them in the same numbers. Nirenburg et al. (1993) propose a word order-sensitive metric based on "string composition discrepancy", and incrementally relax the restriction on the quality of match required to include word lemmata, word synonyms and then word hypernyms, increasing the match penalty as they go. Sato and Kawase (1994) employ a more local model of *character* order in modelling similarity according to N-grams fashioned from the original string.

The greatest advantage in ignoring word/segment order is computational, in that we significantly reduce the search space and require only a single overall comparison per string pair. Below, we analyse whether this gain in speed outweighs any losses in retrieval performance.

3 Similarity metrics

Due to our interest in the effects of both word order and segmentation, we must have a selection of similarity metrics compatible with the various permutations of these two parameter types. We choose to look at a number of bag-of-words and word order-sensitive methods which are compatible with both character-based and word-based indexing, and vary the input to model the effects of the two indexing paradigms. The particular bag-of-word approaches we target are the vector space model (Manning and Schütze, 1999, p300) and "token intersection", a simple ratio-based similarity metric. For word order-sensitive approaches, we test edit distance (Wagner and Fisher, 1974; Planas and Furuse, 1999), "sequential correspondence" and "weighted sequential correspondence".

Each of the similarity metrics empirically describes the similarity between two input strings tm_i and in,[2] where we define tm_i as a source language string taken from the TM and in as the input string which we are seeking to match within the TM.

One feature of all similarity metrics given here is that they have fine-grained discriminatory potential and are able to narrow down the final set of translation candidates to a handful of, and in most cases one, output. This was a deliberate design decision, and aimed at example-based machine translation applications, where human judgement cannot be relied upon to single out the most appropriate translation from multiple system outputs. In this, we set ourselves apart from the research of Sumita and Tsutsumi (1991), for example, who judge the system to have been successful if there are a total of 100 or less outputs, and a useful translation is contained within them. Note that it would be a relatively simple pro-

[2]Note that the ordering here is arbitrary, and that all the similarity metrics described herein are commutative for the given implementations.

cedure to fan out the number of outputs to n in our case, by taking the top n ranking outputs.

For all similarity metrics, we weight different Japanese segment types according to their expected impact on translation, in the form of the *sweight* function:

Segment type	*sweight*
punctuation	0
other segments	1

We experimentally trialled intermediate *sweight* settings for different character types (in the case of character-based indexing) or segment types (in the case of word-based indexing), none of which was found to appreciably improve performance.[3]

3.1 Similarity metrics used in this research

Vector space model

Within our implementation of the vector space model (VSM), the segment content of each string is described as a vector, made up of a single dimension for each segment token occurring within tm_i or in. The value of each vector component is given as the weighted frequency of that token according to its *sweight* value, such that any number of a given punctuation mark will produce a frequency of 0. The string similarity of tm_i and in is then defined as the cosine of the angle between vectors $\vec{tm_i}$ and $\vec{in}$, respectively, calculated as:

$$\cos(\vec{tm_i}, \vec{in}) = \frac{\vec{tm_i} \cdot \vec{in}}{|\vec{tm_i}||\vec{in}|} \quad (1)$$

where dot product and vector length coincide with the standard definitions.

The strings tm_i of maximal similarity are those which produce the maximum value for the vector cosine.

Note that VSM considers only segment frequency and is insensitive to word order.

Token intersection

The token intersection of tm_i and in is defined as the cumulative intersecting frequency of tokens appearing in each of the strings, normalised according to the combined segment lengths of tm_i and in. Formally, this equates to:

$$tint(tm_i, in) = \frac{2 \times \sum_t \min\left(freq_{tm_i}(t), freq_{in}(t)\right)}{len(tm_i)+len(in)} \quad (2)$$

where each t is a token occurring in either tm_i or in, $freq_s(t)$ is defined as the *sweight*-based frequency of token t occurring in string s, and $len(s)$ is the segment length of string s, that is the *sweight*-based count of segments contained in s.

As for VSM, the string(s) tm_i most similar to in are those which generate the maximum value for $tint(tm_i, in)$.

Note that word order does not take any part in calculation.

Edit distance

The first of the word order-sensitive methods is edit distance (Wagner and Fisher, 1974; Planas and Furuse, 1999). Essentially, the segment-based edit distance between strings tm_i and in is the minimum number of primitive edit operations on single segments required to transform tm_i into in (and vice versa), based upon the operations of *segment equality* (segments $tm_{i,m}$ and in_n are identical), *segment deletion* (delete segment a from a given position in string s) and *segment insertion* (insert segment a into a given position in string s). The cost associated with each operation on segment a is defined as:[4]

Operation	*Cost*
segment equality	0
segment deletion	$sweight(a)$
segment insertion	$sweight(a)$

Unlike other similarity metrics, smaller values indicate greater similarity for edit distance, and identical strings have edit distance 0.

The word order sensitivity of edit distance is perhaps best exemplified by way of the following example, where segment delimiters are given as '·'.

(1) E_· $N· 1+ "winter rain"
(2a) 2F· $N· 1+ "summer rain"
(2b) 1+· $N· 2F "a rainy summer"

Here, the edit distance from (1) to (2a) is $1 + 1 = 2$, as one deletion operation is required to remove E_ [*fuyu*] "winter" and one insertion operation required to add 2F [*natu*] "summer". The edit distance from (1) to (2b), on the other hand, is $1 + 1 + 1 + 1 = 4$ despite (2b) being identical in segment content to (2a). In terms of edit distance, therefore, (2a) is adjudged more similar to (1) than (2b).

Sequential correspondence

Sequential correspondence is a measure of the maximum substring similarity between tm_i and in, normalised according to the combined segment lengths $len(tm_i)$ and $len(in)$. Essentially, this method requires that all substring matches $submatch(tm_i, in)$ between tm_i and in be calculated, and the maximum *seqcorr* ratio returned, where *seqcorr* is defined as:

$$seqcorr(tm_i, in) = \frac{2 \times \max|submatch(tm_i, in)|}{len(tm_i)+len(in)} \quad (3)$$

[3] If anything, weighting down hiragana characters, for example, due to their common occurrence as inflectional suffices or particles (as per Fujii and Croft (1993)) led to a significant drop in performance. Similarly, weighting down stop word-like functional parts-of-speech in Japanese had little effect, unlike weighting down stop words in the case of English (see below).

[4] Note that the costs for deletion and insertion must be equal to maintain commutativity.

Here, the cardinality operator applied to $submatch(tm_i, in)$ returns the combined segment length of matching substrings, weighted according to *sweight*. That is:

$$|submatch(tm_i,in)|=\sum_{ss_j}\sum_k sweight(ss_{j,k}) \quad (4)$$

for each segment $ss_{j,k}$ of each matching substring $ss_j \in submatch(tm_i, in)$.

Returning to our example from above, the similarity for (1) and (2a) is $\frac{2\times2}{3+3}=\frac{2}{3}$, whereas that for (1) and (2b) is $\frac{2\times1}{3+3}=\frac{1}{3}$.

Weighted sequential correspondence

Weighted sequential correspondence—the last of the word order-sensitive methods—is an extension of sequential correspondence. It attempts to supplement the deficiency of sequential correspondence that the contiguity of substring matches is not taken into consideration. Given input string $a_1\,a_2\,a_3\,a_4$, for example, sequential correspondence would suggest equal similarity (of $\frac{8}{11}$) with strings $a_1\,b\,a_2\,c\,a_3\,d\,a_4$ and $a_1\,a_2\,a_3\,a_4\,cfg$, despite the second of these being more likely to produce a translation at least partially resembling that of the input string.

We get around this by associating an incremental weight with each matching segment assessing the contiguity of left-neighbouring segments, in the manner described by Sato (1992) for character-based matching. Namely, the kth segment of a matched substring is given the multiplicative weight $\min(k, Max)$, where Max was set to 4 in evaluation after Sato. $|submatch(tm_i, in)|$ from equation (3) thus becomes:

$$\sum_{ss_j}\sum_k \min\Big(k \times sweight(ss_{j,k}), Max\Big) \quad (5)$$

for each substring $ss_j \in submatch(tm_i, in)$. We similarly modify the definition of the *len* function for a string s to:

$$len(s) = \sum_j \min\Big(j \times sweight(s_j), Max\Big) \quad (6)$$

for each segment s_j of s.

3.2 Retrieval speed optimisation

While this paper is mainly concerned with accuracy, we take a moment out here to discuss the potential to accelerate the proposed methods, to get a feel for their relative speeds in actual retrieval.

One immediate and effective way in which we can limit the search space for all methods is to use the current top-ranking score in establishing upper and lower bounds on the length of strings which have the potential to better that score. For token intersection, for example, from the fixed length $len(in)$ of input string in and current top score α, we can calculate the following bounds based on the greatest possible degree of match between in and tm_i:

$$\text{Upper bound:} \quad len(tm_i) \leq \lfloor \tfrac{(2-\alpha)\,len(in)}{\alpha} \rfloor \quad (7)$$

$$\text{Lower bound:} \quad len(tm_i) \geq \lceil \tfrac{\alpha\,len(in)}{2-\alpha} \rceil \quad (8)$$

In a similar fashion, we can stipulate a corridor of allowable segment lengths for tm_i, for sequential correspondence and weighted sequential correspondence.

For edit distance, we make the observation that for a current minimum edit distance of α, the following inequality over $len(tm_i)$ must be satisfied for tm_i to have a chance of bettering α:

$$len(in) - \alpha \leq len(tm_i) \leq len(in) + \alpha \quad (9)$$

We can also limit the number of string comparisons required to reach the optimal match with in, by indexing each tm_i by its component segments and working through the component segments of in in ascending order of global frequency. At each iteration, we consider each previously unmatched translation record containing the current segment token, adjusting the upper and lower bounds as we go, given that translation records for a given iteration cannot have contained segment tokens already processed. The maximum possible segment correspondence between the strings is therefore decreasing on each iteration. We are also able to completely discount strings with no segment component common with in in this way.

Through these two methods, we were able to greatly reduce the number of string comparisons in word-based indexing evaluation for VSM, token intersection, sequential correspondence and weighted sequential correspondence methods in particular, and edit distance to a lesser degree. The degree of reduction for character-based indexing was not as marked, due to the massive increase in numbers of translation records sharing some character content with in.

There is also considerable scope to accelerate the matching mechanisms used by the word order-sensitive approaches. Currently, all approaches are implemented in Perl 5, and the word order-sensitive approaches use a naive, highly recursive method to exhaustively generate all substring matches and determine the similarity for each. One obvious way in which we could enhance this implementation would be to use an N-gram index as proposed by Nagao and Mori (1994). Dynamic Programming (DP) techniques would undoubtedly lead to greater efficiency, as suggested by Cranias et al. (1995, 1997) and also Planas and Furuse (this volume).

4 Evaluation

4.1 Evaluation specifications

Evaluation was partitioned off into character-based and word-based indexing for the various similarity methods. For word-based indexing, segmentation was carried out with ChaSen v2.0b (Matsumoto et al., 1999). No attempt was made to post-edit the segmented output, in interests of maintaining consistency in the data. Segmented and non-segmented strings were tested using a single program, with segment length set to a single character for non-segmented strings.

As test data, we used 2336 unique translation records deriving from technical field reports on construction machinery translated from Japanese into English. Translation records varied in size from

	Similarity metric	*Accuracy*	*Edit discrep.*	*Ave. outputs*	*Ave. time*
CHARACTER-BASED INDEXING	Vector space model (0.5)	44.0	4.86	1.04 (0.97)	2.14
	Token intersection (0.4)	44.3	3.25	1.01 (0.99)	2.24
	Edit distance ($len(in)$)	**50.2**	1.82	1.39 (0.80)	4.75
	Sequential corr. (0.4)	**46.6**	2.92	1.02 (0.98)	3.20
	Weighted seq. corr. (0.2)	**45.6**	2.89	1.04 (0.97)	4.10
WORD-BASED INDEXING	Vector space model (0.5)	43.7 (-0.8%)	5.21	1.17 (0.91)	0.76
	Token intersection (0.4)	43.0 (-2.9%)	3.12	1.01 (0.99)	0.88
	Edit distance ($len(in)$)	47.3 (-5.9%)	2.03	1.90 (0.69)	1.00
	Sequential corr. (0.4)	43.1 (-7.4%)	3.06	1.01 (0.99)	1.10
	Weighted seq. corr. (0.2)	40.7 (-10.7%)	3.30	1.14 (0.92)	1.24

Table 1: Results for the different similarity metrics under character-based and word-based indexing

single-word technical terms taken from a technical glossary, to multiple-sentence strings, at an average segment length of 13.4 and average character length of 26.1. All Japanese strings of length 6 characters or more (a total of 1802 strings) were extracted from the test data, leaving a residue glossary of technical terms (533 strings) as we would not expect to find useful matches in the TM. The retrieval accuracy over the 1802 longer strings was then verified by 10-fold cross validation, including the glossary in the test TM on each iteration.

Note that the test data was pre-partitioned into single technical terms, single sentences or sentence clusters, each constituting a single translation record. Partitions were taken as given in evaluation, whereas for real-world TM systems, the automation of this process comprises an important component of the overall system, preceding translation retrieval. While acknowledging the importance of this step and its interaction with retrieval performance, we choose to sidestep it for the purposes of this paper, and leave it for future research.

In an effort to make evaluation as objective and empirical as possible, appropriateness of translation candidate(s) proposed by the different metrics was evaluated according to the minimum edit distance between the translation candidate(s) and the unique model translation. In this, we transferred the edit distance method described above directly across to the target language (English), with segments as words and the following *sweight* schema:

Segment type	*sweight*
punctuation	0
stop words	0.2
other words	1

Stop words are defined as those contained within the SMART (Salton, 1971) stop word list.[5] The system output was judged to be correct if it contained a translation optimally close to the model translation; the average optimal edit distance from the model translation was 4.73.

[5]ftp://ftp.cornell.cs.edu/pub/smart/english.stop

We set the additional criterion that the different metrics should be able to determine whether the top-ranking translation candidate is likely to be useful to the translator, and that no output should be given if the closest matching translation record was outside a certain range of "translation usefulness". In practice, this was set to the edit distance between the model translation and the empty string (i.e. the edit cost of creating the model translation from scratch). This cutoff point was realised for the different similarity metrics by thresholding over the similarity scores. The different thresholds settled upon experimentally for all similarity metrics are given in brackets in the second column of Table 1, with the threshold for edit distance dynamically set to the edit distance between the input and the empty string.

We set ourselves apart from conventional research on TM retrieval performance in adopting this objective numerical evaluation method. Traditionally, retrieval performance has been gauged by the subjective usefulness of the closest matching element of the system output (as judged by a human), and described by way of a discrete set of translation quality descriptors (e.g. (Nakamura, 1989; Sumita and Tsutsumi, 1991; Sato, 1992)). Perhaps the closest evaluation attempts to what we propose are those of Planas and Furuse (1999) in setting a mechanical cutoff for "translation usability" as the ability to generate the model translation from a given translation candidate by editing less than half the component words, and Nirenburg et al. (1993) in calculating the weighted number of key strokes required to convert the system output into an appropriate translation for the original input. The method of Nirenburg et al. (1993) is certainly more indicative of true target language usefulness, but is dependent on the competence of the translator editing the TM system output, and not automated to the degree our method is.

4.2 Results

The results for the different similarity metrics with character-based and word-based indexing are given in Table 1, with the two bag-of-words approaches partitioned off from the three word order-sensitive approaches for each indexing paradigm. "Accuracy" is an indication of the proportion of inputs for which

an optimal translation was produced; character-based indexing accuracies in bold indicate a significant[6] advantage over the corresponding wprd-based indexing accuracy, and figures in brackets for word-based indexing indicate the relative performance gain over the corresponding character-based indexing configuration. "Edit discrep." refers to the mean minimum edit distance discrepancy between translation candidate(s) and optimal translation(s) in the case of the translation candidate set containing no optimal translations. "Ave. outputs" describes the average number of translation candidates output by the system, with the figure in brackets being the proportion of inputs for which a unique translation candidate was produced. "Ave. time" describes the average time taken to determine the translation candidate(s) for a single output, relative to the time taken for word-based edit distance retrieval.

Perhaps the most striking result is that character-based indexing produces a superior match accuracy to word-based indexing for *all* similarity metrics, at a significant margin for all three word order-based methods. This is the complete opposite of what we had expected, although it does fit in with the findings of Fujii and Croft (1993) that character-based indexing performs comparably with word-based indexing in Japanese information retrieval.

Looking to word order, we see that edit distance outperforms all other methods for both character- and word-based indexing, peaking at just over 50% for character-based indexing. The relative performance of the remaining methods is variable, with the two bag-of-words methods being superior to or roughly equivalent to sequential correspondence and weighted sequential correspondence for word-based indexing, but the word order-based methods having a clear advantage over the bag-of-words methods for character-based indexing. It is thus difficult to draw any hard and fast conclusion as to the relative merits of word order-based versus bag-of-words methods, other than to say that edist distance would appear to have a clear advantage over other methods.

The figures for edit discrepancy in the case of non-optimal translation candidate(s) are equally interesting, and suggest that on the whole, the various methods err more conservatively for character-based than word-based indexing. The most robust method is (source language) edit distance, at an edit discrepancy of 1.82 and 2.03 for character-based and word-based indexing, respectively.

All methods were able to produce just over one translation candidate on average, with all other than edit distance returning a unique translation candidate over 90% of the time. The greater number of outputs for the edit distance method can certainly be viewed as one reason for its inflated performance, although the lower level of ambiguity for character-based indexing but higher accuracy, would tend to suggest otherwise.

Lastly, word-based indexing was found to be faster than character-based indexing across the board, for the simple reason that the number of character segments is always going to be greater than or equal to the number of word segments. The average segment lengths quoted above (26.1 characters vs. 13.4 words) indicate that we generally have twice as many characters as words in a given string. Additionally, the acceleration technique described in § 3.2 of sequentially working through the segment component of the input string in increasing order of global frequency, has a greater effect for word-based indexing than character-based indexing, accentuating any speed disparity.

4.3 Reflections on the results

An immediate explanation for character-based indexing's empirical edge over word-based indexing is the semantic smoothing effects of individual kanji characters, alluded to above (§ 2). To take an example, the single-segment nouns A':n [*sōsa*] and :nF0 [*sadō*] both mean "operation", but would not match under word-based indexing. Character-based indexing, on the other hand, would recognise the overlap in character content, and in the process pick up on the semantic correspondence between the two words.

To take the opposite tack, one reason why word-based indexing may have been disadvantaged is the we did not stem or lemmatise words in word-based indexing. Having said this, the output from ChaSen is such that stems of inflecting words are given as a single segment, with inflectional morphemes each presented as separate segments. In this sense, stemming would only act to delete the inflectional morphemes, and not add anything new.

Another way in which the output of ChaSen could conceivably have affected retrieval performance is that technical terms tended to be over-segmented. Experimentally combining recognised technical terms into a single segment (particularly in the case of contiguous katakana segments in the manner of Fujii and Croft (1993)), however, degraded rather than improved retrieval performance for both character-based and word-based indexing. As such, this side-effect of ChaSen would not appear to have impinged on retrieval accuracy.

One other plausible reason for the unexpected results is that the test data could have been in some way inherently better suited to character-based indexing than word-based indexing, although the fact that the results were cross-validated would tend to rule out this possibility.

A surprising result was the lacklustre performance of the weighted sequential correspondence method as compared to simple sequential correspondence. We have no explanation for the drop in accuracy, other than to speculate that either the proposed formulation is in some way flawed or contiguity of match does not impinge on translation similarity to the degree we had expected.

To return to the original question posed above of retrieval speed vs. accuracy, the word order-sensitive edit distance approach would seem to hold a genuine edge over the other methods, to an order that would suggest the extra computational overhead is warranted, in both accuracy and translation discrepancy. It must be said that the TM used in evalua-

[6] As determined by the paired t test ($p < 0.05$).

tion was too small to get a genuine feel for the computational overhead that would be experienced in a real-world TM system context of potentially millions rather than thousands of translation records. At the same time, however, coding up the edit distance procedure in a language faster than Perl using character rather than string comparison procedures and applying dynamic programming techniques or similar, may well offset the large increase in number of comparisons demanded of the system.

5 Concluding remarks

This research is concerned with the relative import of word order and segmentation on translation retrieval performance for a TM system. We modelled the effects of word order sensitivity vs. bag-of-words word order insensitivity by implementing a total of five similarity metrics: two bag-of-words approaches (the vector space model and "token intersection") and three word order-sensitive approaches (edit distance, "sequential correspondence" and "weighted sequential correspondence"). Each of these metrics was then tested under character-based and word-based indexing, to determine what effect segmentation would have on retrieval performance. Empirical evaluation based around the target language edit distance of proposed translation candidates revealed that character-based indexing consistently produced greater accuracy than word-based indexing, and that the word order-sensitive edit distance metric clearly outperformed all other methods under both indexing paradigms.

The main area in which we feel this research could be enhanced is to validate the findings of this paper in expanding evaluation to other domains and test sets, which we leave as an item for future research. We also skirted around the issue of translation record partitioning, and wish to investigate how different partitioning methods perform against each other. One important area in which we hope to expand our research is to look at the effects of character type on character-based indexing. Kanji would appear to be helping the case of character-based indexing at present, and it would be highly revealing to look at whether comparable results to those presented here would be produced for full kana-based (alphabetic) Japanese input, or other alphabet-based non-segmenting languages such as Thai.

Acknowledgements

Vital input into this research was received from Francis Bond (NTT), Emmanuel Planas (NTT), and three anonymous reviewers.

References

T. Baldwin and H. Tanaka. 1999. The applications of unsupervised learning to Japanese grapheme-phoneme alignment. In *Proc. of the ACL Workshop on Unsupervised Learning in Natural Language Processing*, pages 9–16.

L. Cranias, H. Papageorgiou, and S. Piperidis. 1995. *A Matching Technique in Example-Based Machine Translation.* cmp-lg/9508005.

L. Cranias, H. Papageorgiou, and S. Piperidis. 1997. Example retrieval from a translation memory. *Natural Language Engineering*, 3(4):255–77.

H. Fujii and W.B. Croft. 1993. A comparison of indexing techniques for Japanese text retrieval. In *Proc. of 16th International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 237–46.

E. Kitamura and H. Yamamoto. 1996. Translation retrieval system using alignment data from parallel texts. In *Proc. of the 53rd Annual Meeting of the IPSJ*, volume 2, pages 385–6. (In Japanese).

C. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press.

Y. Matsumoto, A. Kitauchi, T. Yamashita, and Y. Hirano. 1999. *Japanese Morphological Analysis System ChaSen Version 2.0 Manual.* Technical Report NAIST-IS-TR99009, NAIST.

M. Nagao and S. Mori. 1994. A new method of N-gram statistics for large number of N and automatic extraction of words and phrases from large text data of Japanese. In *Proc. of the 15th International Conference on Computational Linguistics (COLING '94)*, pages 611–5.

N. Nakamura. 1989. Translation support by retrieving bilingual texts. In *Proc. of the 38th Annual Meeting of the IPSJ*, volume 1, pages 357–8. (In Japanese).

S. Nirenburg, C. Domashnev, and D.J. Grannes. 1993. Two approaches to matching in example-based machine translation. In *Proc. of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-93)*, pages 47–57.

E. Planas and O. Furuse. 1999. Formalizing translation memories. In *Proc. of Machine Translation Summit VII*, pages 331–9.

E. Planas. 1998. *A Case Study on Memory Based Machine Translation Tools.* PhD Fellow Working Paper, United Nations University.

G. Salton. 1971. *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall.

S. Sato and T. Kawase. 1994. *A High-Speed Best Match Retrieval Method for Japanese Text.* Technical Report IS-RR-94-9I, JAIST.

S. Sato. 1992. CTM: An example-based translation aid system. In *Proc. of the 14th International Conference on Computational Linguistics (COLING '92)*, pages 1259–63.

E. Sumita and Y. Tsutsumi. 1991. A practical method of retrieving similar examples for translation aid. *Transactions of the IEICE*, J74-D-II(10):1437–47. (In Japanese).

H. Tanaka. 1997. An efficient way of gauging similarity between long Japanese expressions. In *Information Processing Society of Japan SIG Notes*, volume 97, no. 85, pages 69–74. (In Japanese).

A. Wagner and M. Fisher. 1974. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–73.