IEICE Technical Report, .

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

表層的・構造的対訳対検索の長所と短所

Timothy Baldwin† 岡崎 篤†† 徳永 健伸†† 田中 穂積††

† Centre for the Study of Language and Information (CSLI)
210 Panama Street
Stanford, CA 94305-4115, U.S.A.
†† 東京工業大学 情報理工学研究科
〒 152-8552 東京都目黒区大岡山 2-12-1

E-mail: †tbaldwin@csli.stanford.edu, ††{okazaki,take,tanaka}@cl.cs.titech.ac.jp

あらまし 本研究では、SENSEVAL2 の対訳対検索タスクにおいて、2 つの全く異なった検索手法を紹介する。1 つ目の手法は表層的類似に基づくもので、文字列を文字バイグラムの集合として扱う。2 つ目の手法は構造的類似に基づくもので、構文解析木および概念的類似を用い文字列間の類似度を計算する。さらに、この 2 つの手法を組み合わせるハイブリッド手法も提案する。評価実験では、単純でありながらも、表層的類似度計算法が構造的計算法より勝ることを明らかにし、全体ではハイブリッド手法が再優良であることを実証している。

キーワード 対訳対検索, 表層的類似, 構造的類似

The Successes and Failures of Lexical and Structural Translation Retrieval

Timothy BALDWIN†, Atsushi OKAZAKI††, Takenobu TOKUNAGA††, and Hozumi TANAKA††

† Centre for the Study of Language and Information (CSLI)
210 Panama Street
Stanford, CA 94305-4115, U.S.A.
†† Department of Computer Science
Graduate School of Information Science and Engineering
Tokyo Institute of Technology
2-12-1 Ō-okayama, Meguro-ku, Tokyo 152-8552, Japan

E-mail: †tbaldwin@csli.stanford.edu, ††{okazaki,take,tanaka}@cl.cs.titech.ac.jp

Abstract This paper describes two distinct translation retrieval methods within the context of the SENSEVAL2 Japanese translation task. The first is based on lexical similarity and models strings as a bag of character bigrams, whereas the second is based on structural similarity and determines similarity via parse trees and conceptual similarity. We also discuss a hybrid approach, combining the results of the lexical and structural methods. Despite its simplistic nature, the lexical method was found to outperform the structural method by a clear margin, but the hybrid method to be the strongest overall contender.

Key words translation retrieval, lexical similarity, structural similarity

1. Introduction

Translation retrieval is defined as the task of, for a given source language (L1) input, retrieving the target language (L2) string which best translates it. Retrieval is carried out over a **translation memory** made up of **translation records**, that is L1 strings coupled with an L2 translation. A single translation retrieval task was offered in SENSEVAL2, from Japanese into English, and it is this task that we target in this paper.

Conventionally, translation retrieval is carried out by way of determining the L1 string in the translation memory most similar to the input, and returning the L2 string paired with that string as a translation for the input. It is important to realise that at no point is the output compared back to the input to determine its "translation adequacy", a job which is left up to the system user.

Determination of the degree of similarity between the input and L1 component of each translation record can take a range of factors into consideration, including lexical (character or word) content, word order, parse tree topology and conceptual similarity. In this paper, we focus on a simple character-based (**lexical**) method and more sophisticated parse tree comparison (**structural**) method. We also touch on the possibility of a hybrid method combining the results of the lexical and structural methods.

Both methods discussed herein are fully unsupervised. The lexical method makes use of no external resources or linguistic knowledge whatsoever. It treats each string as a "bag of character bigrams" and calculates similarity according to Dice's Coefficient. The structural method, on the other hand, relies on both morphological and syntactic analysis, in the form of the publicly-available JUMAN (Kurohashi and Nagao 1998b) and KNP (Kurohashi and Nagao 1998a) systems, respectively, and also the Japanese Goi-Taikei thesaurus (Ikehara *et al.* 1997) to measure conceptual distance. A parse tree is generated for the L1 component of each translation record, and also each input, and similarity gauged by both topological resemblance between parse trees and conceptual similarity between nodes of the parse tree.

Translation records used by the two systems were taken exclusively from the translation memory provided for the task.

In the proceeding sections, we briefly review the Japanese translation task (Section 2.) and detail our particular use of the data provided for the task (Section 3.). Next, we outline the lexical method (Section 4.) and structural method (Section 5.), and compare and discuss the performance of the lexical and structural methods, and also a number of hybrid methods (Section 6.).

2. Basic task description

The Japanese translation task data took the form of a translation memory and test set. The translation memory was dissected into 320 disjoint segments according to **headwords**, with an average of 21.6 translation records per headword (i.e. 6920 translation records overall). The purpose of the task was to select for a given headword which of the translation records gave a suitable translation for that word. The task stipulated that a maximum of one translation record could be selected for each input (allowing for the possibility of an `unassignable` output, indicating that no appropriate translation could be found). Translations were selected by way of a translation record ID, and systems were not required to actually identify what part of the L2 string in the selected translation record was the translation for the headword.

Translation records took the form of Japanese–English pairs of word clusters, isolated phrases, clauses and sentences containing the headword, at an average of 8.0 Japanese characters[1] and 4.0 English words per translation record. In some instances, multiple semantically-equivalent translations were given for a single expression, such as "corporation which is in danger of bankruptcy" and "unsound corporation" for *abunai kigyō*; all such occurrences were marked by the annotator. For some other translation records, the annotator had provided a list of lexical variants or a paraphrase of the L1 expression to elucidate its meaning (not necessarily involving the headword), or made a note

[1] Based on the count of kana and kanji characters, and individual digits.

as to typical arguments taken by that expression (e.g. "refers to a person"). Generally, more frequent/general senses occurred earlier in the translation memory.

In the test data, inputs took the form of paragraphs taken from newspaper articles, within which a single headword had been identified for translation. The average input length was 697.9 characters, nearly 90 times the L1 component of each translation record. In its raw form, therefore, the translation task differs from a conventional translation retrieval task in that translation records and inputs are not directly comparable, in the sense that translation records are never going to provide a full translation approximation for the overall input.

For each input in the test data, three disjoint sets of translation record IDs were provided, discriminated according to translation quality. The first set of translation record IDs (**quality level 1**) contained those which provided a high-quality translation for the headword, from the translation memory; the second set (**quality level 2**) listed those translation records which provided an acceptable but sub-optimal translation; and the third set (**quality level 3**) listed those translation records which provided a marginally acceptable translation for the input. In the official system evaluation, an output which fell into any of the three categories was taken to be correct.

3. Data preparation

In adapting the task data to our purposes, we first carried out limited normalisation of both the translation memory and test data by: (a) replacing all numerical expressions with a common `NUM` marker, (b) normalising sentence-final punctuation, and (c) standardising in-clause punctuation.

In order to maximise the disambiguating potential of the translation memory, we next set about automatically deriving as many discrete translation records as possible from the original translation memory. Multiple lexical variants of the same basic translation record (indexed identically) were generated in the case that: (a) a lexical alternate was provided (in which case all variants were listed in parallel); (b) a paraphrase was provided by the annotator (irrespective of whether the paraphrase included the headword or not); (c) syntactic or semantic preferences were listed for particular arguments in the basic translation record (in which case lexical variants took the form of strings expanded by adding in each preference as a string). At the same time, for each headword, any repetitions of the same L1 string were completely removed from the translation record data. This equates to the assumption that the translation listed first in the translation memory is the most salient or commonplace.

This method of translation record derivation resulted in a total of 152 new translation records, whereas the removal of duplicate L1 strings for a given headword resulted in the deletion of 670 translation records; the total number of translation records was thus 6402, at an average of 20.0 translation records per headword.

We experimented with a number of methods for abbreviating the inputs, so as to achieve direct comparability between inputs and translation records. First, we extracted the clause containing the headword instance to be translated. This was achieved through a number of ad hoc heuristics driven by the analysis of punctuation. These clause-level instances served as the inputs for the *structural* method. We then tested further "windowing" the inputs for the *lexical* method, by allowing a maximum of 10 characters to either side of the headword. No attempt was made to identify or enforce the observation of word boundaries in this process. In evaluation, we test the lexical method on both windowed and full clauses.

4. The lexical method

As stated above, the lexical method is based on character-based indexing, meaning that each string is naively treated as a sequence of characters. Rather than treat each individual character as a single segment, however, we chunk adjacent characters into bigrams in order to capture local character contiguity. String similarity is then determined by way of Dice's Coefficient, calculated according to:

$$sim_1(IN^*_m, TR_i) = \frac{2 \times \sum_{e \in IN^*_m, TR_i} \min\left(freq_{IN^*_m}(e), freq_{TR_i}(e)\right)}{len(IN^*_m) + len(TR_i)}$$

where IN^*_m is the abbreviated version of the input string IN_m (see above) and TR_i is a translation record; each e is a character bigram occurring in either IN^*_m or TR_i, $freq_{IN^*_m}(e)$ is defined as the weighted frequency of bigram type e in IN^*_m, and $len(IN^*_m)$ is the character bigram length of IN^*_m.[2] Bigram frequency is weighted according to character type: a bigram made up entirely of hiragana characters (generally used in functional words/particles) is given a weight of 0.2 and all other bigrams a weight of 1. Note that Dice's Coefficient ignores segment order, and that each string is thus treated as a "bag of character bigrams".

Our choice of the combination of Dice's Coefficient, character-based indexing and character bigrams (rather than any other n-gram order or mixed n-gram model) is based on the findings of Baldwin (2001b, 2001a), who compared character- and word-based indexing in combination with both segment order-sensitive and bag-of-words similarity measures and with various n-gram models. As a result of extensive evaluation, Baldwin found the combination of character bigram-based indexing and a bag-of-words method (in the form of either the vector space model or Dice's Coefficient) to be optimal. Our choice of Dice's Coefficient over the vector space model is due to the vector space model tending to have a bias towards shorter strings in cases of low-level character overlap, and the ability of Dice's Coefficient to pick on subtle string similarities under such high-noise conditions.[3]

Given the limited lexical context in translation records (8.0 Japanese characters on average), our method is highly susceptible to the effects of data sparseness. While we have no immediate way of reconciling this shortcoming, it is possible to make use of the rich lexical context of the full inputs (i.e. in original paragraph form rather than clause or windowed clause form). Direct comparison of the full inputs with translation records is undesirable as high levels of spurious matches can be expected outside the scope of the original translation record expression. Inter-comparison of full inputs, on the other hand, provides a primitive model of domain similarity. Assuming that high similarity correlates with a high level of domain correspondence, we can apply a cross-lingual corollary of the "one sense per discourse" observation (Gale *et al.* 1992) in stipulating that a given word will be translated consistently within a given domain. By ascertaining that a given input closely resembles a second input, we can use the combined translation retrieval results for the two inputs to hone in on the optimal translation for the two. We term this procedure **domain-based similarity consolidation**.

The overall retrieval process thus takes the form of first carrying out standard translation retrieval based on the abbreviated input, and second using the original test set to determine the full input string most similar to the current input, and performing translation retrieval independently using the abbreviated form of the maximally similar alternate input. Numerically, the combined similarity is calculated as:

$$sim_2(IN_m, TR_i) = 0.5 \Big(sim_1(IN^*_m, TR_i) + \max_{n \neq m} sim_1(IN_m, IN_n)\ sim_1(IN^*_n, TR_i) \Big)$$

where IN_m is the current input (full form), IN^*_m is the abbreviated form of IN_m, sim_1 is as defined above, and $IN_{n \neq m}$ is any input string other than the current input. Note that the multiplication by 0.5 simply normalises the output of sim_2 to the range $[0, 1]$. For each input IN_m, the ID for that translation record which is deemed most similar to IN_m is returned, with translation records occurring earlier in the translation memory selected in the case of a tie.[4]

As a slightly-modified alternative to sim_2, it is possible to take the mean overall trans-document similarity rather than the single highest similarity, as follows:

$$sim_3(IN_m, TR_i) = 0.5 \Big(sim_1(IN^*_m, TR_i)$$

[2] $freq_{TR_i}(e)$ and $len(TR_i)$ are defined similarly.

[3] This prediction for Dice's Coefficient to outperform the vector space model was validated empirically, and found to hold for all lexical method parameter settings given in Section 6.

[4] Based on the observation that translation records are roughly ordered according to commonality. Ties were observed 7.5% of the time, with the mean number of top-scoring translation records being 1.12.

$$+\frac{1}{|IN|-1}\sum_{n\neq m} sim_1(IN_m, IN_n)\ sim_1(IN^*_n, TR_i)\Big)$$

Here, IN is the set of all inputs.

One advantage to having such a fine-grained numeric representation of similarity, is that we are able to take the similarity value as being indicative of the quality of the associated translation. We are thus able to set an arbitrary threshold value, for example, to filter off more questionable translation outputs, and by tailoring the threshold value to the needs of the task at hand, can be more or less conservative in our provision of translation candidates.

5. The structural method

The structural method contrasts starkly with the lexical method in that it is heavily resource-dependent, requiring a morphological analyser, parser and thesaurus. It operates over the same translation memory data as the lexical method, but uses only the abbreviated forms of the inputs (to the clause level) and does not make use of domain-based similarity consolidation.

JUMAN (Kurohashi and Nagao 1998b) is first used to segment each string (translation records and inputs), based on the output of which, the KNP parser (Kurohashi and Nagao 1998a) is used to derive a parse tree for the string. The reason for abbreviating inputs only as far as the clause level for the structural method, is to enhance parseability. Further pruning takes place implicitly further downstream as part of the parse tree matching process.

KNP returns a binary parse tree, with leaves corresponding to optionally case-marked phrases. Each leaf node is simplified to the phrase head and the (optional) case marker normalised (according to the KNP output).

As for the lexical method, all translation records corresponding to the current headword are matched against the parse tree for the input, and the ID of the closest-matching tree returned. In comparing a given pair of parse trees T^1 and T^2, we proceed as follows in direction $d \in \{\text{up}, \text{down}\}$:

(1) Set p^1 to the leaf node containing the headword in T^1, and similarly initialise p^2 in T^2; initialise n to 0

(2) If $p^1_m \neq p^2_m$, return $(n, 0)$

(3) If $p^1_f \neq p^2_f$, return $(n, \mathit{concept\_sim}(p^1_f, p^2_f))$

(4) Increment n by 1, set p^1 and p^2 to their respective adjacent leaf nodes in direction d within the parse tree; goto step 2.

Here, p^i_m is the case marker associated with node p^i, p^i_f is the filler associated with node p^i, and the $\neq$ operator represents lexical equality. *concept_sim* calculates the conceptual similarity of the two fillers in question according to the Goi-Taikei thesaurus (Ikehara *et al.* 1997). We do this by, for each sense pairing of the fillers, determining the least common hypernym and the number of edges separating each sense node from the least common hypernym. The conceptual distance of the given senses is then determined according to the inverse of the greater of the two edge distances to the hypernym node, and the overall conceptual distance for the two fillers as the minimum such sense-wise conceptual distance.

We match both up and down the tree structure from the headword node, and evaluate the combined similarity as the sum of the individual elements of the returned tuples. That is, if an upward match returned (i, m) and a downward match (j, n), the overall similarity would be $(i + j, m + n)$. The translation output is the ID of the translation record producing the greatest such similarity, where $(w, x) > (y, z)$ iff $w > y$ or $(w = y \ \wedge \ x > z)$. As a result, conceptual similarity is essentially a tie-breaking mechanism, and the principal determining factor is the number of phrase levels over which the parse trees match. In the case that there is a tie for best translation, the translation record with the longest L1 string is chosen, and in the case that this doesn't resolve the stalemate, one of the tied translation records is chosen randomly. In the case that all translation records score $(0, 0)$, we deem there to be no suitable translation in the translation memory, and return `unassignable`.

As mentioned in Section 2., crude selectional preferences (of the form PERSON or BUILDING) were provided on certain argument slots in translation records. These were supported by semi-automatically mapping the preference type onto the Goi-Taikei thesaurus structure, and modifying the

$\dagger$ operator to non-sense subsumption of the translation record filler by the input selectional preference, in step 3 of the parse tree match algorithm. Selectional preferences were automatically mapped onto nodes of the same name if they existed, and manually linked to the thesaurus otherwise.

6. Results and discussion

In this section, we evaluate the lexical and structural methods in the configuration used for the final SENSEVAL2 data submission, so as to maintain direct comparability with other systems performing in the translation task. In an effort to justify a number of design decisions made in the development of the final formulation of the two methods, we additionally test an array of close variants of the original configurations, and also a range of hybrid methods combining the outputs of the lexical and structural methods. The translation retrieval accuracies are given in Table 1, along with a baseline accuracy arrived at through random selection of a translation record from among the translation records associated with the given headword.[5] Note that as we attempt to translate all inputs, recall and precision are equal to the presented accuracy values.

First, the lexical method was tested with and without windowing of the input (*window* and *clause*, respectively), and with each of sim_1, sim_2 and sim_3. Of these, the official version used in our data submission was sim_2 with windowing. In all cases, we used a threshold of 0.02 to filter off more questionable translation outputs, setting the output to `unassignable` in the case that this threshold was not equalled or bettered.

Next, we experimented with a number of simple techniques for combining the results of the lexical and structural methods. First, we complemented the structural method by replacing all `unassignable` outputs with the corresponding output from the lexical method (*struct*>*lex*). We then repeated this method preference, complementing the lexical method with the structural method (*lex*>*struct*) by replacing any outputs below a fixed threshold with the corresponding output from the structural method, given that the structural method produces an output other than `unassignable`.

We break down the results for the various system configurations into three categories, reflecting how stringent our evaluation of system output translation quality is. In the first category ("quality level 3+"), the system output is judged to be correct if it corresponds to any of the three grades of translation quality given in the solution set (see Section 2.); in the second category ("quality level 2+"), the output must be contained in either the quality level 1 or 2 sets; and in the third category ("quality level 1"), only outputs evaluated as being of quality level 1 are treated as correct. By subclassifying system performance in this way, we gain some insight into the system's ability to select the better of the translations on offer. Given that quality level 3+ subsumes quality level 2+, and quality level 2+ in turn subsumes quality level 1, system accuracy across the three levels must be decreasing.[6] The lesser the observed decrease, however, the higher the overall quality of the translation output.

Looking to the actual results in Table 1, the most striking feature is that the lexical method has a clear advantage over the structural method for all quality levels, and both methods outperform the baseline by a reasonable margin in all their various incarnations. In its optimal configuration, (clause-level input representation using the sim_2 similarity measure), the lexical method just nudges past the 50% accuracy mark for quality level 3+; in the system configuration used for our official submission of results (*window*+sim_2), the quality level 3+ accuracy is 49.17%.[7] Notice that domain-based similarity consolidation (in the form of sim_2 and sim_3) enhances the results appreciably, and that the best method of similarity consolidation is to take the maximum rather than the mean trans-input similarity.

Obviously, it would be going too far to discount structural methods outright based on this limited evaluation, particularly as the lexical method

[5] Not including the `unassignable` output option.

[6] Although not necessarily strictly decreasing, as there is the possibility of constant accuracy across all three categories in the case that all outputs are of quality level 1.

[7] Thresholding was not used with the submitted version, resulting in an accuracy of 49.08%.

Retrieval method		*Accuracy (%)*		
		Quality level 3+	***Quality level 2+***	***Quality level 1***
Lexical	window+sim_3	46.25	45.42	41.50
	clause+sim_3	47.42	46.42	41.50
	window+sim_2	49.17	48.25	41.92
	clause+sim_2	50.08	49.08	42.00
	clause+sim_1	46.50	45.83	40.92
Structural		41.17	40.25	34.08
Lexical + structural	struct>lex	44.42	43.42	35.08
	lex>struct (0.1)	48.50	47.58	39.67
	lex>struct (0.06)	**50.83**	**49.83**	42.00
	lex>struct (0.04)	50.58	49.67	**42.08**
	lex>struct (0.02)	50.25	49.25	41.83
Baseline		38.17	37.65	30.72

Table 1 Results for the different system configurations, over varying levels of translation quality (system configurations used for official submission are underlined)

has been tested thoroughly over various datasets whereas the structural method was developed anew for this task. It is surprising, however, that a technique as simple as the lexical method, requiring no external resources and ignoring both word boundaries and word order, should perform so well. Perhaps one conclusion that can be drawn is that the given simplistic lexical method sets a high standard for any structural method to better, and the effort/level of mathematical sophistication required to equal the performance of the lexical method will most likely be great.

The main area in which the structural method fell short of the lexical method was inputs where no translation record displayed even the same case marking on the headword as the input and were hence judged to be `unassignable`. Indeed 130 or 10.8% of inputs fell into this category, despite there being only 34 or 0.3% of `unassignable` inputs in the test set. Ideally, we would like to be able to replace the current lexical match mechanism for case markers with a conceptual similarity measure analogous to that for case fillers, returning a real value in the range $[0, 1]$, which remains as a matter for future research.

For both the lexical and structural methods, there was a strong correlation between high-quality translations and higher-scoring matches, and the main area in which errors arose was more tentative (low-scoring) matches.

Hybridisation represents one way of overcoming the coverage problem for the structural method, and also the relatively depleted performance over low-scoring matches for the lexical method. Replacing all `unassignable` outputs from the structural method with the corresponding output from the lexical method (*struct>lex*) produces an appreciable gain over the basic structural method. The resultant system performance still falls short of that of the various lexical method variations, however. More encouraging is the finding that it is possible to boost the lexical method by substituting outputs from the structural method (assuming that an output exists) in the case of a low similarity score. The results presented in Table 1 are based on the *clause+*sim_2 version of the lexical method, and amount to a gain of nearly a percentage point for quality levels 3+ and 2+. The best overall results (presented in bold) for the various quality levels were observed with the similarity score threshold set to 0.06 and 0.04. That the hybrid methods should surpass both the component methods suggests that each has its individual merits, although we clearly need to have the more robust lexical method as our system backbone.

In conclusion, this paper has served to describe a structural translation retrieval method, a number of instantiations of a lexical translation retrieval method, and a hybrid method, in the context of the SENSEVAL2 Japanese translation task. The lexical method modelled strings as a bag of character bigrams, but incorporated a number of novel tech-

niques including domain-based similarity consolidation in reaching a final decision as to the translation record most similar to the input. The structural method, on the other hand, compared parse trees and had recourse to conceptual similarity, but in a relatively rudimentary form. The results of the lexical and structural methods were then combined in the hybrid methods, based on simple thresholding or substitution of `unassignable` outputs. Of the lexical and structural methods, the lexical method proved to be clearly superior, although both methods were well above the baseline performance. The best overall performance was observed for the hybrid method, combining the results of the lexical and structural methods.

Acknowledgements

This paper was supported in part by the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Company and CSLI, Stanford University.

References

Baldwin, T. 2001a. Low-cost, high-performance translation retrieval: Dumber is better. In *Proc. of the 39th Annual Meeting of the ACL and 10th Conference of the EACL (ACL-EACL 2001)*.

——, 2001b. *Making Lexical Sense of Japanese–English Machine Translation: A Disambiguation Extravaganza*. Tokyo Institute of Technology dissertation.

Gale, W., K. Church, and D. Yarowsky. 1992. One sense per discourse. In *Proc. of the 4th DARPA Speech and Natural Language Workshop*, 233–7.

Ikehara, S., M. Miyazaki, A. Yokoo, S. Shirai, H. Nakaiwa, K. Ogura, Y. Ooyama, and Y. Hayashi. 1997. *Nihongo Goi Taikei – A Japanese Lexicon*. Iwanami Shoten. 5 volumes. (In Japanese).

Kurohashi, S., and M. Nagao. 1998a. Building a Japanese parsed corpus while improving the parsing system. In *Proc. of the 1st International Conference on Language Resources and Evaluation (LREC'98)*, 719–24.

——, and ——. 1998b. *Nihongo keitai-kaiseki sisutemu JUMAN* [Japanese morphological analysis system JUMAN] version 3.5. Technical report, Kyoto University. (In Japanese).