The Japanese Translation Task: Lexical and Structural Perspectives

Timothy Baldwin,* **Atsushi Okazaki,†** **Takenobu Tokunaga†** and **Hozumi Tanaka†**

* **CSLI, Stanford University** <tbaldwin@csli.stanford.edu>

† **Tokyo Institute of Technology** <{okazaki,take,tanaka}@cl.cs.titech.ac.jp>

Abstract

This paper describes two distinct attempts at the Senseval2 Japanese translation task. The first implementation is based on lexical similarity and builds on the results of Baldwin (2001b; 2001a), whereas the second is based on structural similarity via the medium of parse trees and includes a basic model of conceptual similarity. Despite its simplistic nature, the lexical method was found to perform the better of the two, at 49.1% accuracy, as compared to 41.2% for the structural method and 36.8% for the baseline.

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.

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 (§ 2) and detail our particular use of the data provided for the task (§ 3). Next, we outline the lexical method (§ 4) and structural method (§ 5), and compare and discuss the performance of the two methods (§ 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

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

In Proceedings of the ACL/EACL 2001 Senseval2 Workshop, Touluse, France.

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 as to typical arguments taken by that expression (e.g. "refers to a person").

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.

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 further "windowed" 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.

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:

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

where IN^* is the abbreviated version of the input string IN (see above) and TR_i is a translation record; each e is a character bigram occurring in either IN^* or TR_i, $freq_{IN^*}(e)$ is defined as the weighted frequency of bigram type e in IN^*, and $len(IN^*)$ is the character bigram length of IN^*.[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 prefer 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.

Given the limited lexical context in translation records (8.0 Japanese characters on average), our method is highly susceptible to the effects of data

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

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, does provide 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:

$$sim_2(IN_m, TR_i) = 0.5 \times \big(sim(IN^*_m, TR_i) + sim(abbrev(\arg\max_{n \neq m} sim(IN_m, IN_n)), TR_i)\big)$$

where IN_m is the current input (full form), IN^*_m is the abbreviated form of IN_m, *sim* is as defined above, $IN_{n \neq m}$ is any input string other than the current input, and the *abbrev* operator returns the abbreviated version of the operand string (i.e. for IN, IN^* would be returned). 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 ties broken randomly.[3]

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 consider inter-input similarity.

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 $(n, -1)$
3. If $p^1_f \neq p^2_f$, return $(n, concept\_sim(p^1_f, p^2_f))$
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, translation record 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".

[3] Ties were observed 7.5% of the time, with the mean number of top-scoring translation records being 1.12.

Method	*Accuracy*
Lexical	49.1%
Structural	41.2%
Baseline	36.8%

Table 1: Results

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 $\neq$ 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

The translation retrieval accuracy for the two methods is given in Table 1, along with a baseline accuracy arrived at through random translation record selection for the given headword. Note that as we attempt to translate all inputs, the presented accuracy figures indicate both recall and precision.

The most striking feature of the results is that the lexical method has a clear advantage over the structural method, while both methods outperform the baseline. Obviously, it would be going too far to discount structural methods outright based on this limited evaluation, particularly as the lexical method has been tested thoroughly over various datasets, whereas the structural method is novel to 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.

The main area in which the structural method fell short was "unassignable" inputs where no translation record displayed even the same case marking on the headword. Indeed 130 or 10.8% of inputs were tagged "unassignable", despite there being only 34 or 0.3% of unassignable inputs in the solution 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.

Conversely for the lexical method, at present, a translation record is selected irrespective of the magnitude of the similarity value, and it would be a trivial process to implement a similarity cutoff, below which an "unassignable" result would be returned. Preliminary analysis of the correlation between the lowest similarity values and inputs annotated as "unassignable" indicates that this method could be moderately successful.

The translation task was designed such that participants didn't get access to annotated inputs until after the submission of final results, meaning that parameter settings and fine-tuning of techniques had to be carried out according to intuition only. It thus remains to be determined just what affect the domain-based similarity consolidation, for example, had on the results for the lexical method, and whether alternative formulations would lead to enhanced results. It would also be insightful to implement an equivalent of domain-based similarity consolidation for the structural method.

In conclusion, this paper has served to describe each of a lexical and structural translation retrieval method, as applied to the Senseval2 Japanese translation task. The lexical method modelled strings as a bag of character bigrams, but incorporated a number of novel techniques 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. Of the two proposed methods, the lexical method proved to be clearly superior, although both methods were well above the baseline performance.

Acknowledgements

This research was supported in part by the NTT/Stanford Research Collaboration, research project on multi-word expressions.

References

T. Baldwin. 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).*

T. Baldwin. 2001b. *Making Lexical Sense of Japanese–English Machine Translation: A Disambiguation Extravaganza.* Ph.D. thesis, Tokyo Institute of Technology.

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

S. Ikehara, 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).

S. Kurohashi 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)*, pages 719–24.

S. Kurohashi and M. Nagao. 1998b. *Nihongo keitaikaiseki sisutemu JUMAN* [Japanese morphological analysis system JUMAN] version 3.5. Technical report, Kyoto University. (In Japanese).