Detection of Incorrect Case Assignments in Automatically Generated Paraphrases of Japanese Sentences

Atsushi Fujita **Kentaro Inui** **Yuji Matsumoto**
Graduate School of Information Science,
Nara Institute of Science and Technology
{atsush-f,inui,matsu}@is.aist-nara.ac.jp

Abstract

This paper addresses the issue of correcting transfer errors in paraphrasing. Our previous investigation into transfer errors occurring in lexical and structural paraphrasing of Japanese sentences revealed that case assignment tends to be incorrect, irrespective of the types of transfer (Fujita and Inui, 2003). Motivated by this observation, we propose an empirical method to detect incorrect case assignment. Our error detection model combines two error detection models. They are separately trained on a large collection of positive examples and a small collection of manually labeled negative examples. Experimental results show that our combined model significantly enhances the baseline model which is trained only on positive examples. We also propose a selective sampling scheme to reduce the cost of collecting negative examples, and confirm the effectiveness for the error detection task.

1 Introduction

Recently, automatic paraphrasing has been attracting increasing attention due to its potential in a wide range of natural language processing application (NLPRS, 2001; ACL, 2003). For example, paraphrasing has been applied to pre-editing and post-editing in machine translation (Shirai et al., 1993), query expansion for question answering (Ravichandran and Hovy, 2002), and reading assistance (Carroll et al., 1999; Inui et al., 2003).

There are various levels of *lexical and structural paraphrasing* as the following examples demonstrate[1]:

(1) s. He *accomplished* the mission perfectly.
t. He *achieved* the mission perfectly.

(2) s. *It was* a Honda *that* John *sold to* Tom.
t. John *sold* a Honda *to* Tom.

In automating such paraphrasing, the difficulty of specifying the applicability conditions of each paraphrasing pattern is one of the major problems. For example, it is not easy to specify under what conditions "accomplish" can be paraphrased into "achieve". Paraphrasing patterns with wrong applicability conditions would produce various types of erroneous paraphrases from input, which we call *transfer errors*. We thus need to develop a robust method to detect and correct transfer errors in the post-transfer process by way of a safety net.

Our previous investigation revealed that case assignment tends to be a major error source in paraphrasing of Japanese sentences (Fujita and Inui, 2003). Here is an example of incorrect case assignment: applying the *paraphrasing rule* "accomplish $\Rightarrow$ achieve" (cf. (1)) to sentence (3s) generates (3t). But (3t) is incorrect, because the word "achieve" requires the words, such as "aim", "record" and "success", for its direct object.

(3) s. He *accomplished* the journey in an hour.
t.*He *achieved* the journey in an hour.

One may suspect that incorrect case assignment can be detected simply by referring to a handcrafted case frame dictionary which describes allowable cases and their selectional restrictions for

[1]For each example, s denotes an input and t denotes its paraphrase. Note that our target language is Japanese. English examples are used here for an explanatory purpose.

each verb. However, in existing case frame dictionaries of Japanese, selectional restrictions are generally specified based on coarse-grained semantic classes of noun. It is therefore not adequate for the purpose of the detection of incorrect case assignments (the detail will be given in Section 2).

To capture even the difference between the usages of near-synonyms, we deal with words directly instead of relying on their semantic classes. Since a considerably large number of positive examples can be collected from existing corpora, one can construct a statistical language model and apply it to the error detection task. In this paper, to enhance such a statistical language model, we introduce the use of negative examples and address the following two issues:

1. Unlike positive examples, negative examples are generally not available. A challenging issue is therefore how to effectively use a limited number of manually collected negative examples combining with a large number of positive examples.
2. Manual collection of negative examples is costly and time-consuming. Moreover, any such collection is sparse in the combinatorial space of words. Hence, we need an effective labeling scheme to collect negative examples that truly contribute to error detection.

2 Incorrect case assignment

2.1 Frequency

In (Fujita and Inui, 2003), we investigated transfer errors in Japanese from two points of view: (i) what types of errors occur in performing lexical and structural paraphrasing of Japanese sentences, and (ii) which of them tend to be serious problem. We implemented about 28,000 paraphrasing rules[2] consisting of various levels of lexical and structural paraphrasing, and analyzed 630 automatically generated sentences.

An important observation in (Fujita and Inui, 2003) is that case assignment tends to be incorrect, irrespective of the types of paraphrasing. A quarter of the paraphrased sentences (162 / 630) exhibit this type of errors. This ratio indicates the second most frequent errors, whereas the most dominant type is inappropriate conjugation forms of verbs and adjectives (303 / 630)[3], which can be easily corrected by revising the conjugation forms.

2.2 Causes of errors

At least in Japanese, case assignment can be incorrect at three different levels:

(i) Violation of syntactic constraints: Though both of the verbs "*tessuru*" and "*tsuranuku*" have the same meaning "devote" in the context of example (4), the paraphrased sentence (4t) is incorrect because "*tsuranuku*" cannot take the "*ni* (dative)" case.

(4) s. *Team play-ni* *tessuru*.
team play-DAT devote-PRES
He devotes himself to team play.

t.* *Team play-ni* *tsuranuku*.
team play-DAT devote-PRES

(ii) Violation of selectional restrictions: The verb "*katameru* (strengthen)" requires a concrete object for its "*o* (accusative)" case. Since the noun "*kontei* (basis)" in the paraphrased sentence (5t) does not satisfy the constraint, (5t) is incorrect.

(5) s. *Building-no* *kiban-o* *katameta*.
building-GEN foundation-ACC strengthen-PAST
He strengthened the foundation of the building.

t.* *Building-no* *kontei-o* *katameta*.
building-GEN basis-ACC strengthen-PAST
* He strengthened the basis of the building.

(iii) Semantic inconsistency between sibling cases: The nouns "*hyogen* (expressions)" and "*kakuchi* (every land)" satisfy the semantic constraint for "*ga* (nominative)" and "*ni* (locative)" cases of the verb "*aru* (exist)", respectively. Nevertheless, the paraphrased sentence (6t) is incorrect, because the meanings described by the sibling cases are semantically inconsistent.

(6) s. *Nankai-na* *hyogen-ga*
crabbed-ADJ expressions-NOM
zuisho-ni *aru*.
many places-LOC exist-PRES
There are crabbed expressions in many places.

[2] http://cl.aist-nara.ac.jp/lab/kura/KuraData/

[3] The third most frequent error was incorrect functional word connections that occurred in 78 sentences. The other errors occurred in less than 40 sentences.

t.*Nankai-na hyogen-ga*
crabbed-ADJ expressions-NOM
kakuchi-ni aru.
every land-LOC exist-PRES
*There are crabbed expressions in every land.

2.3 Task setting

Supposing that the case assignments in input sentences into paraphrasing are all correct, the target of error detection is to detect anomalies yielded in the paraphrased case structures that consist of a verb, case particles, and case fillers (nouns). To handle them, we assume dependency structures due to the following reasons:

- For english, linear structure-based statistics, e.g., n-grams can predict human plausibility judgments, namely, correctness (Lapata et al., 2001; Keller et al., 2002). However, there is no guarantee that they perform well in Japanese, because word ordering in Japanese is relatively unrestricted compared with English.
- Most of the paraphrasing systems for Japanese deal with dependency structures (Kondo et al., 2001; Takahashi et al., 2001; Kaji et al., 2002). That is, such a system generates paraphrases annotated with a dependency structure, whatever transfer error occurs.

As mentioned in Section 1, existing case frame dictionaries tend to specify selectional restrictions relying on a coarse-grained semantic typology. For example, the difference between two near-synonyms "*kiban* (foundation)" and "*kontei* (basis)" is crucial in the context of example (5), but most of the dictionaries do not distinguish them, classifying them as the same semantic class "basis". Such a dictionary is not adequate for detection of incorrect case assignment.

Instead, we deal with words directly. Let v, n and c be a verb, a noun and the case particle which connects v and n, respectively. We reduce the error detection task into the classification of triplet $\langle v, c, n\rangle$ into *correct* or *incorrect*. A given paraphrased sentence is judged to be incorrect if and only if any of the triplets included in the sentence is classified as incorrect.

By dealing with $\langle v, c_1, n_1, c_2, n_2\rangle$ to take into account the association between two sibling cases, as in (Torisawa, 2002), we might be able to detect semantic inconsistency as (6t) exhibits. However, considering that the sibling cases could rarely be semantically inconsistent[4] and building a distribution model of $\langle v, c_1, n_1, c_2, n_2\rangle$ is likely to cause a data sparseness problem, we build a error detection model taking only $\langle v, c, n\rangle$ into account.

3 Error detection models

3.1 Issues

The error detection task looks similar to statistical machine translation in the sense that both involve the process of evaluating the appropriateness of given sentences (e.g., (Knight and Chander, 1994)). In statistical machine translation, systems use statistics to compare output candidates. Therefore, what is needed to estimate is *relative likelihood*. For error detection in paraphrasing, however, we need a model for judging the *absolute correctness* of each output candidate for the following reason. Paraphrasing systems are developed typically for a particular purpose such as simplifying text and controlling wording. In such systems, the variety of paraphrasing rules tends to be restricted; so the rule set sometimes produces no appropriate paraphrase candidate for a given input sentence. An error detection model therefore needs the ability to not only compare candidates but also give up producing output when none of the candidates is correct.

If error detection is defined as the task of classifying the candidates as *correct* or *incorrect*, one may want to use both positive and negative examples to train a classifier. However, positive and negative examples are significantly imbalanced, and any collection of negative examples is likely to be too small to represent the distribution of the negative class. Therefore, it is probably not a good choice to input them into a single classifier induction algorithm such as support vector machines.

Instead, we separately train two models, the positive model (*Pos*) and the negative model (*Neg*) as illustrated in Figure 1, then combine them to create another model (*Com*). Since negative examples have to be collected by hand, we also investigate the effectiveness of a selective

[4]According to the analysis in (Fujita and Inui, 2003), only 8 cases of the 162 incorrect case assignments had semantically inconsistent sibling cases.

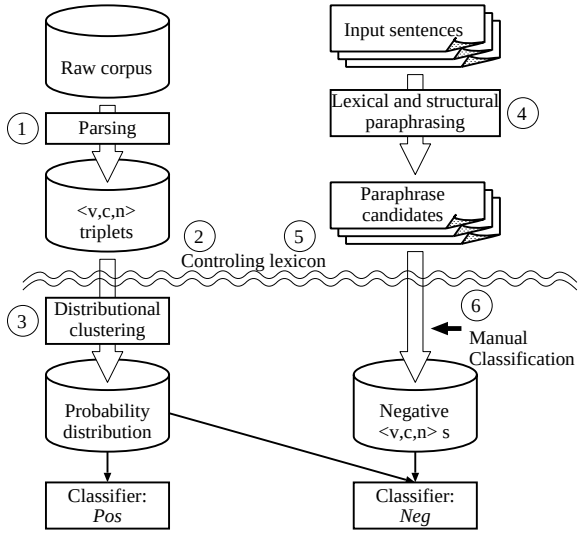

Figure 1: Model construction scheme.

sampling scheme to reduce human labor.

The rest of this section elaborates on our error detection model and selective sampling scheme.

3.2 Combining separately trained models

3.2.1 Positive model

Since a considerably large number of positive examples can be collected from existing corpora using a parser, one can estimate the probability $P(\langle v, c, n\rangle)$ with reasonable accuracy. On that account, we first construct a baseline model *Pos*, a statistical language model trained only on the positive examples.

To calculate $P(\langle v, c, n\rangle)$ avoiding the data sparseness problem, one can use *Probabilistic Latent Semantic Indexing* (*PLSI*) (Hofmann, 1999) which bases itself on *distributional clustering* (Pereira et al., 1993). *PLSI* is a maximum likelihood estimation method. Dividing[5] $\langle v, c, n\rangle$ into $\langle v, c\rangle$ and n, one can estimate $P(\langle v, c, n\rangle)$ by:

$$P(\langle v, c, n\rangle) \;=\; \sum_{z \in Z} P(\langle v, c\rangle|z) P(n|z) P(z),$$

where Z denotes a set of *latent classes* of co-occurrence, and probabilistic parameters $P(\langle v, c\rangle|z)$, $P(n|z)$, and $P(z)$ can be estimated by the EM algorithm.

Given $P(\langle v, c, n\rangle)$, we can use various co-occurrence measures to estimate the likelihood of a given pair of $\langle v, c\rangle$ and n. Well-known options are $P(\langle v, c, n\rangle)$ (*Prob*), *mutual information* (*MI*), and the *Dice coefficient* (*Dice*).

[5] $P(\langle v, c, n\rangle)$ can be represented by the product of $P(\langle v, c\rangle)$ and $P(n|\langle v, c\rangle)$. Both of the marginal distributions corresponds existing linguistic concept; the former indicates the likelihood of a case structure, while the latter does the satisfaction degree of semantic constraint.

3.2.2 Negative model

Pos might not be able to properly judge the *correctness* of $\langle v, c, n\rangle$ by setting a simple threshold, particularly in cases where $P(\langle v, c\rangle)$ or $P(n)$ is low. This defect is expected to be compensated for by the use of negative examples. However, we cannot incorporate negative examples into the statistical language model directly. We thus construct a negative model *Neg* separately from *Pos*.

One simple way of using negative examples is the k-nearest neighbor (k-NN) averaging method. Assuming that the distance between an input triplet $\langle v, c, n\rangle$ and a labeled negative example $\langle v', c', n'\rangle$ depends on both the distance between $\langle v, c\rangle$ and $\langle v', c'\rangle$ and the distance between n and n', we formulate the following distance function:

$$Dist(\langle v, c, n\rangle, \langle v', c', n'\rangle) = DS\Big(P(Z|n), P(Z|n')\Big) + DS\Big(P(Z|\langle v, c\rangle), P(Z|\langle v', c'\rangle)\Big),$$

Here, $P(Z|\langle v, c\rangle)$ and $P(Z|n)$ are the feature vectors for $\langle v, c\rangle$ and n. These probability distributions are obtained through the EM algorithm for *Pos*, and the function DS denotes *distributional similarity* between two probability distributions. One popular measure of distributional similarity is *Jensen-Shannon divergence* (DS_{JS}), which is examined in (Lapata et al., 2001; Lee, 2001). Given the pair of probability distributions q and r, DS_{JS} is given by:

$$DS_{JS}(q, r) = \frac{1}{2}\left[D\left(q \,\middle\|\, \frac{q+r}{2}\right) + D\left(r \,\middle\|\, \frac{q+r}{2}\right)\right],$$

where the function D is the *Kullback-Leibler divergence*. Therefore, DS_{JS} is always non-negative, and $DS_{JS} = 0$ iff $q = r$.

Given an input $\langle v, c, n\rangle$, *Neg* outputs the weighted average distance $Score_{Neg}$ between the input and its k nearest neighbors as the score indicating the degree of correctness. Formally,

$$Score_{Neg} = \frac{1}{k}\sum_{i=1}^{k} \lambda_i \, Dist\left(\langle v, c, n\rangle, \langle v', c', n'\rangle_i\right),$$

where λ_i is the weight for $\langle v', c', n'\rangle_i$, the i-th nearest neighbor.

3.2.3 Combined model

Given the pair of scores output by *Pos* and *Neg*, our error detection model *Com* converts them into normalized confidence values C_{Pos} and C_{Neg} ($0 \leq C_{Pos}, C_{Neg} \leq 1$). Each normalization function can be derived using development data (see Section 4). *Com* then outputs the weighted average of C_{Pos} and C_{Neg} as the overall score:

$$Score_{Com} \quad = \quad \beta\, C_{Pos} + (1 - \beta)\, C_{Neg},$$

where $0 \leq \beta \leq 1$ determines the weights of the models. $Score_{Com}$ indicates the degree of correctness.

3.3 Selective sampling of negative data

We need negative examples that are expected to be useful in improving *Neg* and *Com*. For the current purpose, an example is not useful if it is positive. An example is not useful, either, if it is similar to any of the known negative examples. In other words, we prefer negative examples that are not similar to any existing labeled negative example. We henceforth refer to unlabeled instances as *samples*, and labeled ones as *examples*.

Our strategy for selecting samples can be implemented straightforwardly. We use *Pos* to estimate how likely a sample is negative. To compute the similarity between an unlabeled sample and labeled examples, we use *Neg*. Let p_x be the estimated probability of an unlabeled sample x, and $s_x (> 0)$ be the similarity between x and its nearest negative example. The preference for a given sample x is given by, e.g., $Pref(x) = -s_x \log(p_x)$, which we use in the experiments below.

Our selective sampling scheme is as follows:

1. Generate a set of paraphrases by applying paraphrasing rules to sentences sampled from documents in a given target domain.
2. Extract a set of triplets from the set of paraphrases. We call it a *sample pool*.
3. Sample a small number of triplets randomly from the sample pool, and label them manually. Use only negative samples as the seed of the negative example set for *Neg*.
4. For each sample in the sample pool, calculate its preference by $Pref$ given above.
5. Select the most preferred sample, and label it manually. If it is negative, add it into the negative example set.
6. Repeat Steps 4 and 5 until a certain stopping condition is satisfied (for example, the performance for development data is converged).

4 Experiments

4.1 Data

We trained *Pos* and *Neg* in the following way (Also see Figure 1). During this process, paraphrase candidates were constructed for evaluation as well.

1. 53 million tokens (8.0 million types) of triplets $\langle v, c, n \rangle$ were collected from the parsed[6] sentences of newspaper articles[7].
2. Triplets occurring only once were filtered out. To handle case alteration properly, we dealt with active and passive forms of verbs separately. We restricted c to be the most frequent seven case particles: "*ga* (NOM)", "*o* (ACC)", "*ni* (DAT)", "*de* (LOC)", "*e* (*to*)", "*kara* (*from*)", and "*yori* (*from / than*)". This procedure resulted in 3.1 million types of triplets consisting of 38,512 types of n and 66,484 of $\langle v, c \rangle$.
3. We estimated the probabilistic parameters of *PLSI* by applying the EM algorithm[8] to the data, changing the number of latent classes $|Z|$ from 2 through 1,500.
4. To develop a negative example set, we excerpted 90,000 sentences from the newspaper articles used in Step 1, input them into a paraphrasing system for Japanese[9], and obtained 7,167 paraphrase candidates by applying the same paraphrasing rules that were used for our previous investigation into transfer errors (Fujita and Inui, 2003).
5. We filtered out the generated candidates that contain no changed case structure and those that include either v or n with a frequency of less than 2,000 in the collection given in Step 1. As a result, 3,166 candidates remained.

[6] We used the statistical Japanese dependency parser CaboCha (Kudo and Matsumoto, 2002) for parsing. http://cl.aist-nara.ac.jp/~taku-ku/software/cabocha/

[7] Extracts from 9 years of the Mainichi Shinbun and 10 years of the Nihon Keizai Shinbun consisting of 25,061,504 sentences are used.

[8] http://cl.aist-nara.ac.jp/~taku-ku/software/plsi/

[9] We used KURA (Takahashi et al., 2001). http://cl.aist-nara.ac.jp/lab/kura/doc/

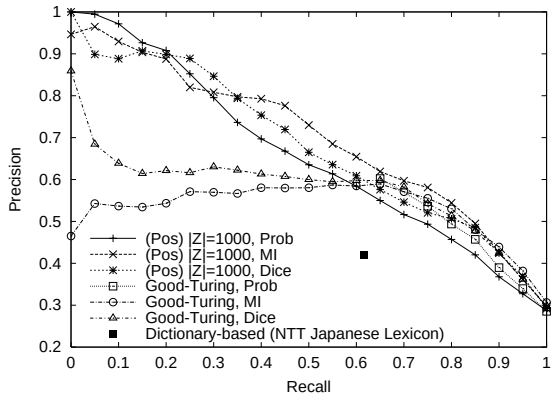

Figure 2: R-P curves of baseline models.

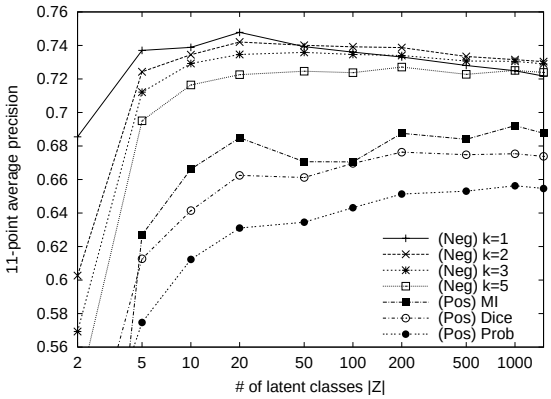

Figure 3: 11-point precision of models over $|Z|$.

6. Finally, we manually labeled the 3,166 candidates and their triplets. We obtained (i) 2,358 positive and 808 (25.5%) negative candidates[10], and (ii) 3,704 types of triplets consisting of 2,853 positive and 851 negative. The former set was used for evaluation, while the latter was used for training *Neg*.

4.2 Evaluation measures

For evaluation, we compared the performance of *Pos*, *Neg*, and *Com*. For each model, we set a threshold and used it so that a given input was classified as erroneous if and only if it received a lower score than the threshold. Given such a threshold, recall R and precision P of a model are defined as follows:

$$R = \frac{\text{\# of correctly detected erroneous candidates}}{\text{\# of erroneous candidates}},$$

$$P = \frac{\text{\# of correctly detected erroneous candidates}}{\text{\# of candidates the model classified as erroneous}}.$$

While we could estimate the optimal threshold for each model, in the experiments, we plotted recall-precision (R-P) curves by varying the threshold. To summarize a R-P curve, we used 11-point average precision (*11-point precision*, hereafter) where the eleven points are $R = 0.0, 0.1, \ldots, 1.0$. To compare R-P curves, we conducted *Wilcoxon rank-sum test* using precision at eleven point above, assuming $p < 0.05$ as the significancy level.

4.3 Results

4.3.1 Baseline

First, to illustrate the complexity of the task, we show the performance of the baseline models: a dictionary-based model, a word-based naive smoothing model, and our statistical language model *Pos*. We regard *Pos* as a baseline because our concern is to what extent *Pos* can be enhanced by introducing *Neg* and *Com*. For the case frame dictionary, we used the largest Japanese case frame dictionary, the NTT Japanese Lexicon (Ikehara et al., 1997) (*Dic*), and the Good-Turing estimation (*GT*) for the naive smoothing model.

As shown in Figure 2, *Pos* significantly outperforms both *Dic* and *GT*. *Prob*, *MI* and *Dice* with $|Z| = 1{,}000$ achieve 65.6%, 69.2% and 67.5% 11-point precision, while *Dic* achieves 41.9% precision under 61.6% recall[11], and *MI* and *Dice* based on *GT* achieve 51.9% and 58.0% 11-point precision[12]. Regarding *Pos*, *Prob* outperforms *MI* and *Dice* for lower recall, while *MI* and *Dice* outperform *Prob* for higher recall. But there is no significant difference among them.

The classification performance of *Pos* is shown over the number of latent classes $|Z|$ in Figure 3. The larger $|Z|$ achieves higher 11-point precision. However, overly enlarging $|Z|$ will presumably not work well since the performance of *Pos* hits a ceiling. Since the optimal $|Z|$ relies on the lexicon, we need to estimate it for a given lexicon using development data. However, since the performance distribution over $|Z|$ is so moderate, we can optimize $|Z|$ with a reasonable cost.

[10] 41 out of 808 were incorrect due to semantic inconsistency between sibling cases.

[11] *Dic* classifies a given $\langle v, c, n \rangle$ as correct or not if and only if both v and n is described in the dictionary. In our experiment, since 338 paraphrase candidates (10.7%) are not judged, we calculated recall and precision using judged 2,828 candidates.

[12] Notice that *Prob* based on *GT* does not perform for a lower recall ($R \leq 0.66$, in our experiment) because it does not distinguish the triplets that have the same frequency.

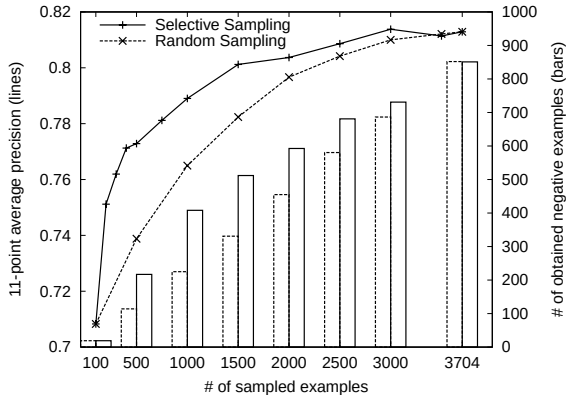

Figure 4: Learning curves of *Com*: curves: 11-point precision, bars: # of obtained negative examples.

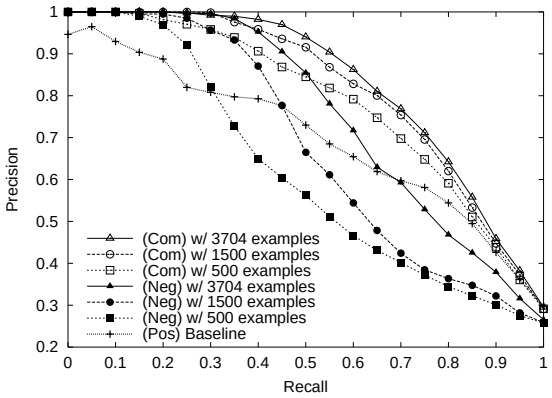

Figure 5: R-P curves of our models.

4.3.2 Properties of negative model

Neg was evaluated by conducting 5-fold cross-validation over the labeled negative examples to keep training and test data exclusive. The weighting function λ_i for i-th nearest neighbor is set to $1/i$, the reciprocal of the similarity rank. The 11-point precision for combinations of parameters are shown in Figure 3. In contrast to *Pos*, *Neg* achieves the best with small $|Z|$. This is good news because a larger number of $|Z|$ obliges a higher computation cost for calculating each distance. With regard to the number of consulting neighbors k, the 11-point precision peaks at $k = 1$. We speculate that the combinatorial space is so large that a larger k causes more noise. Hence, we can conclude that $k = 1$ is enough for this task.

The performance of *Neg* may seem too high given the number of negative examples we used. However, it is not necessarily unlikely. Recall that the set of paraphrasing rules we used was built for the purpose of text simplification. In such a case, presumably the variety of triplets involved in generated paraphrases is relatively small. Therefore, a limited number of negative examples suffices to cover the negative classes. This is expected to be a common property in applied paraphrasing systems as mentioned in Section 3.1.

4.3.3 Combining models with selectively sampled examples

To evaluate the effectiveness of (a) combining *Pos* with *Neg* and (b) selective sampling, we conducted simulations using the 3,704 types of labeled triplets.

We first randomly sampled two sets of 100 samples from 3,704 labeled triplets. One involved 16 negative examples and the other 22. We used these two different sets of negative examples as seeds of the negative example set. We then conducted the selective sampling scheme for each seed, regarding the remaining 3,604 triplets as the sample pool. Parameters and metrics employed are *Prob* and $|Z| = 1{,}000$ for *Pos*, $|Z| = 20$ and $k = 1$ for *Neg*.

In each stage of selective sampling (learning), we formed a combined model *Com*, employing the parameters and metrics on which each component model performed best, i.e., *MI* and $|Z| = 1{,}000$ for *Pos*, and $|Z| = 20$ and $k = 1$ for *Neg*. Combining ratio β was set to 0.5 just for averaging. We then evaluated *Com* by conducting 5-fold cross-validations as well as for *Neg*.

Figure 4 compares the performance of selective and random sampling, showing the averaged results for two seeds. In the figure, the horizontal axis denotes the number of sampled examples. The bars in the figure denote the number of obtained negative examples, showing that preference function efficiently selects negative examples compared to random sampling. The curves in the figure denote the performance curves which show a remarkable advantage of selective sampling, particularly in the early stage of learning.

Figure 5 shows the R-P curves of *Pos*, *Neg*, and *Com*. *Com* surpasses *Pos* and *Neg* over all ranges of recall. One can see that the models based on selective sampling exhibit R-P curves as nicely as the model with the largest negative example set. It is therefore confirmed that even if the collection of negative examples are not sufficient to represent the distribution of the negative classes, we can enhance the baseline model *Pos*

by combining it with *Neg*. With the largest negative examples, *Com* achieved 81.3% 11-point precision, a 12.1 point improvement over *Pos*. Concerning the optimal β which depends on the set of negative examples, it can be easily estimated using development data. For the present settings, the performance peaks when a slightly greater weight is given to *Neg*, i.e., $\beta = 0.45$. However, there is no significant difference in performance between $\beta = 0.45$ and 0.5. Hence, we can regard 0.5 as the default value for β.

5 Conclusions

We addressed the task of detecting incorrect case assignment, a major error source in paraphrasing of Japanese sentences. Our proposal are: (i) an empirical method to detect incorrect case assignments, where we enhanced a statistical language model by combining it with another model which was trained only on a small collection of negative examples, and (ii) a selective sampling scheme for effective collection of negative examples. Our methods were justified through empirical experiments.

Since our aim is to generate correct paraphrases, we should correct the detected errors. In (Fujita and Inui, 2003), we observed that a small part of incorrect case assignments (22 / 162) could be corrected by replacing case markers with the other ones, while the remaining large part could not be. Furthermore, even if we could correct all incorrect case assignments, other types of errors would still be in the paraphrased sentences. We thus think that coping with various type of errors is more important. The errors discussed in this paper appear on relatively shallow levels of syntax and semantics. Our next challenge will go to a deeper level such as the verification of whether meaning is preserved or not.

References

ACL. 2003. *The 2nd International Workshop on Paraphrasing: Paraphrase Acquisition and Applications* (*IWP*).

J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J. Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics* (*EACL*), pages 269–270.

A. Fujita and K. Inui. 2003. Exploring transfer errors in lexical and structural paraphrasing. *Journal of Information Processing Society of Japan*, 44(11):2826–2838. (in Japanese).

T. Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (*SIGIR*), pages 50–57.

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

K. Inui, A. Fujita, T. Takahashi, R. Iida, and T. Iwakura. 2003. Text simplification for reading assistance: a project note. In *Proceedings of the 2nd International Workshop on Paraphrasing: Paraphrase Acquisition and Applications* (*IWP*), pages 9–16.

N. Kaji, D. Kawahara, S. Kurohashi, and S. Sato. 2002. Verb paraphrase based on case frame alignment. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 215–222.

F. Keller, M. Lapata, and O. Ourioupina. 2002. Using the Web to overcome data sparseness. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 230–237.

K. Knight and I. Chander. 1994. Automated postediting of documents. In *Proceedings of the 12th National Conference on Artificial Intelligence* (*AAAI*), pages 779–784.

K. Kondo, S. Sato, and M. Okumura. 2001. Paraphrasing by case alternation. *Journal of Information Processing Society of Japan*, 42(3):465–477. (in Japanese).

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of 6th Conference on Natural Language Learning* (*CoNLL*), pages 63–69.

M. Lapata, F. Keller, and S. McDonald. 2001. Evaluating smoothing algorithms against plausibility judgements. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 346–353.

L. Lee. 2001. On the effectiveness of the skew divergence for statistical language analysis. In *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics*, pages 65–72.

NLPRS. 2001. *Workshop on Automatic Paraphrasing: Theories and Applications*.

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 183–190.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 215–222.

S. Shirai, S. Ikehara, and T. Kawaoka. 1993. Effects of automatic rewriting of source language within a Japanese to English MT system. In *Proceedings of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation* (*TMI*), pages 226–239.

T. Takahashi, T. Iwakura, R. Iida, A. Fujita, and K. Inui. 2001. KURA: a transfer-based lexico-structural paraphrasing engine. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium* (*NLPRS*) *Workshop on Automatic Paraphrasing: Theories and Applications*, pages 37–46.

K. Torisawa. 2002. An unsupervised learning method for associative relationships between verb phrases. In *Proceedings of the 19th International Conference on Computational Linguistics* (*COLING*), pages 1009–1015.