Detection of Incorrect Case Assignments in Paraphrase Generation

Atsushi Fujita, Kentaro Inui, and Yuji Matsumoto

Graduate School of Information Science,
Nara Institute of Science and Technology
{atsush-f,inui,matsu}@is.naist.jp

Abstract. This paper addresses the issue of post-transfer process in paraphrasing. Our previous investigation into transfer errors revealed that case assignment tends to be incorrect, irrespective of the types of transfer in lexical and structural paraphrasing of Japanese sentences [3]. Motivated by this observation, we propose an empirical method to detect incorrect case assignments. Our error detection model combines two error detection models that 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 in 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 [11, 1]. For example, paraphrasing has been applied to pre-editing and post-editing in machine translation [14], query expansion for question answering [13], and reading assistance [2, 6].

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.

[1] For each example, 's' denotes an input and 't' denotes its paraphrase. A sentence with the mark '$*$' indicates it is incorrect. Note that our target language is Japanese. English examples are used here for an explanatory purpose.

Our previous investigation revealed that case assignment tends to be a major error source in paraphrasing of Japanese sentences [3]. 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 each verb. However, in existing case frame dictionaries of Japanese, selectional restrictions are generally specified based on coarse-grained semantic classes of noun. They are therefore not adequate for the purpose of the detection of incorrect case assignments (the detail will be given in Section 2).

To capture 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, namely, correct examples of case assignments, can be collected from existing corpora, one can construct a statistical language model and apply it to the error detection task [9, 7]. In this paper, to enhance such a statistical language model, we introduce the use of negative examples and address the following 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 way to collect negative examples that truly contribute to error detection.

2 Incorrect case assignment

2.1 Characteristics

In [3], we investigated transfer errors in Japanese from two points of view: (i) what types of errors occur in 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. Through the investigation, we observed that case assignment tended to be incorrect, irrespective of the types of paraphrasing. A quarter of the paraphrased sentences (162 / 630) involved this type of errors. This ratio indicated the second most frequent errors[3].

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", the paraphrased sentence (4t) is incorrect because "*tsuranuku*" cannot take the "*ni* (dative)" case.

[2] http://cl.naist.jp/lab/kura/KuraData/

[3] The most dominant type was inappropriate conjugation forms of verbs and adjectives (303 / 630), which could be easily corrected by changing their conjugation forms. The third most frequent error was incorrect functional word connections that occurred in 78 sentences. The other errors occurred in less than 40 sentences.

(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 constraint, (5t) becomes 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)" in the paraphrased sentence (6t) satisfy the semantic constraint for "*ga* (nominative)" and "*ni* (locative)" cases of the verb "*aru* (exist)," respectively. Nevertheless, (6t) is incorrect, because of semantic discrepancy between the nouns of the nominative and locative cases.

(6) s.

Nankai-na	*hyogen-ga*	*zuisho-ni*	*aru.*
crabbed-ADJ	expressions-NOM	many places-LOC	exist-PRES

There are crabbed expressions in many places (of the document).

t.

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

*There are crabbed expressions in every land.

2.2 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).

As mentioned in Section 1, existing case frame dictionaries specify selectional restrictions relying on a coarse-grained semantic typology. For example, most of the dictionaries do not distinguish two near-synonyms "*kiban* (foundation)" and "*kontei* (basis)", and classifies them into the same semantic class "basis," although the difference between them is crucial in the context of example (5).

Instead, we deal with words directly. Let v, n and c be a verb, a noun and the case particle which relates n to v, respectively. We decompose 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. If we deal with $\langle v, c_1, n_1, c_2, n_2\rangle$ to take into account the association between two sibling cases, as in [16], we might be able to detect semantic inconsistency. However, we have so far examined an error detection model taking only $\langle v, c, n\rangle$ into account, because the sibling cases can rarely be semantically inconsistent[4]

[4] According to the analysis in [3], only 8 cases of the 162 incorrect case assignments had semantically inconsistent sibling cases.

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.

3 Error detection models

In generative approaches, such as parsing and statistical machine translation, systems use statistics to estimate *relative likelihood* of output candidates. For the error detection in paraphrasing, however, we need a model for judging the *absolute correctness* of output candidates for the following reason. Paraphrasing systems are typically developed 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 may produce no appropriate paraphrase candidate for a given input sentence. An error detection model therefore needs an ability not only to compare candidates but also to give up producing output when none of the candidates is correct.

If error detection is defined as a discriminative task, i.e. classifying the candidates into *correct* or *incorrect*, one may want to use both positive and negative examples to train a classifier. However, any collection of negative examples is likely to be too small to represent the distribution of the negative class. Thus, it is probably not a good choice to input them together with a vast amount of positive examples into a single classifier induction algorithm such as support vector machines. We therefore separately train two models, the positive model (*Pos*) and the negative model (*Neg*), then combine them to create another model (*Com*) as shown in Figure 1. Since negative examples have to be collected by hand, we also investigate the effectiveness of a selective sampling scheme to reduce human labor.

3.1 Combining separately trained models

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 positive examples.

To calculate $P(\langle v, c, n \rangle)$ avoiding the data sparseness problem, one can use *Probabilistic Latent Semantic Indexing* (*PLSI*) [4] which bases itself on *distributional clustering* [12]. *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.

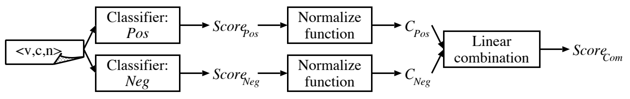

Fig. 1. Proposed model.

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) 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. We employ one of the popular measures of distributional similarity, *Jensen-Shannon divergence* (DS_{JS}) [9, 10]. Given the pair of probability distributions q and r, DS_{JS} is given by:

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

where the function D is the *Kullback-Leibler divergence*. 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. 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, larger value of $Score_{Neg}$ indicates the input is more likely to be correct.

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. *Com* then outputs the weighted average of C_{Pos} and C_{Neg} as the overall score:

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

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

3.2 Selective sampling of negative examples

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, or 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.

Our selective sampling scheme is summarized as follows:

Step 1. Generate a set of paraphrases by applying paraphrasing rules to sentences sampled from documents in a given target domain.

Step 2. Extract a set of triplets from the set of paraphrases. We call it a *sample pool*.

Step 3. Sample a small number of triplets at random from the sample pool, and label them manually. Use only negative samples as the seed of the negative example set.

Step 4. For each sample x in the sample pool, calculate its preference by $Pref(x)$.

Step 5. Select the most preferred sample, and label it manually. If it is negative, add it into the negative example set.

Step 6. Repeat Steps 4 and 5 until a certain stopping condition is satisfied.

4 Experiments

4.1 Data and evaluation measures

We constructed data for training *Pos* and *Neg* in the following way (Also see Figure 2). During this process, paraphrase candidates were constructed for evaluation as well.

Step 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]. To handle case alteration properly, we dealt with active and passive forms of verbs separately.

Step 2. Triplets occurring only once were filtered out. We also 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 $\langle v, c\rangle$.

Step 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.

[6] We used the statistical Japanese dependency parser CaboCha [8] for parsing. http://chasen.naist.jp/~taku/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://chasen.naist.jp/~taku/software/plsi/

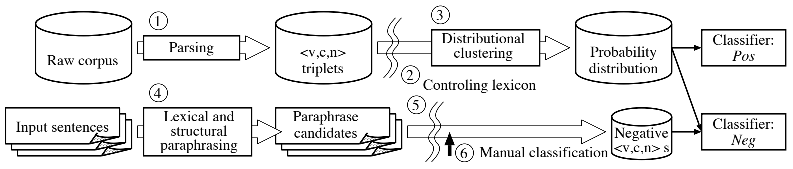

Fig. 2. Model construction scheme.

Step 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 [3].

Step 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. Then, 3,166 candidates remained.

Step 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*.

For evaluation, we compare 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 can be calculated[11]. While we can estimate the optimal threshold for each model, in the experiments, we plot recall-precision (R-P) curves by varying the threshold. To summarize a R-P curve, we use 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 conduct *Wilcoxon rank-sum test* using precision at eleven point above, assuming $p < 0.05$ as the significance level.

4.2 Results

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 [5] (*Dic*), and the Good-Turing estimation (*GT*) for the naive smoothing model.

As shown in Figure 3, *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[12], and *MI* and *Dice* based on *GT* achieve

[9] We used KURA [15]. http://cl.naist.jp/lab/kura/doc/

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

[11] R = # of correctly detected erroneous candidates / # of erroneous candidates,
P = # of correctly detected erroneous candidates / # of candidates classified as incorrect.

[12] *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.

51.9% and 58.0% 11-point precision[13]. Regarding *Pos*, there is no significant difference among the co-occurrence measures.

The performance of *Pos* is shown over the number of latent classes $|Z|$ in Figure 4. The larger $|Z|$ achieves higher 11-point precision. However, overly enlarging $|Z|$ presumably does not work well since the performance of *Pos* hits a ceiling. The optimal $|Z|$ relies on the lexicon but the performance distribution over $|Z|$ looks moderate. We therefore expect it can be estimated using development data with a reasonable cost.

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 weight 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 4. In contrast to *Pos*, the performance of *Neg* peaks at small $|Z|$. This is good news because a larger number of $|Z|$ obliges a higher computational cost for calculating each distance. Regarding the number of referring neighbors k, the 11-point precision peaks at $k = 1$. We speculate that the negative examples are so sparse against the combinatorial space 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. It is, however, not necessarily unlikely. We speculate the variety of triplets involved in generated paraphrases is relatively small, because the set of paraphrasing rules we used was build for the purpose of text simplification. Since it is common property in applied paraphrasing systems as mentioned in Section 3, we can expect a limited number of negative examples are sufficient to cover the negative classes.

Combining models with selectively sampled examples Using the 3,704 types of labeled triplets, we evaluated the effectiveness of (a) combining *Pos* with *Neg* and (b) selective sampling. We first sampled at random two sets of 100 samples from 3,704 labeled triplets. One involved 16 negative examples and the other 22. Using for each negative example set, we then simulated the selective sampling scheme, 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. We then evaluated *Com* by conducting 5-fold cross-validations as well as for *Neg*.

Figure 5 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, which denote the number of obtained negative examples, designate that our preference function efficiently selects negative examples. The curves in the figure, which denote the performance curves, designate a remarkable advantage of selective sampling, particularly in the early stage of learning.

Figure 6 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 ex-

[13] 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.

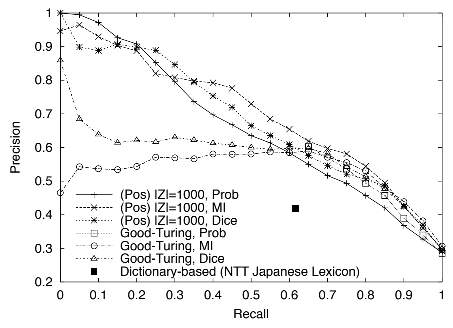

Fig. 3. R-P curves of baseline models.

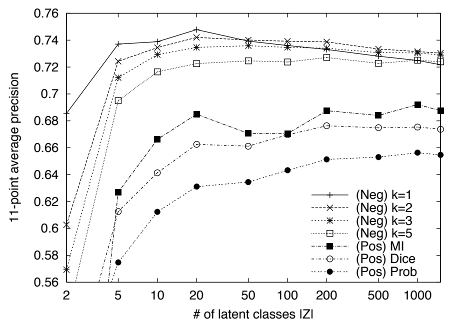

Fig. 4. 11-point precision of models over $|Z|$.

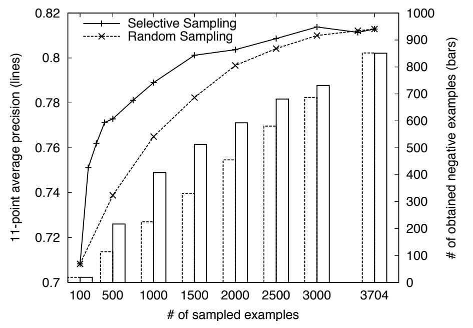

Fig. 5. Learning curves of *Com*. Lines: 11-point average precision, bars: # of obtained negative examples.

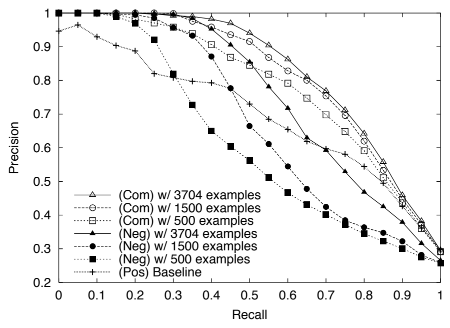

Fig. 6. R-P curves of our models.

hibit 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 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, we can use $\beta = 0.5$ as default, because there is no significant difference in performance between $\beta = 0.45$ and $\beta = 0.5$.

5 Conclusions

We presented 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, correcting the detected errors is another important issue. In [3], however, we observed that a only small part of incorrect case assignments (22 / 162) could be corrected by replacing the case markers, while the remaining large part could not be. Moreover, even if we could correct all incorrect case assignments, other types of frequent errors would still be in the paraphrased sentences. We thus think that coping with various type of errors should be given a preference.

References

1. ACL. *The 2nd International Workshop on Paraphrasing: Paraphrase Acquisition and Applications* (*IWP*), 2003.
2. J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J. Tait. 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, 1999.
3. A. Fujita and K. Inui. Exploring transfer errors in lexical and structural paraphrasing. *Journal of Information Processing Society of Japan*, 44(11):2826–2838, 2003. (in Japanese).
4. T. Hofmann. 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, 1999.
5. S. Ikehara, M. Miyazaki, S. Shirai, A. Yokoo, H. Nakaiwa, K. Ogura, Y. Ooyama, and Y. Hayashi, editors. *Nihongo Goi Taikei – A Japanese Lexicon.* Iwanami Shoten, 1997. (in Japanese).
6. K. Inui, A. Fujita, T. Takahashi, R. Iida, and T. Iwakura. 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, 2003.
7. F. Keller, M. Lapata, and O. Ourioupina. Using the Web to overcome data sparseness. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 230–237, 2002.
8. T. Kudo and Y. Matsumoto. Japanese dependency analysis using cascaded chunking. In *Proceedings of 6th Conference on Natural Language Learning* (*CoNLL*), pages 63–69, 2002.
9. M. Lapata, F. Keller, and S. McDonald. Evaluating smoothing algorithms against plausibility judgements. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 346–353, 2001.
10. L. Lee. 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, 2001.
11. NLPRS. *Workshop on Automatic Paraphrasing: Theories and Applications*, 2001.
12. F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 183–190, 1993.
13. D. Ravichandran and E. Hovy. 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, 2002.
14. S. Shirai, S. Ikehara, and T. Kawaoka. 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, 1993.
15. T. Takahashi, T. Iwakura, R. Iida, A. Fujita, and K. Inui. 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, 2001.
16. K. Torisawa. An unsupervised learning method for associative relationships between verb phrases. In *Proceedings of the 19th International Conference on Computational Linguistics* (*COLING*), pages 1009–1015, 2002.