Application-driven Statistical Paraphrase Generation

Shiqi Zhao, Xiang Lan, Ting Liu, Sheng Li
Information Retrieval Lab, Harbin Institute of Technology
6F Aoxiao Building, No.27 Jiaohua Street, Nangang District
Harbin, 150001, China
{zhaosq,xlan,tliu,lisheng}@ir.hit.edu.cn

Abstract

Paraphrase generation (PG) is important in plenty of NLP applications. However, the research of PG is far from enough. In this paper, we propose a novel method for statistical paraphrase generation (SPG), which can (1) achieve various applications based on a uniform statistical model, and (2) naturally combine multiple resources to enhance the PG performance. In our experiments, we use the proposed method to generate paraphrases for three different applications. The results show that the method can be easily transformed from one application to another and generate valuable and interesting paraphrases.

1 Introduction

Paraphrases are alternative ways that convey the same meaning. There are two main threads in the research of paraphrasing, i.e., paraphrase recognition and paraphrase generation (PG). Paraphrase generation aims to generate a paraphrase for a source sentence in a certain application. PG shows its importance in many areas, such as question expansion in question answering (QA) (Duboue and Chu-Carroll, 2006), text polishing in natural language generation (NLG) (Iordanskaja et al., 1991), text simplification in computer-aided reading (Carroll et al., 1999), and sentence similarity computation in the automatic evaluation of machine translation (MT) (Kauchak and Barzilay, 2006) and summarization (Zhou et al., 2006).

This paper presents a method for statistical paraphrase generation (SPG). As far as we know, this is the first statistical model specially designed for paraphrase generation. It's distinguishing feature is that it achieves various applications with a uniform model. In addition, it exploits multiple resources, including paraphrase phrases, patterns, and collocations, to resolve the data shortage problem and generate more varied paraphrases.

We consider three paraphrase applications in our experiments, including sentence compression, sentence simplification, and sentence similarity computation. The proposed method generates paraphrases for the input sentences in each application. The generated paraphrases are then manually scored based on adequacy, fluency, and usability. The results show that the proposed method is promising, which generates useful paraphrases for the given applications. In addition, comparison experiments show that our method outperforms a conventional SMT-based PG method.

2 Related Work

Conventional methods for paraphrase generation can be classified as follows:

Rule-based methods: Rule-based PG methods build on a set of paraphrase rules or patterns, which are either hand crafted or automatically collected. In the early rule-based PG research, the paraphrase rules are generally manually written (McKeown, 1979; Zong et al., 2001), which is expensive and arduous. Some researchers then tried to automatically extract paraphrase rules (Lin and Pantel, 2001; Barzilay and Lee, 2003; Zhao et al., 2008b), which facilitates the rule-based PG methods. However, it has been shown that the coverage of the paraphrase patterns is not high enough, especially when the used paraphrase patterns are long or complicated (Quirk et al., 2004).

Thesaurus-based methods: The thesaurus-based methods generate a paraphrase **t** for a source sentence **s** by substituting some words in **s** with their synonyms (Bolshakov and Gelbukh, 2004;

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 834–842,
Suntec, Singapore, 2-7 August 2009. ©2009 ACL and AFNLP

Kauchak and Barzilay, 2006). This kind of method usually involves two phases, i.e., candidate extraction and paraphrase validation. In the first phase, it extracts all synonyms from a thesaurus, such as WordNet, for the words to be substituted. In the second phase, it selects an optimal substitute for each given word from the synonyms according to the context in **s**. This kind of method is simple, since the thesaurus synonyms are easy to access. However, it cannot generate other types of paraphrases but only synonym substitution.

NLG-based methods: NLG-based methods (Kozlowski et al., 2003; Power and Scott, 2005) generally involve two stages. In the first one, the source sentence **s** is transformed into its semantic representation **r** by undertaking a series of NLP processing, including morphology analyzing, syntactic parsing, semantic role labeling, etc. In the second stage, a NLG system is employed to generate a sentence **t** from **r**. **s** and **t** are paraphrases as they are both derived from **r**. The NLG-based methods simulate human paraphrasing behavior, i.e., understanding a sentence and presenting the meaning in another way. However, deep analysis of sentences is a big challenge. Moreover, developing a NLG system is also not trivial.

SMT-based methods: SMT-based methods viewed PG as monolingual MT, i.e., translating **s** into **t** that are in the same language. Researchers employ the existing SMT models for PG (Quirk et al., 2004). Similar to typical SMT, a large parallel corpus is needed as training data in the SMT-based PG. However, such data are difficult to acquire compared with the SMT data. Therefore, data shortage becomes the major limitation of the method. To address this problem, we have tried combining multiple resources to improve the SMT-based PG model (Zhao et al., 2008a).

There have been researchers trying to propose uniform PG methods for multiple applications. But they are either rule-based (Murata and Isahara, 2001; Takahashi et al., 2001) or thesaurus-based (Bolshakov and Gelbukh, 2004), thus they have some limitations as stated above. Furthermore, few of them conducted formal experiments to evaluate the proposed methods.

3 Statistical Paraphrase Generation

3.1 Differences between SPG and SMT

Despite the similarity between PG and MT, the statistical model used in SMT cannot be directly applied in SPG, since there are some clear differences between them:

1. SMT has a unique purpose, i.e., producing high-quality translations for the inputs. On the contrary, SPG has distinct purposes in different applications, such as sentence compression, sentence simplification, etc. The usability of the paraphrases have to be assessed in each application.

2. In SMT, words of an input sentence should be totally translated, whereas in SPG, not all words of an input sentence need to be paraphrased. Therefore, a SPG model should be able to decide which part of a sentence needs to be paraphrased.

3. The bilingual parallel data for SMT are easy to collect. In contrast, the monolingual parallel data for SPG are not so common (Quirk et al., 2004). Thus the SPG model should be able to easily combine different resources and thereby solve the data shortage problem (Zhao et al., 2008a).

4. Methods have been proposed for automatic evaluation in MT (e.g., BLEU (Papineni et al., 2002)). The basic idea is that a translation should be scored based on their similarity to the human references. However, they cannot be adopted in SPG. The main reason is that it is more difficult to provide human references in SPG. Lin and Pantel (2001) have demonstrated that the overlapping between the automatically acquired paraphrases and handcrafted ones is very small. Thus the human references cannot properly assess the quality of the generated paraphrases.

3.2 Method Overview

The SPG method proposed in this work contains three components, i.e., sentence preprocessing, paraphrase planning, and paraphrase generation (Figure 1). Sentence preprocessing mainly includes POS tagging and dependency parsing for the input sentences, as POS tags and dependency information are necessary for matching the paraphrase pattern and collocation resources in the following stages. Paraphrase planning (Section 3.3) aims to select the units to be paraphrased (called *source units* henceforth) in an input sentence and the candidate paraphrases for the source

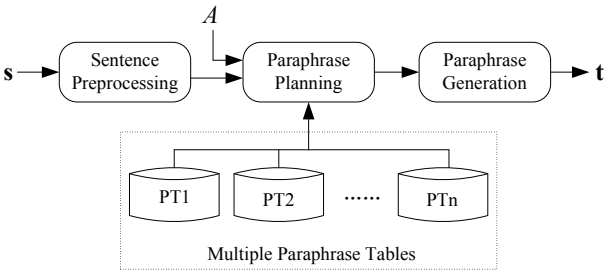

Figure 1: Overview of the proposed SPG method.

units (called *target units*) from multiple resources according to the given application *A*. Paraphrase generation (Section 3.4) is designed to generate paraphrases for the input sentences by selecting the optimal target units with a statistical model.

3.3 Paraphrase Planning

In this work, the multiple paraphrase resources are stored in paraphrase tables (PTs). A paraphrase table is similar to a phrase table in SMT, which contains fine-grained paraphrases, such as paraphrase phrases, patterns, or collocations. The PTs used in this work are constructed using different corpora and different score functions (Section 3.5).

If the applications are not considered, all units of an input sentence that can be paraphrased using the PTs will be extracted as source units. Accordingly, all paraphrases for the source units will be extracted as target units. However, when a certain application is given, only the source and target units that can achieve the application will be kept. We call this process paraphrase planning, which is formally defined as in Figure 2.

An example is depicted in Figure 3. The application in this example is sentence compression. All source and target units are listed below the input sentence, in which the first two source units are phrases, while the third and fourth are a pattern and a collocation, respectively. As can be seen, the first and fourth source units are filtered in paraphrase planning, since none of their paraphrases achieve the application (i.e., shorter in bytes than the source). The second and third source units are kept, but some of their paraphrases are filtered.

3.4 Paraphrase Generation

Our SPG model contains three sub-models: a paraphrase model, a language model, and a usability model, which control the adequacy, fluency,

Input: source sentence **s**
Input: paraphrase application *A*
Input: paraphrase tables *PTs*
Output: set of source units *SU*
Output: set of target units *TU*

Extract source units of **s** from *PTs*: $SU=\{su_1, \ldots, su_n\}$
For each source unit su_i
  Extract its target units $TU_i=\{tu_{i1}, \ldots, tu_{im}\}$
  For each target unit tu_{ij}
    If tu_{ij} cannot achieve the application *A*
      Delete tu_{ij} from TU_i
    End If
  End For
  If TU_i is empty
    Delete su_i from *SU*
  End If
End for

Figure 2: The paraphrase planning algorithm.

and usability of the paraphrases, respectively[1].

Paraphrase Model: Paraphrase generation is a decoding process. The input sentence **s** is first segmented into a sequence of I units $\bar{s}_1^I$, which are then paraphrased to a sequence of units $\bar{t}_1^I$. Let $(\bar{s}_i, \bar{t}_i)$ be a pair of paraphrase units, their paraphrase likelihood is computed using a score function $\phi_{pm}(\bar{s}_i, \bar{t}_i)$. Thus the paraphrase score $p_{pm}(\bar{s}_1^I, \bar{t}_1^I)$ between **s** and **t** is decomposed into:

$$p_{pm}(\bar{s}_1^I, \bar{t}_1^I) = \prod_{i=1}^{I} \phi_{pm}(\bar{s}_i, \bar{t}_i)^{\lambda_{pm}} \quad (1)$$

where λ_{pm} is the weight of the paraphrase model. Actually, it is defined similarly to the translation model in SMT (Koehn et al., 2003).

In practice, the units of a sentence may be paraphrased using different PTs. Suppose we have K PTs, $(\bar{s}_{k_i}, \bar{t}_{k_i})$ is a pair of paraphrase units from the k-th PT with the score function $\phi_k(\bar{s}_{k_i}, \bar{t}_{k_i})$, then Equation (1) can be rewritten as:

$$p_{pm}(\bar{s}_1^I, \bar{t}_1^I) = \prod_{k=1}^{K} (\prod_{k_i} \phi_k(\bar{s}_{k_i}, \bar{t}_{k_i})^{\lambda_k}) \quad (2)$$

where λ_k is the weight for $\phi_k(\bar{s}_{k_i}, \bar{t}_{k_i})$.

Equation (2) assumes that a pair of paraphrase units is from only one paraphrase table. However,

[1]The SPG model applies monotone decoding, which does not contain a reordering sub-model that is often used in SMT. Instead, we use the paraphrase patterns to achieve word reordering in paraphrase generation.

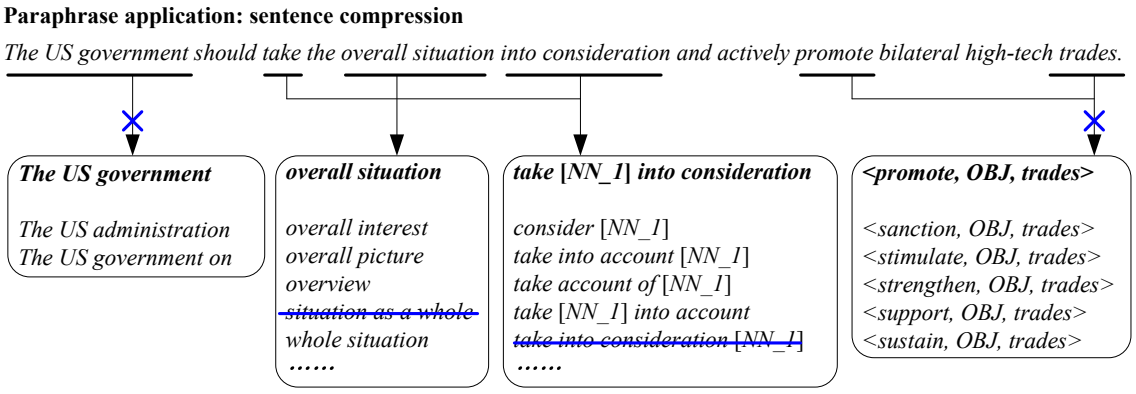

Figure 3: An example of paraphrase planning.

we find that about 2% of the paraphrase units appear in two or more PTs. In this case, we only count the PT that provides the largest paraphrase score, i.e., $\hat{k} = arg\max_k\{\phi_k(\bar{s}_i, \bar{t}_i)^{\lambda_k}\}$.

In addition, note that there may be some units that cannot be paraphrased or prefer to keep unchanged during paraphrasing. Therefore, we have a *self-paraphrase* table in the K PTs, which paraphrases any separate word w into itself with a constant score c: $\phi_{self}(w, w) = c$ (we set $c = e^{-1}$).

Language Model: We use a tri-gram language model in this work. The language model based score for the paraphrase **t** is computed as:

$$p_{lm}(\mathbf{t}) = \prod_{j=1}^{J} p(t_j|t_{j-2}t_{j-1})^{\lambda_{lm}} \quad (3)$$

where J is the length of **t**, t_j is the j-th word of **t**, and λ_{lm} is the weight for the language model.

Usability Model: The usability model prefers paraphrase units that can better achieve the application. The usability of **t** depends on paraphrase units it contains. Hence the usability model $p_{um}(\bar{s}_1^I, \bar{t}_1^I)$ is decomposed into:

$$p_{um}(\bar{s}_1^I, \bar{t}_1^I) = \prod_{i=1}^{I} p_{um}(\bar{s}_i, \bar{t}_i)^{\lambda_{um}} \quad (4)$$

where λ_{um} is the weight for the usability model and $p_{um}(\bar{s}_i, \bar{t}_i)$ is defined as follows:

$$p_{um}(\bar{s}_i, \bar{t}_i) = e^{\mu(\bar{s}_i, \bar{t}_i)} \quad (5)$$

We consider three applications, including sentence compression, simplification, and similarity computation. $\mu(\bar{s}_i, \bar{t}_i)$ is defined separately for each:

- **Sentence compression:** Sentence compression[2] is important for summarization, subtitle generation, and displaying texts in small screens such as cell phones. In this application, only the target units shorter than the sources are kept in paraphrase planning. We define $\mu(\bar{s}_i, \bar{t}_i) = len(\bar{s}_i) - len(\bar{t}_i)$, where $len(\cdot)$ denotes the length of a unit in bytes.

- **Sentence simplification:** Sentence simplification requires using common expressions in sentences so that readers can easily understand the meaning. Therefore, only the target units more frequent than the sources are kept in paraphrase planning. Here, the frequency of a unit is measured using the language model mentioned above[3]. Specifically, the langauge model assigns a score $score_{lm}(\cdot)$ for each unit and the unit with larger score is viewed as more frequent. We define $\mu(\bar{s}_i, \bar{t}_i) = 1$ iff $score_{lm}(\bar{t}_i) > score_{lm}(\bar{s}_i)$.

- **Sentence similarity computation:** Given a reference sentence $\mathbf{s}'$, this application aims to paraphrase **s** into **t**, so that **t** is more similar (closer in wording) with $\mathbf{s}'$ than **s**. This application is important for the automatic evaluation of machine translation and summarization, since we can paraphrase the human translations/summaries to make them more similar to the system outputs, which can refine the accuracy of the evaluation (Kauchak and Barzilay, 2006). For this application,

[2]This work defines compression as the shortening of sentence length in bytes rather than in words.

[3]To compute the language model based score, the matched patterns are instantiated and the matched collocations are connected with words between them.

only the target units that can enhance the similarity to the reference sentence are kept in planning. We define $\mu(\bar{s}_i, \bar{t}_i) = sim(\bar{t}_i, \mathbf{s}') - sim(\bar{s}_i, \mathbf{s}')$, where $sim(\cdot,\cdot)$ is simply computed as the count of overlapping words.

We combine the three sub-models based on a log-linear framework and get the SPG model:

$$p(\mathbf{t}|\mathbf{s}) = \sum_{k=1}^{K}(\lambda_k \sum_{k_i} \log \phi_k(\bar{s}_{k_i}, \bar{t}_{k_i})) + \lambda_{lm} \sum_{j=1}^{J} \log p(t_j|t_{j-2}t_{j-1}) + \lambda_{um} \sum_{i=1}^{I} \mu(\bar{s}_i, \bar{t}_i) \quad (6)$$

3.5 Paraphrase Resources

We use five PTs in this work (except the self-paraphrase table), in which each pair of paraphrase units has a score assigned by the score function of the corresponding method.

Paraphrase phrases (PT-1 to PT-3): Paraphrase phrases are extracted from three corpora: (1) Corp-1: bilingual parallel corpus, (2) Corp-2: monolingual comparable corpus (comparable news articles reporting on the same event), and (3) Corp-3: monolingual parallel corpus (parallel translations of the same foreign novel). The details of the corpora, methods, and score functions are presented in (Zhao et al., 2008a). In our experiments, PT-1 is the largest, which contains 3,041,822 pairs of paraphrase phrases. PT-2 and PT-3 contain 92,358, and 17,668 pairs of paraphrase phrases, respectively.

Paraphrase patterns (PT-4): Paraphrase patterns are also extracted from Corp-1. We applied the approach proposed in (Zhao et al., 2008b). Its basic assumption is that if two English patterns e_1 and e_2 are aligned with the same foreign pattern f, then e_1 and e_2 are possible paraphrases. One can refer to (Zhao et al., 2008b) for the details. PT-4 contains 1,018,371 pairs of paraphrase patterns.

Paraphrase collocations (PT-5): Collocations[4] can cover long distance dependencies in sentences. Thus paraphrase collocations are useful for SPG. We extract collocations from a monolingual corpus and use a binary classifier to recognize if any two collocations are paraphrases. Due to the space limit, we cannot introduce the detail of the approach. We assign the score "1" for any pair of paraphrase collocations. PT-5 contains 238,882 pairs of paraphrase collocations.

[4]A *collocation* is a lexically restricted word pair with a certain syntactic relation. This work only considers verb-object collocations, e.g., <*promote, OBJ, trades*>.

3.6 Parameter Estimation

To estimate parameters $\lambda_k (1 \leq k \leq K)$, λ_{lm}, and λ_{um}, we adopt the approach of minimum error rate training (MERT) that is popular in SMT (Och, 2003). In SMT, however, the optimization objective function in MERT is the MT evaluation criteria, such as BLEU. As we analyzed above, the BLEU-style criteria cannot be adapted in SPG. We therefore introduce a new optimization objective function in this paper. The basic assumption is that a paraphrase should contain as many correct unit replacements as possible. Accordingly, we design the following criteria:

Replacement precision (rp): rp assesses the precision of the unit replacements, which is defined as $rp = c_{dev}(+r)/c_{dev}(r)$, where $c_{dev}(r)$ is the total number of unit replacements in the generated paraphrases on the development set. $c_{dev}(+r)$ is the number of the correct replacements.

Replacement rate (rr): rr measures the paraphrase degree on the development set, i.e., the percentage of words that are paraphrased. We define rr as: $rr = w_{dev}(r)/w_{dev}(s)$, where $w_{dev}(r)$ is the total number of words in the replaced units on the development set, and $w_{dev}(s)$ is the number of words of all sentences on the development set.

Replacement f-measure (rf): We use rf as the optimization objective function in MERT, which is similar to the conventional f-measure and leverages rp and rr: $rf = (2 \times rp \times rr)/(rp + rr)$.

We estimate parameters for each paraphrase application separately. For each application, we first ask two raters to manually label all possible unit replacements on the development set as correct or incorrect, so that rp, rr, and rf can be automatically computed under each set of parameters. The parameters that result in the highest rf on the development set are finally selected.

4 Experimental Setup

Our SPG decoder is developed by remodeling Moses that is widely used in SMT (Hoang and Koehn, 2008). The POS tagger and dependency parser for sentence preprocessing are SVM-

Tool (Gimenez and Marquez, 2004) and MST-Parser (McDonald et al., 2006). The language model is trained using a 9 GB English corpus.

4.1 Experimental Data

Our method is not restricted in domain or sentence style. Thus any sentence can be used in development and test. However, for the sentence similarity computation purpose in our experiments, we want to evaluate if the method can enhance the string-level similarity between two paraphrase sentences. Therefore, for each input sentence **s**, we need a reference sentence $\mathbf{s}'$ for similarity computation.

Based on the above consideration, we acquire experiment data from the human references of the MT evaluation, which provide several human translations for each foreign sentence. In detail, we use the first translation of a foreign sentence as the source **s** and the second translation as the reference $\mathbf{s}'$ for similarity computation. In our experiments, the development set contains 200 sentences and the test set contains 500 sentences, both of which are randomly selected from the human translations of 2008 NIST Open Machine Translation Evaluation: Chinese to English Task.

4.2 Evaluation Metrics

The evaluation metrics for SPG are similar to the human evaluation for MT (Callison-Burch et al., 2007). The generated paraphrases are manually evaluated based on three criteria, i.e., adequacy, fluency, and usability, each of which has three scales from 1 to 3. Here is a brief description of the different scales for the criteria:

Adequacy	1: The meaning is evidently changed.
	2: The meaning is generally preserved.
	3: The meaning is completely preserved.
Fluency	1: The paraphrase **t** is incomprehensible.
	2: **t** is comprehensible.
	3: **t** is a flawless sentence.
Usability	1: **t** is opposite to the application purpose.
	2: **t** does not achieve the application.
	3: **t** achieves the application.

5 Results and Analysis

We use our method to generate paraphrases for the three applications. Results show that the percentages of test sentences that can be paraphrased are 97.2%, 95.4%, and 56.8% for the applications of sentence compression, simplification, and similarity computation, respectively. The reason why the last percentage is much lower than the first two is that, for sentence similarity computation, many sentences cannot find unit replacements from the PTs that improve the similarity to the reference sentences. For the other applications, only some very short sentences cannot be paraphrased.

Further results show that the average number of unit replacements in each sentence is 5.36, 4.47, and 1.87 for sentence compression, simplification, and similarity computation. It also indicates that sentence similarity computation is more difficult than the other two applications.

5.1 Evaluation of the Proposed Method

We ask two raters to label the paraphrases based on the criteria defined in Section 4.2. The labeling results are shown in the upper part of Table 1. We can see that for adequacy and fluency, the paraphrases in sentence similarity computation get the highest scores. About 70% of the paraphrases are labeled "3". This is because in sentence similarity computation, only the target units appearing in the reference sentences are kept in paraphrase planning. This constraint filters most of the noise. The adequacy and fluency scores of the other two applications are not high. The percentages of label "3" are around 30%. The main reason is that the average numbers of unit replacements for these two applications are much larger than sentence similarity computation. It is thus more likely to bring in incorrect unit replacements, which influence the quality of the generated paraphrases.

The usability is needed to be manually labeled only for sentence simplification, since it can be automatically labeled in the other two applications. As shown in Table 1, for sentence simplification, most paraphrases are labeled "2" in usability, while merely less than 20% are labeled "3". We conjecture that it is because the raters are not sensitive to the slight change of the simplification degree. Thus they labeled "2" in most cases.

We compute the kappa statistic between the raters. Kappa is defined as $K = \frac{P(A)-P(E)}{1-P(E)}$ (Carletta, 1996), where $P(A)$ is the proportion of times that the labels agree, and $P(E)$ is the proportion of times that they may agree by chance. We define $P(E) = \frac{1}{3}$, as the labeling is based on three point scales. The results show that the kappa statistics for adequacy and fluency are 0.6560 and 0.6500, which indicates a substantial agreement (K: 0.61-0.8) according to (Landis and Koch, 1977). The

		Adequacy (%)			Fluency (%)			Usability (%)		
		1	2	3	1	2	3	1	2	3
Sentence compression	rater1	32.92	44.44	22.63	21.60	47.53	30.86	0	0	100
	rater2	40.54	34.98	24.49	25.51	43.83	30.66	0	0	100
Sentence simplification	rater1	29.77	44.03	26.21	22.01	42.77	35.22	25.37	61.84	12.79
	rater2	33.33	35.43	31.24	24.32	39.83	35.85	30.19	51.99	17.82
Sentence similarity	rater1	7.75	24.30	67.96	7.75	22.54	69.72	0	0	100
	rater2	7.75	19.01	73.24	6.69	21.48	71.83	0	0	100
Baseline-1	rater1	47.31	30.75	21.94	43.01	33.12	23.87	-	-	-
	rater2	47.10	30.11	22.80	34.41	41.51	24.09	-	-	-
Baseline-2	rater1	29.45	52.76	17.79	25.15	52.76	22.09	-	-	-
	rater2	33.95	46.01	20.04	27.61	48.06	24.34	-	-	-

Table 1: The evaluation results of the proposed method and two baseline methods.

kappa statistic for usability is 0.5849, which is only moderate (K: 0.41-0.6).

Table 2 shows an example of the generated paraphrases. A source sentence **s** is paraphrased in each application and we can see that: (1) for sentence compression, the paraphrase **t** is 8 bytes shorter than **s**; (2) for sentence simplification, the words *wealth* and *part* in **t** are easier than their sources *asset* and *proportion*, especially for the non-native speakers; (3) for sentence similarity computation, the reference sentence $\mathbf{s}'$ is listed below **t**, in which the words appearing in **t** but not in **s** are highlighted in blue.

5.2 Comparison with Baseline Methods

In our experiments, we implement two baseline methods for comparison:

Baseline-1: Baseline-1 follows the method proposed in (Quirk et al., 2004), which generates paraphrases using typical SMT tools. Similar to Quirk et al.'s method, we extract a paraphrase table for the SMT model from a monolingual comparable corpus (PT-2 described above). The SMT decoder used in Baseline-1 is Moses.

Baseline-2: Baseline-2 extends Baseline-1 by combining multiple resources. It exploits all PTs introduced above in the same way as our proposed method. The difference from our method is that Baseline-2 does not take different applications into consideration. Thus it contains no paraphrase planning stage or the usability sub-model.

We tune the parameters for the two baselines using the development data as described in Section 3.6 and evaluate them with the test data. Since paraphrase applications are not considered by the baselines, each baseline method outputs a single best paraphrase for each test sentence. The generation results show that 93% and 97.8% of the test sentences can be paraphrased by Baseline-1 and Baseline-2. The average number of unit replacements per sentence is 4.23 and 5.95, respectively. This result suggests that Baseline-1 is less capable than Baseline-2, which is mainly because its paraphrase resource is limited.

The generated paraphrases are also labeled by our two raters and the labeling results can be found in the lower part of Table 1. As can be seen, Baseline-1 performs poorly compared with our method and Baseline-2, as the percentage of label "1" is the highest for both adequacy and fluency. This result demonstrates that it is necessary to combine multiple paraphrase resources to improve the paraphrase generation performance.

Table 1 also shows that Baseline-2 performs comparably with our method except that it does not consider paraphrase applications. However, we are interested how many paraphrases generated by Baseline-2 can achieve the given applications by chance. After analyzing the results, we find that 24.95%, 8.79%, and 7.16% of the paraphrases achieve sentence compression, simplification, and similarity computation, respectively, which are much lower than our method.

5.3 Informal Comparison with Application Specific Methods

Previous research regarded sentence compression, simplification, and similarity computation as totally different problems and proposed distinct method for each one. Therefore, it is interesting to compare our method to the application-specific methods. However, it is really difficult for us to

Source sentence	Liu Lefei says that in the long term, in terms of asset allocation, overseas investment should occupy a certain proportion of an insurance company's overall allocation.
Sentence compression	Liu Lefei says that in [the long run]$_{phr}$, [in area of [asset allocation]$_{[NN\_1]}$]$_{pat}$, overseas investment should occupy [a [certain]$_{[JJ\_1]}$ part of [an insurance company's overall allocation]$_{[NN\_1]}$]$_{pat}$.
Sentence simplification	Liu Lefei says that in [the long run]$_{phr}$, in terms of [wealth]$_{phr}$ [distribution]$_{phr}$, overseas investment should occupy [a [certain]$_{[JJ\_1]}$ part of [an insurance company's overall allocation]$_{[NN\_1]}$]$_{pat}$.
Sentence similarity	Liu Lefei says that in [the long run]$_{phr}$, in terms [of capital]$_{phr}$ allocation, overseas investment should occupy [the [certain]$_{[JJ\_1]}$ ratio of [an insurance company's overall allocation]$_{[NN\_1]}$]$_{pat}$.
	(**reference sentence:** Liu Lefei said that in terms of capital allocation, outbound investment should make up a certain ratio of overall allocations for insurance companies in the long run .)

Table 2: The generated paraphrases of a source sentence for different applications. The target units after replacement are shown in blue and the pattern slot fillers are in cyan. $[\cdot]_{phr}$ denotes that the unit is a phrase, while $[\cdot]_{pat}$ denotes that the unit is a pattern. There is no collocation replacement in this example.

reimplement the methods purposely designed for these applications. Thus here we just conduct an informal comparison with these methods.

Sentence compression: Sentence compression is widely studied, which is mostly reviewed as a word deletion task. Different from prior research, Cohn and Lapata (2008) achieved sentence compression using a combination of several operations including word deletion, substitution, insertion, and reordering based on a statistical model, which is similar to our paraphrase generation process. Besides, they also used paraphrase patterns extracted from bilingual parallel corpora (like our PT-4) as a kind of rewriting resource. However, as most other sentence compression methods, their method allows information loss after compression, which means that the generated sentences are not necessarily paraphrases of the source sentences.

Sentence Simplification: Carroll et al. (1999) has proposed an automatic text simplification method for language-impaired readers. Their method contains two main parts, namely the lexical simplifier and syntactic simplifier. The former one focuses on replacing words with simpler synonyms, while the latter is designed to transfer complex syntactic structures into easy ones (e.g., replacing passive sentences with active forms). Our method is, to some extent, simpler than Carroll et al.'s, since our method does not contain syntactic simplification strategies. We will try to address sentence restructuring in our future work.

Sentence Similarity computation: Kauchak and Barzilay (2006) have tried paraphrasing-based sentence similarity computation. They paraphrase a sentence $\mathbf{s}$ by replacing its words with WordNet synonyms, so that $\mathbf{s}$ can be more similar in wording to another sentence $\mathbf{s}'$. A similar method has also been proposed in (Zhou et al., 2006), which uses paraphrase phrases like our PT-1 instead of WordNet synonyms. These methods can be roughly viewed as special cases of ours, which only focus on the sentence similarity computation application and only use one kind of paraphrase resource.

6 Conclusions and Future Work

This paper proposes a method for statistical paraphrase generation. The contributions are as follows. (1) It is the first statistical model specially designed for paraphrase generation, which is based on the analysis of the differences between paraphrase generation and other researches, especially machine translation. (2) It generates paraphrases for different applications with a uniform model, rather than presenting distinct methods for each application. (3) It uses multiple resources, including paraphrase phrases, patterns, and collocations, to relieve data shortage and generate more varied and interesting paraphrases.

Our future work will be carried out along two directions. First, we will improve the components of the method, especially the paraphrase planning algorithm. The algorithm currently used is simple but greedy, which may miss some useful paraphrase units. Second, we will extend the method to other applications, We hope it can serve as a universal framework for most if not all applications.

Acknowledgements

The research was supported by NSFC (60803093, 60675034) and 863 Program (2008AA01Z144). Special thanks to Wanxiang Che, Ruifang He, Yanyan Zhao, Yuhang Guo and the anonymous reviewers for insightful comments and suggestions.

References

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.

Igor A. Bolshakov and Alexander Gelbukh. 2004. Synonymous Paraphrasing Using WordNet and Internet. In *Proceedings of NLDB*, pages 312-323.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) Evaluation of Machine Translation. In *Proceedings of ACL Workshop on Statistical Machine Translation*, pages 136-158.

Jean Carletta. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. In *Computational Linguistics*, 22(2): 249-254.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, John Tait. 1999. Simplifying Text for Language-Impaired Readers. In *Proceedings of EACL*, pages 269-270.

Trevor Cohn and Mirella Lapata. 2008. Sentence Compression Beyond Word Deletion In *Proceedings of COLING*, pages 137-144.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the Question You Wish They Had Asked: The impact of paraphrasing for Question Answering. In *Proceedings of HLT-NAACL*, pages 33-36.

Jesus Gimenez and Lluis Marquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of LREC*, pages 43-46.

Hieu Hoang and Philipp Koehn. 2008. Design of the Moses Decoder for Statistical Machine Translation. In *Proceedings of ACL Workshop on Software engineering, testing, and quality assurance for NLP*, pages 58-65.

Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In Cécile L. Paris, William R. Swartout, and William C. Mann (Eds.): Natural Language Generation in Artificial Intelligence and Computational Linguistics, pages 293-312.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of HLT-NAACL*, pages 455-462.

Philipp Koehn, Franz Josef Och, Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*, pages 127-133.

Raymond Kozlowski, Kathleen F. McCoy, and K. Vijay-Shanker. 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of IWP*, pages 1-8.

J. R. Landis and G. G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. In *Biometrics* 33(1): 159-174.

De-Kang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question Answering. In *Natural Language Engineering* 7(4): 343-360.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual Dependency Parsing with a Two-Stage Discriminative Parser. In *Proceedings of CoNLL*.

Kathleen R. McKeown. 1979. Paraphrasing Using Given and New Information in a Question-Answer System. In *Proceedings of ACL*, pages 67-72.

Masaki Murata and Hitoshi Isahara. 2001. Universal Model for Paraphrasing - Using Transformation Based on a Defined Criteria. In *Proceedings of NLPRS*, pages 47-54.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*, pages 160-167.

Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, pages 311-318.

Richard Power and Donia Scott. 2005. Automatic generation of large-scale paraphrases. In *Proceedings of IWP*, pages 73-79.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*, pages 142-149.

Tetsuro Takahashi, Tomoyam Iwakura, Ryu Iida, Atsushi Fujita, Kentaro Inui. 2001. KURA: A Transfer-based Lexico-structural Paraphrasing Engine. In *Proceedings of NLPRS*, pages 37-46.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008a. Combining Multiple Resources to Improve SMT-based Paraphrasing Model. In *Proceedings of ACL-08:HLT*, pages 1021-1029.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008b. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. In *Proceedings of ACL-08:HLT*, pages 780-788.

Liang Zhou, Chin-Yew Lin, Dragos Stefan Munteanu, and Eduard Hovy. 2006. ParaEval: Using Paraphrases to Evaluate Summaries Automatically. In *Proceedings of HLT-NAACL*, pages 447-454.

Chengqing Zong, Yujie Zhang, Kazuhide Yamamoto, Masashi Sakamoto, Satoshi Shirai. 2001. Approach to Spoken Chinese Paraphrasing Based on Feature Extraction. In *Proceedings of NLPRS*, pages 551-556.