The True Score of Statistical Paraphrase Generation

Jonathan Chevelu, Ghislain Putois, Yves Lepage

The True Score of Statistical Paraphrase Generation

Jonathan Chevelu[1,2] **Ghislain Putois**[2] **Yves Lepage**[3]
(1) GREYC, université de Caen Basse-Normandie
(2) Orange Labs
(3) Waseda University
{jonathan.chevelu,ghislain.putois}@orange-ftgroup.com,
yves.lepage@aoni.waseda.jp

Abstract

This article delves into the scoring function of the statistical paraphrase generation model. It presents an algorithm for exact computation and two applicative experiments. The first experiment analyses the behaviour of a statistical paraphrase generation decoder, and raises some issues with the ordering of n-best outputs. The second experiment shows that a major boost of performance can be obtained by embedding a true score computation inside a Monte-Carlo sampling based paraphrase generator.

1 Introduction

A paraphrase generator is a program which, given a source sentence, produces a new sentence with almost the same meaning. The modification place is not imposed but the paraphrase has to differ from the original sentence.

Paraphrase generation is useful in applications where it is needed to choose between different forms to keep the most fit. For instance, automatic summary can be seen as a particular paraphrasing task (Barzilay and Lee, 2003) by selecting the shortest paraphrase. They can help human writers by proposing alternatives and having them choose the most appropriate (Max and Zock, 2008).

Paraphrases can also be used to improve natural language processing (NLP) systems. In this direction, (Callison-Burch et al., 2006) tried to improve machine translations by enlarging the coverage of patterns that can be translated. In the same way, most NLP systems like information retrieval (Sekine, 2005) or question-answering (Duclaye et al., 2003), based on pattern recognition, can be improved by a paraphrase generator.

Most of these applications need a n-best set of solutions in order to rerank them according to a task-specific criterion.

In order to produce the paraphrases, a promising approach is to see the paraphrase generation problem as a statistical translation problem. In that approach, the target language becomes the same as the source language (Quirk et al., 2004; Bannard and Callison-Burch, 2005; Max and Zock, 2008).

The first difficulty of this approach is the need of a paraphrase table. A paraphrase table is a monolingual version of a translation table in the statistical machine translation (SMT) field. In this field, the difficulty is basically overcome by using huge aligned bilingual corpora like the Europarl (Koehn, 2005) corpus. In the paraphrase generation field, one needs a huge aligned monolingual corpus to build a paraphrase table.

The low availability of such monolingual corpora nurtures researches in order to find heuristics to produce them (Barzilay and Lee, 2003; Quirk et al., 2004). On the other hand, an interesting method proposed by (Bannard and Callison-Burch, 2005) tries to make a paraphrase table using a translation table learned on bilingual corpora. The method uses a well-known heuristic (Lepage and Denoual, 2005) which says that if two sentences have the same translation, then they should be paraphrases of each others.

Another aspect, less studied, is the generation process of paraphrases, *i.e.* the decoding process in SMT. This process is subject to combinatorial

explosions. Heuristics are then frequently used to drive the exploration process in the *a priori* intractable high dimensional spaces. On the one hand, these heuristics are used to build a paraphrase step by step according to the paraphrase table. On the other hand, they try to evaluate the relevance of a step according to the global paraphrase generation model. The SMT model score is related to the path followed to generate a paraphrase. Because of the step-by-step computation, different ways can produce the same paraphrase, but with different scores. Amongst these scores, the best one is the true score of a paraphrase according to the SMT model.

Most paraphrase generators use some standard SMT decoding algorithms (Quirk et al., 2004) or some off-the-shelf decoding tools like MOSES. The goal of these decoders is to find the best path in the lattice produced by the paraphrase table. This is basically achieved by using dynamic programming – especially the *Viterbi* algorithm – and beam searching (Koehn et al., 2007). The best paraphrase proposed by these programs is known not to be the optimal paraphrase. One can even question if the score returned is the true score.

We first show in Section 2 that in the particular domain of statistical paraphrase generation, one can compute true *a posteriori* scores of generated paraphrases. We then explore some applications of the true score algorithm in the paraphrase generation field. In Section 3, we show that scores returned by SMT decoders are not always true scores and they plague the ranking of output n-best solutions. In Section 4, we show that the true score can give a major boost for holistic paraphrases generators which do not rely on decoding approaches.

2 True Score Computing

2.1 Context

The phrase based SMT model (Koehn et al., 2003) can be transposed to paraphrase generation as follows:

$$t^* = \arg\max_t P(t) \times P(s|t, B)$$

where s is the source sentence, t the target sentence *i.e.* the paraphrase, t^* the best paraphrase and B a model of the noisy channel between the source and target languages *i.e.* the paraphrase table. This can be decomposed into:

$$t^* \approx \arg\max_{t,I} P(t) \prod_{i \in I} P(s_i^I | t_i^I, B)$$

where I is a partition of the source sentence and x_i^I the i^{th} segment in the sentence x. For a given couple of s, t sentences, it exists several segmentations I with different probabilities.

This is illustrated in Example 1. Depending on the quality of the paraphrase table, one can find up to thousands of paraphrase segments for a source sentence. Note that the generated paraphrases are not always semantically or even syntactically correct, as in P2. P3 illustrates the score evaluation problem: it can be generated by applying to the source sentence the sequences of transformations $\{T1, T2\}$, $\{T1, T4, T5\}$ or even $\{T5, T1, T4\}$ ...

Example 1 Decoding

Source sentence:

The dog runs after the young cat.

Paraphrase table excerpt:

T1: P(the beast | the dog) = 0.8

T2: P(the kitten | the young cat) = 0.7

T3: P(after it | after the) = 0.4

T4: P(the | the young) = 0.05

T5: P(cat | kitten) = 0.1

Some possible generated paraphrases:

P1: the beast runs after the young cat.

P2: *the dog runs after it young cat.

P3: the beast runs after the kitten.

We define the score of a potential paraphrase t following a segmentation I as:

$$Z_t^I = P(t) \prod_{i \in I} P(s_i^I | t_i^I, B)$$

The true score of a potential paraphrase t is defined as:

$$Z_t^* = \max_I Z_t^I$$

Because of high-dimension problems, decoders apply sub-optimal algorithms to search for t^*. They produce estimated solutions over all possible paraphrases t and over all possible segmentations I. Actually, for a given paraphrase t, they consider only some Z_t^I where they should estimate Z_I^*. SMT decoders are overlooking the partitioning step in their computations.

There is no reason for the decoder solution to reach the true score. Troubles arise when one needs the scores of generated paraphrases, for instance when the system must produce an ordered n-best solution. What is the relevance of the estimated scores – and orders – with respect to the true scores – and orders – of the model? Is the true score able to help the generation process?

2.2 Algorithm

Let us first adopt the point of view proposed in (Chevelu et al., 2009). The paraphrase generation problem can be seen as an exploration problem. We seek the best paraphrase according to a scoring function in a space to search by applying successive transformations. This space is composed of states connected by actions. An action is a transformation rule with a place where it applies in the sentence. States are a sentence with a set of possible actions. Applying an action in a given state consists in transforming the sentence of the state and removing all rules that are no more applicable. In this framework, each state, except the root, can be a final state.

The SMT approach fits within this point of view. However, generation and evaluation need not to be coupled any longer. Computing the true score of a generated paraphrase is in reality a task computationally easier than generating the best paraphrases. Once the target result is fixed, the number of sequences transforming the source sentence into the target paraphrase becomes computationally tractable under a reasonable set of assumptions:

A1: the transformation rules have disjoint supports (meaning that no rule in the sequence should transform a segment of the sentence already transformed by one of of the previous applied rules) ;

A2: no reordering model is applied during the paraphrasing transformation.

Under this set of assumptions, the sequence (ordered) of transformation rules becomes a set (unordered) of transformation rules. One can therefore easily determine all the sets of transformation rules from the source sentence to the target paraphrase: they are a subset of the cross-product set of every transformation rule with a source included in the source sentence and with a result included in the target paraphrase. And this cross-product set remains computationally tractable. Note that to guarantee a solution, the corpus of all rules should be augmented with an identity rule for each word of the source sentence (with an associated probability of applicability set to 1) missing in the paraphrase table.

The algorithm for computing *ex post* the true score is given on algorithm 1.

Algorithm 1 Algorithm for true score

Let S be the source sentence.
Let T be the target sentence.
Let $R : s_R \rightarrow t_R$ be a transformation rule

Let $map : (S, T) \rightarrow C$ be a function

Let $C = \{\emptyset\}$

$\forall s_{head} | S = s_{head}.s_{tail},$

$\forall R \in \{\Omega | s_R = s_{head}, T = t_R.t_{tail}\}$

$C = C \cup (\{R\} \otimes map(S_{tail}, T_{tail}))$

return C

Let $score$ be the scoring function for a transformation rule set

$$truescore_{S,\Omega}(T) = \arg \max_{c \in map(S,T)} (score(c))$$

For our toy example, we would get the steps shown in Example 2.

3 True Score of SMT Decoders

We have shown that it is possible to compute the true score according to the paraphrase model. We now evaluate scores from a state-of-the-art

Example 2 True Score Computation

Generated sets:

$\{R1\}, \{R1, R3\}, \{R1, R2\},$

$\{R1, R4\}, \{R1, R4, R5\},$

$\{R3\},$

$\{R2\},$

$\{R4\}, \{R4, R5\},$

$\{R5\}$

For a better readability, all identity rules are omitted.

The true scores are computed as in the following examples:

$score($ "the dog runs after the small cat." $\rightarrow$
"the beast runs after it small cat")
$= score(\{R1\})$

$score($ "the dog runs after the small cat." $\rightarrow$
"the beast runs after the kitten")
$= \max(score(\{R1, R2\}), score(\{R1, R4, R5\}))$

decoder against this baseline. In particular, we are interested in the order of n-best outputs. We use the MOSES decoder (Koehn et al., 2007) as a representative SMT decoder inside the system described below.

3.1 System description

Paraphrase generation tools based on SMT methods need a language model and a paraphrase table. Both are computed on a training corpus.

The language models we use are n-gram language models with back-off. We use SRILM (Stolcke, 2002) with its default parameters for this purpose. The length of the n-grams is five.

To build a paraphrase table, we use a variant of the construction method via a pivot language proposed in (Bannard and Callison-Burch, 2005). The first step consists in building a bilingual translation table from the aligned corpus. Given a source phrase s^i and another phrase t^i in a different language, a bilingual translation table provides the two probabilities $p(s^i|t^i)$ and $p(t^i|s^i)$. We use GIZA++ (Och and Ney, 2003) with its default parameters to produce phrase alignments. The paraphrase table is then built from the phrase translation table. The probability for a phrase s^i to be paraphrased by a phrase s'^i in the same language is estimated by the sum of each round-trip from s^i to s'^i through any phrase t^i of a pivot language.

The construction of this table is very simple. Given a bilingual translation table sorted by pivot phrases, the algorithm retrieves all the phrases linked with the same pivot (named a *pivot cluster*). For each ordered pair of phrases, the program assigns a probability that is the product of there probabilities. This process realizes a self-join of the bilingual translation table. It produces a paraphrase table composed of tokens, instead of items. The program just needs to sum up all probabilities for all entries with identical paraphrase tokens to produce the final paraphrase table.

Three heuristics are used to prune the paraphrase table. The first heuristic prunes any entry in the paraphrase table composed of tokens with a probability lower than a threshold ϵ. The second, called *pruning pivot heuristic*, consists in deleting all pivot clusters larger than a threshold τ. The last heuristic keeps only the κ most probable paraphrases for each source phrase in the final paraphrase table. For this study, we empirically fix $\epsilon = 10^{-5}$, $\tau = 200$ and $\kappa = 20$.

The MOSES scoring function is set by four weighting factors $\alpha_\Phi, \alpha_{LM}, \alpha_D, \alpha_W$. Conventionally, these four weights are adjusted during a tuning step on a training corpus. The tuning step is inappropriate for paraphrasing because there is no such tuning corpus available. We empirically set $\alpha_\Phi = 1$, $\alpha_{LM} = 1$, $\alpha_D = 10$ and $\alpha_W = 0$. This means that the paraphrase table and the language model are given the same weight, no reordering is allowed and no specific sentence length is favored.

3.2 Experimental Protocol

For experiments reported in this paper, we use one of the largest, multi-lingual, freely available aligned corpus, Europarl (Koehn, 2005). It consists of European parliament debates. We choose French as the language for paraphrases and English as the pivot language. For this pair of languages, the corpus consists of 1,723,705 sentences. Note that the sentences in this corpus are long, with an average length of 30 words per French sentence and 27.8 for English. We randomly extract 100 French sentences as a test cor-

pus.

For each source sentence from the test corpus, the SMT decoder tries to produce a 100-best distinct paraphrase sequence. Using the algorithm 1, we compute the true score of each paraphrase and rerank them. We then compare orders output by the decoder with the true score order by using the Kendall rank correlation coefficient (τ_A) (Kendall, 1938). In this context, the Kendall rank correlation coefficient considers each couple of paraphrases and checks if their relative order is preserved by the reranking. The τ_A formula is:

$$\tau_A = \frac{n_p - n_i}{\frac{1}{2}n(n-1)}$$

where n_p the number of preserved orders, n_d the number of inverted orders and n the number of elements in the sequence. The coefficient provides a score – between -1 and 1 – that can be interpreted as a correlation coefficient between the two orders. In order to compare same length sequences, we filter out source sentences when MOSES can not produce enough distinct paraphrases. The test corpus is therefore reduced to 94 sentences.

3.3 Results

The evolution of τ_A means relative to the length of the n-best sequence is given Figure 1. The τ_A means drops to 0.73 with a standard deviation of 0.41 for a 5-best sequence which means that the orders are clearly different but not decorrelated.

A finer study of the results reveals that amongst the generated paraphrases, 32% have seen their score modified. 18% of the MOSES 1-best paraphrases were not optimal anymore after the true score reranking. After reranking, the old top best solutions have dropped to a mean rank of 2.0 ± 17.7 (40th rank at worse). When considering only the paraphrases no longer optimal, they have dropped to a mean rank of 6.8 ± 12.9.

From the opposite point of view, new top paraphrases after reranking have come from a mean rank of 4.4 ± 12.1. When considering only the paraphrases that were not optimal, they have come from a mean rank of 21.2 ± 23.5. Some have come from the 67th rank. Even an *a posteriori* reranking would not have retrieved this top solution if the size of MOSES n-best list were too short. This

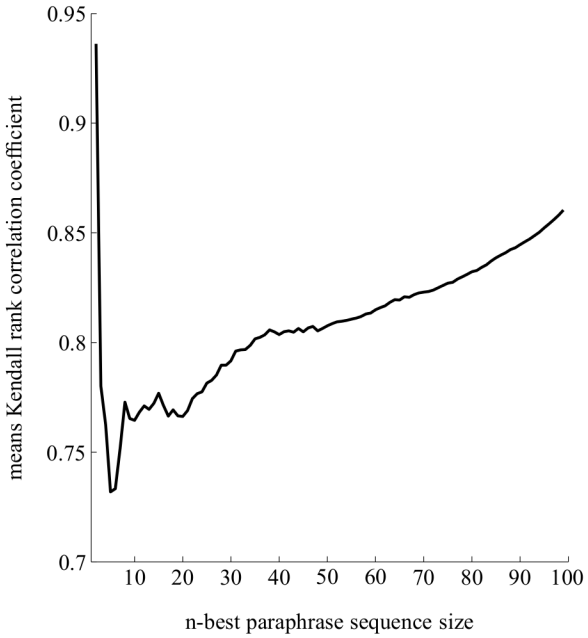

Figure 1: Evolution of τ_A means relative to the length of the n-best sequence

advocates for a direct embedding of the true score function inside the generation process.

In this section we have shown that MOSES scores are not consistent with the true score as expected from the paraphrase model. In particular, the n-best paraphrase sequence computed by MOSES is not trustworthy while it is an input for the task system.

4 True Score to boost Monte-Carlo based Paraphrase Generation

There exist other less common approaches more lenient than the *Viterbi* algorithm, which are holistic, *i.e.* they work on the whole sentence rather than step-by-step. The *Monte-Carlo based Paraphrase Generation* algorithm (MCPG) proposed in (Chevelu et al., 2009) turns out to be an interesting algorithm for the study of paraphrase generation. It does not constraint the scoring function to be incremental. In this section, we embed the non incremental true score function in MCPG to drive the generation step and produce n-best orders compliant with the paraphrase model, and show that the true score function can be used to provide a major boost to the performance of such

an algorithm.

4.1 Description

The MCPG algorithm is a derivative of the *Upper Confidence bound applied to Tree* algorithm (UCT). UCT (Kocsis and Szepesvári, 2006), a *Monte-Carlo* planning algorithm, has recently become popular in two-player game problems.

UCT has some interesting properties:

- it expands the search tree non-uniformly and favours the most promising sequences, without pruning branch;
- it can deal with high branching factors;
- it is an any-time algorithm and returns best solutions found so far when interrupted;
- it does not require expert domain knowledge to evaluate states.

These properties make it ideally suited for problems with high branching factors and for which there is no strong evaluation function.

For the same reasons, this algorithm is interesting for paraphrase generation. In particular, it does not put constraint on the scoring function. A diagram of the MCPG algorithm is presented Figure 2.

The main part of the algorithm is the sampling step. An episode of this step is a sequence of states and actions, $s_1, a_1, s_2, a_2, \ldots, s_T$, from the root state to a final state. Basically, a state is a partially generated paraphrase associated with a set of available actions. A final state is a potential paraphrase. An action is a transformation rule from the paraphrase table. During an episode construction, there are two ways to select the action a_i to perform from a state s_i.

If the current state was already explored in a previous episode, the action is selected according to a compromise between exploration and exploitation. This compromise is computed using the UCB-Tunned formula (Auer et al., 2001) associated with the RAVE heuristic (Gelly and Silver, 2007). If the current state is explored for the first time, its score is estimated using *Monte-Carlo* sampling. In other words, to complete the

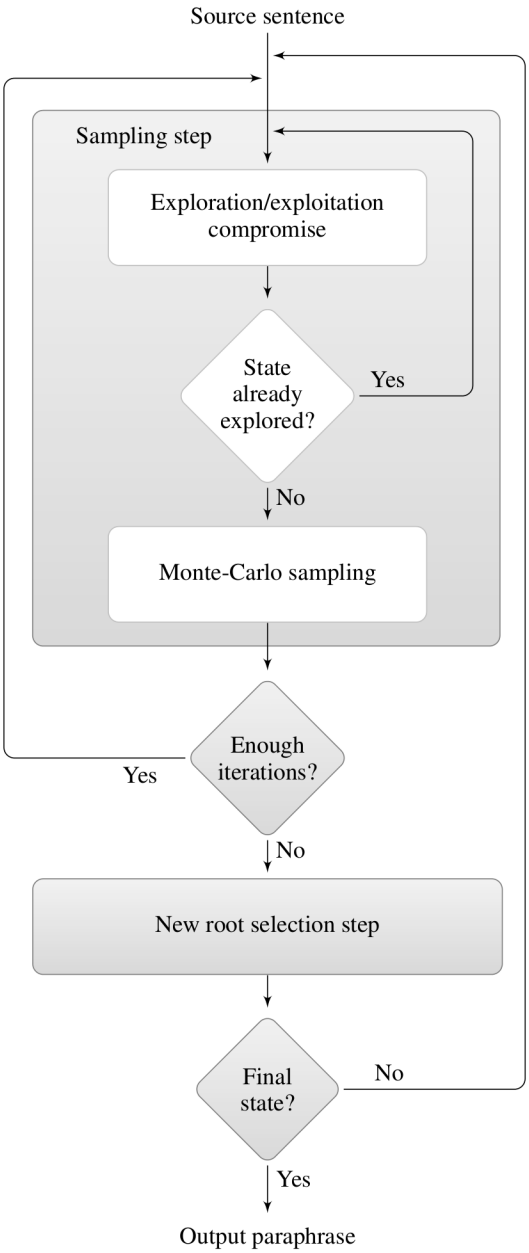

Figure 2: The MCPG algorithm.

episode, the actions $a_i, a_{i+1}, \ldots, a_{T-1}, a_T$ are selected randomly until reaching a final state.

At the end of each episode, a reward is computed for the final state s_T using a scoring function, and the value of each (state, action) pair of the episode is updated. Then, the algorithm computes another episode with the new values.

Periodically, the sampling step is stopped and the best action at the root state is selected. This action is then definitively applied and a sampling is restarted from the new root state. The action sequence is incrementally built and selected after being sufficiently sampled. For our experiment, we have chosen to stop sampling regularly after a fixed amount η of episodes.

The adaptation of the original algorithm takes place in the (state, action) value updating procedure. Since the goal of the algorithm is to maximise a scoring function, it uses the maximum reachable score from a state as value instead of the score expectation. This algorithm suits the paradigm recalled in Section 2 for paraphrase generation.

To provide scores comparable with the paraphrase model scores, the standard version of MCPG has to apply rules until the whole source sentence is covered. With this behaviour, MCPG acts in a monolingual "translator" mode.

The embedding of the true score algorithm in MCPG has given meaningful scores to all states. The algorithm needs not to "translate" the whole sentence to get a potential paraphrase and its score. This MCPG algorithm in "true-score" mode can choose to stop its processing with segments still unchanged, which solves, amongst others, out-of-vocabulary questions found in decoder-based approaches.

4.2 Experimental Protocol

For this experiment, we reuse the paraphrase table and the corpora generated for the experiment presented in Section 3.2;

We compare the 1-best outputs from MOSES reranked by the true score function and from MCPG in both "translator" and "true-score" modes. For MCPG systems, we set the following parameters: $\eta = 100{,}000$ iterations.

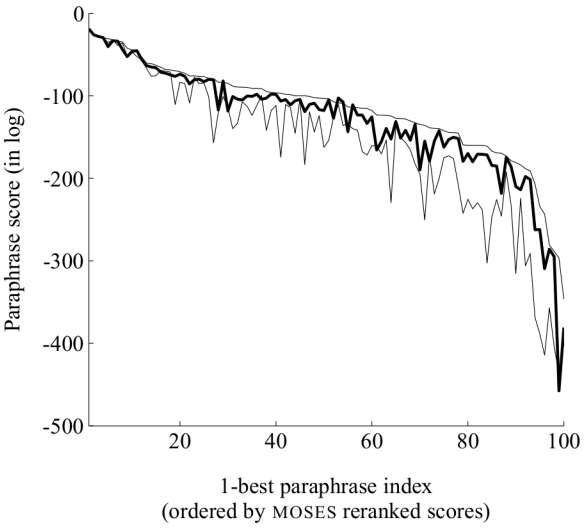

Figure 3: Comparison of paraphrase generators. Top: the MOSES baseline; middle and bold: the "true-score" MCPG; down: the "translator" MCPG. The use of "true-score" improves the MCPG performances. MCPG reaches MOSES performance level.

4.3 Results

Figure 3 presents a comparison between the scores from each systems, ordered by MOSES reranked scores.

The boost of performance gained by using true scores inside the MCPG algorithm reaches a means of 28.79 with a standard deviation of 34.19. The mean difference between "true-score" MCPG and MOSES is -14.13 (standard deviation 19.99). Although the performance remains inferior to the MOSES true score baseline, it still leads to an improvement over the "translator" MCPG system. The later system has a mean difference of performance with MOSES of -42.92 (standard deviation of 40.14).

The true score reduces the number of transformations needed to generate a paraphrase, which simplifies the exploration task. Moreover, it reduces the number of states in the exploration space: two sets of transformations producing the same paraphrase now leads to the same state. These points explain why MCPG has become more efficient.

Although MCPG is improved by embedding the

true score algorithm, there is still room for improvement. In its current version, MCPG does not adapt the number of exploration episodes to the input sentence.

5 Conclusion and perspectives

In this paper, we have developed a true scoring algorithm adapted to the statistical paraphrase generation model. We have studied its impacts on a common SMT decoder and a Monte-Carlo sampling based paraphrase generator. It has revealed that the n-best outputs by SMT decoders were not viable. It has also proved useful in simplifying the exploration task and in improving holistic paraphrase generators.

Thanks to the boost introduced by the true score algorithm in holistic paraphrase generators, their performances are now on a par with scores produced by statistical translation decoders. Moreover, they produce guaranteed ordering, and enable the integration of a global task scoring function, which seems still out of reach for decoder-based systems.

A more general problem remains open: what do the scores and the orders output by the model mean when compared to a human subjective evaluation?

In preliminary results on our test corpus, less than 37% of the MOSES generated paraphrases can be considered both syntactically correct and semantically a paraphrase of their original sentence. One could study the relations between scores from the model and subjective evaluations to create predictive regression models. The true score algorithm can autonomously score existing paraphrase corpora which could be used to adapt the SMT tuning step for paraphrase generation.

We note that the hundredth best paraphrases from MOSES have a score close to the best paraphrase: the mean difference is 5.9 (standard deviation 4.5) on our test corpus. This is smaller than the mean difference score between MOSES and MCPG. In (Chevelu et al., 2009), both systems were rated similar by a subjective evaluation. One could question the relevance of small score differences and why the best paraphrase should be selected instead of the hundred next ones. Given the current state of the art, the next step to improve paraphrase generation does not lie in score optimisation but in refining the model and its components: the language model and the paraphrase table.

Human based evaluations reveal that the current most important issue of paraphrase generation lies in the syntax (Chevelu et al., 2009). It seems difficult to assess the syntax of a potential paraphrase while not considering it as a whole, which is impossible with a local scoring function inherent to the SMT decoding paradigm. Holistic paraphrase generators have now reached a level of performance comparable to SMT decoders, without suffering from their limitations. They are paving the way for experiments with more complex semantic and linguistic models to improve paraphrase generation.

References

Auer, P., N. Cesa-Bianchi, and C. Gentile. 2001. Adaptive and self-confident on-line learning algorithms. *Machine Learning*.

Bannard, Colin and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604, Morristown, NJ, USA. Association for Computational Linguistics.

Barzilay, Regina and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL 2003: Main Proceedings*, pages 16–23.

Callison-Burch, Chris, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 17–24, Morristown, NJ, USA. Association for Computational Linguistics.

Chevelu, Jonathan, Thomas Lavergne, Yves Lepage, and Thierry Moudenc. 2009. Introduction of a new paraphrase generation tool based on Monte-Carlo sampling. In Su, Keh-Yih, Jian Su, Janyce Wiebe, and Haizhou Li, editors, *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 249–252, Singapoure, August. Association for Computational Linguistics.

Duclaye, Florence, François Yvon, and Olivier Collin. 2003. Learning paraphrases to improve a question-answering system. In *In Proceedings of the 10th Conference of EACL Workshop Natural Language Processing for Question-Answering*, page 3541.

Gelly, Sylvain and David Silver. 2007. Combining online and offline knowledge in UCT. In *24th International Conference on Machine Learning (*ICML*'07)*, pages 273–280, June.

Kendall, Maurice G. 1938. A New Measure of Rank Correlation. *Biometrika*, 1–2(30):81–89, June.

Kocsis, Levente and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *17th European Conference on Machine Learning, (*ECML*'06)*, pages 282–293, September.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 48–54, Edmonton, May. Association for Computational Linguistics.

Koehn, Philipp, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session*, pages 177–180, June.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

Lepage, Yves and Etienne Denoual. 2005. Automatic generation of paraphrases to be used as translation references in objective evaluation measures of machine translation. In *IWP2005*.

Max, Aurélien and Michael Zock. 2008. Looking up phrase rephrasings via a pivot language. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 97–104, Manchester, UK, August. Coling 2008 Organizing Committee.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Quirk, Chris, Chris Brockett, and Bill Dolan. 2004. Monolingual machine translation for paraphrase generation. In Lin, Dekang and Dekai Wu, editors, *the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149., Barcelona, Spain, 25-26 July. Association for Computational Linguistics.

Sekine, Satoshi. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of International Workshop on Paraphrase (IWP2005)*.

Stolcke, Andreas. 2002. Srilm – an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*.