Neural-Driven Multi-criteria Tree Search for Paraphrase Generation
Betty Fabre, Tanguy Urvoy, Jonathan Chevelu, Damien Lolive

To cite this version:

Betty Fabre, Tanguy Urvoy, Jonathan Chevelu, Damien Lolive. Neural-Driven Multi-criteria Tree Search for Paraphrase Generation. Learning Meets Combinatorial Algorithms (LMCA) Workshop at NeurIPS 2020, Dec 2020, online, France. hal-03127865

HAL Id: hal-03127865
https://inria.hal.science/hal-03127865
Submitted on 1 Feb 2021
Neural-Driven Multi-criteria Tree Search for Paraphrase Generation

Betty Fabre
Orange Labs, Lannion
Univ Rennes, IRISA
betty.fabre@orange.com

Tanguy Urvoy
Orange Labs Lannion
tanguy.urvoy@orange.com

Jonathan Chevelu
Univ Rennes, IRISA Lannion
jonathan.chevelu@irisa.fr

Damien Lolive
Univ Rennes, IRISA Lannion
damien.lolive@irisa.fr

Abstract

A good paraphrase is semantically similar to the original sentence but it must be also well formed, and syntactically different to ensure diversity. To deal with this trade-off, we propose to cast the paraphrase generation task as a multi-objectives search problem on the lattice of text transformations. We use BERT and GPT2 to measure respectively the semantic distance and the correctness of the candidates. We study two search algorithms: Monte-Carlo Tree Search For Paraphrase Generation (MCPG) and Pareto Tree Search (PTS) that we use to explore the huge sets of candidates generated by applying the PPDB-2.0 edition rules. We evaluate this approach on 5 datasets and show that it performs reasonably well and that it outperforms a state-of-the-art edition-based text generation method.

1 Introduction

The generation of paraphrase, i.e. the transformation of a sentence into a well-formed but lexically different one while preserving its original meaning, is a fundamental task of NLP. Its ability to provide diversity and coverage finds applications in several domains like question answering [5], machine-translation [4], dialog systems [15], privacy [22] or more recently adversarial learning [19].

The formal definition of paraphrase may vary according to the targeted application and the tolerance we set along several axes including:

1. the semantic distance from the source that we want to minimize;
2. the quality of the syntax that we want to keep high;
3. the lexical distance from the source that we want to maximize to ensure diversity.

Automatically generating paraphrases that are semantically accurate, diverse, and well-formed remains a challenging problem, and with the lack of large and high-quality aligned paraphrase corpora, it is difficult to train generic supervised models on this task.

On the other hand, assessing the quality of a given sentence as a paraphrase has become a much more tractable task. Indeed, the availability of contextual embeddings networks trained on huge corpora have led to the development of high-quality semantic distances [18][17][25][26]. Regarding the quality of the syntax, the perplexity of a large-scale language model like GPT2 [24] is an excellent ranking criterion as well. For the lexical distance we use a simple Levenshtein edition distance. Leveraging on these three metrics, we cast the paraphrase generation task as a multi-criteria search problem. In this experiment we use PPDB 2.0 [13], a large-scale database of rewriting rules derived from bilingual
corpora, to potentially generate billions of ’naive’ candidate paraphrases by edition from the source sentence.

To sort efficiently the good candidates from the others, we experiment two algorithms. The first one, called MCPG for Monte-Carlo Paraphrase Generation, is a variant of the Monte-Carlo Tree Search algorithm (MCTS) [6,7,9]. This algorithm is famous for its successes on mastering the – highly combinatorial – game of Go [7,14]. The second one is a novel search algorithm that we call Pareto Tree Search (PTS). In contrast to MCTS which is a single-criterion search algorithm, we designed PTS to retrieve an approximation of the whole Pareto optimal set. This allows for more flexibility on paraphrase generation where the balance between semantic distance, lexical distance, and syntax quality is hard to tune a priori.

2 Paraphrase Generation Lattice

We model the paraphrase generation as a sequence of editions and transformations from a source sentence into its paraphrase. In this work, we only consider local transformations, i.e. replacement of certain words or phrases by others, but the method we propose should work with more sophisticated transformations schemes as well.

The Paraphrase Database (PPDB) [12,13] is a large collection of paraphrase edition rules that was automatically constructed from various bilingual corpora using a pivoting alignment method [8]. The PPDB database is divided into increasingly large and decreasingly accurate subsets [1]. We used the XL subset, and we removed the rules labeled as ”Independent”. This left us with a set of 5.5 million rewriting rules. We give some examples of these rules on Table 1. By iteratively applying the rules from a source sentence like the one on Table 1, we obtain a vast lattice of candidate paraphrases. Some of these candidates like ”he’s talking on june 14” are well formed, but many like ”he is speak now on june 14” are syntactically broken. The number of rules that apply, depends on the size of the source sentence and the words it contains. For instance on the MSRPARAPHRASE corpora, to potentially generate billions of paraphrase candidates by edition from the source sentence.

<table>
<thead>
<tr>
<th>Source sentence</th>
<th>Edited sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>he is speaking on june 14</td>
<td>he is found speaking on june 14</td>
</tr>
<tr>
<td>is → is found</td>
<td>he is found speaking on june 14</td>
</tr>
<tr>
<td>is speaking → ’s talking</td>
<td>he ’s talking on june 14</td>
</tr>
<tr>
<td>speaking → speak now</td>
<td>he is speak now on june 14</td>
</tr>
<tr>
<td>14 → 14th</td>
<td>he is speaking on june 14</td>
</tr>
</tbody>
</table>

Table 1: PPDB rules samples applied to a source sentence

3 Searching Algorithms

Searching for good paraphrases in the large lattice of candidates generated by PPDB is a costly task. The two algorithms that we propose are described as pseudo-code on Tables 2 and 3. They share a similar structure: the outer loop explores the lattice at different depths, while the inner loop explores the candidates at each depth. Both algorithms are anytime: they return the best solutions found so far when the time or space budget is depleted.

MCPG: Monte-Carlo Tree Search for Paraphrase Generation. Following the idea of [2], we used Monte-Carlo Tree Search (MCTS) to explore the PPDB lattice. The three key ingredients of MCTS are: the use of a bandit policy at each node of a search-tree to select the most promising paths, the use of randomized roll-outs to estimate the quality of these paths, and the back-propagation of rewards along the paths to update the bandit. We opted here for the EXP3 [2] randomized bandit policy which proved to be robust than UCB [1].

function MCPG(input_sentence)
    candidates ← REWRITE(input_sentence)
    depth ← 1
    while time/space < budget do
        while time/space < layer-budget do
            nodes ← MAB.SELECTS_FROM(candidates)
            paths ← ROLLOUT_FROM(nodes)
            scored ← scored ∪ NN.SCORERLEAVES(paths.leaves)
            Backward node rewards along paths
            MAB.UPDATE_POLICY(node rewards)
        end while
        layer_best ← ARGMAX(scored)
        candidates ← candidates ∪ REWRITE(layer_best)
        depth ← depth + 1
    end while
    return ARGMAX(all scored nodes)
end function

Table 2: Monte-Carlo Tree Search (MCPG) algorithm
The MCTS algorithm being not designed for multi-objective problems, we had to combine semantic similarity BERT_S, syntax correctness GPT2 and surface diversity LevS into a single criterion. The simplest option was a linear scalarization but after some quantitative analysis of the scores distributions like in Figure 1, we realized that it was easy for the search algorithm to maximize the score by just applying a lot of editions to the source sentence. To overcome this, we replaced the LevS term by LevS \cdot BERT_S which led us to the polynomial form \( \alpha \cdot BERT_S + \beta \cdot LevS \cdot BERT_S - \gamma \cdot GPT2 \). After a few experiments on train sets, we tuned empirically the weights to \( \alpha = 3 \), \( \beta = 0.5 \) and \( \gamma = 0.025 \) in order to obtain a balanced policy.

**PTS: Pareto Tree Search in the paraphrase lattice**  
The MCTS algorithm is powerful but we observed two drawbacks for paraphrase generation. First, it was designed for combinatorial problems like Go where the evaluation is only possible on the leaves of the search tree. This is not the case for paraphrase generation where our neural models can cheaply evaluate any rewriting step and where rewriting from good candidates is more likely to provide good paraphrases than rewriting from bad ones. Secondly, because it was designed for single criterion search it requires to set the balance between semantic similarity, lexical distance and syntax quality definitively before any paraphrase search begins. This is not very flexible and it becomes painful when we want to generate sets of candidates.

By plotting the distributions of the scores as on Figure 1, we noticed that most of the candidates were dominated in the Pareto sense: it was possible to eliminate most of the bad candidates without any hyper-parameter tuning. We hence adapted MCPG in order to explore the paraphrase lattice and recover an approximation of the Pareto front, postponing the balance between the criteria as a quick post-optimization stage. This flexibility is also useful for specific tasks like adversarial learning [19] where a valid paraphrase is selected in post-optimization according to its ability to deceive a given text classification model.

### 4 Experiments

**Datasets and metrics**  
To evaluate the paraphrase generation models, we rely on machine translation metrics that compare the generated paraphrase to one or several references [11].

---

**Table 3: Pareto Tree Search (PTS)**

<table>
<thead>
<tr>
<th>Function</th>
<th>PTS(input_sentence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>candidates ← REWRITE(input_sentence)</td>
<td></td>
</tr>
<tr>
<td>depth ← 1</td>
<td></td>
</tr>
<tr>
<td>while time/space &lt; budget do</td>
<td></td>
</tr>
<tr>
<td>while time/space &lt; layer-budget do</td>
<td></td>
</tr>
<tr>
<td>batch ← SAMPLE(candidates)</td>
<td></td>
</tr>
<tr>
<td>scored ← scored ∪ NN.SCORES(batch)</td>
<td></td>
</tr>
<tr>
<td>end while</td>
<td></td>
</tr>
<tr>
<td>layer_front_set ← PARETO-FRONT(scored)</td>
<td></td>
</tr>
<tr>
<td>candidates ← REWRITE(layer_front_set)</td>
<td></td>
</tr>
<tr>
<td>depth ← depth + 1</td>
<td></td>
</tr>
<tr>
<td>end while</td>
<td></td>
</tr>
<tr>
<td>return PARETO-FRONT(all scored nodes)</td>
<td></td>
</tr>
<tr>
<td>end function</td>
<td></td>
</tr>
</tbody>
</table>

---
We report the surface metric \textbf{BLEU} \cite{papineni2002bleu} and the semantic metric \textbf{BERT} \cite{devlin2018bert}: the average BERT score of the generated sentence with respect to the \textbf{ground truth} sentences\footnote{This usage of BERT score against \textbf{reference/ground-truth} must not be confused with the BERT score against source sentence ($BERT_S$) that we use in the MCPG score.}

On Table \ref{tab:datasets} we give a summary of the datasets we used. On one hand we have large datasets like MSCOCO or OPUSPARCUS (OPUSPARA.) that are noisy and very specific. On the other hand we have MSRPARAPHRASE (MSRPARA.), a high-quality but small human-labeled dataset.

### Baseline

Constrained Sentence Generation by Metropolis-Hastings Sampling (CGMH) \cite{balahur2013constraint} is an approach that uses Metropolis-Hastings sampling \cite{roberts1996general} for constrained sentence generation.

Starting from the source sentence, the CGMH model samples a sequence of sentences by using local editions: word replacement, deletion, and insertion. For paraphrase generation, CGMH constraints the sentence generation using a matching function that combines a measure of semantic similarity and a measure of English fluency. This model is therefore directly comparable with the MCPG and PTS approaches.

### Results

On Table \ref{tab:experiments} we report the evaluation of the CGMH, MCPG and PTS models on the five datasets of aligned paraphrases pairs. MCPG and PTS models outperform the CGMH baseline model on all corpora except on the MSCOCO dataset where the results are mixed. The low PTS upper-bound on MSCOCO and OPUSPARCUS indicates that PPDB-XL vocabulary does not match well with these datasets.

## 5 Discussion

We experimented two search-based approaches for paraphrase generation. These approaches are pragmatic and flexible. When compared against CGMH, another search-based and weakly-supervised method, our algorithms proved to be faster and more efficient. The use of local transformations however lead our models to produce paraphrases that are a bit too conservative, especially when starting from short sentences. For future work, we plan to refine the scoring with deep reinforcement learning techniques and enrich the rules with more sophisticated patterns like phrases permutations. We also plan to hybridize the search method with supervised seq2seq models \cite{vaswani2017attention} by data-augmentation, or through a planning-then-realization scheme like in \cite{grasmug2016combining} and \cite{zhou2016neural}.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|l|}
\hline
Corpus & Size & Len & PPDB@1 & PPDB@3 \\
\hline
MSCOCO & $2.4 \times 10^6$ & 11 & 119 & $7.4 \times 10^4$ \\
MSRPARA. & $7.8 \times 10^3$ & 23 & 454 & $7.0 \times 10^6$ \\
OPUSPARA. & $4.2 \times 10^7$ & 6 & 113 & $1.5 \times 10^4$ \\
PAWS & $3.6 \times 10^5$ & 20 & 289 & $1.3 \times 10^6$ \\
QQP & $1.5 \times 10^5$ & 10 & 141 & $1.3 \times 10^5$ \\
\hline
\end{tabular}
\caption{Datasets statistics. 'Size' is the number of instances. 'Len' is the median number of words per sentence. The 'PPDB@x' columns give the median number of candidates that PPDB-XL generates from one sentence respectively at one and three rewriting steps.}
\label{tab:datasets}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|l|}
\hline
Corpus & Model & \textit{BLEU} & \textit{BERT} \\
\hline
MSCOCO & CGMH & 17.3 & 0.7 \\
 & MCPG & 16.5 & \textbf{0.71} \\
 & PTS & 17.0 & 0.64 \\
 & \textit{PTS upper-b.} & 17.6 & 0.64 \\
 & \textit{PTS random} & 6.4 & 0.56 \\
MSRPARA. & CGMH & 3.7 & 0.48 \\
 & MCPG & 39.3 & \textbf{0.81} \\
 & PTS & \textbf{40.3} & 0.80 \\
 & \textit{PTS upper-b.} & 40.9 & 0.82 \\
 & \textit{PTS random} & 28.3 & 0.74 \\
OPUSPARA. & CGMH & 7.0 & 0.58 \\
 & MCPG & 9.6 & 0.67 \\
 & PTS & 9.1 & \textbf{0.68} \\
 & \textit{PTS upper-b.} & 14.5 & 0.68 \\
 & \textit{PTS random} & 4.3 & 0.57 \\
PAWS & CGMH & 15.4 & 0.61 \\
 & MCPG & 55.5 & \textbf{0.93} \\
 & PTS & \textbf{57.9} & 0.92 \\
 & \textit{PTS upper-b.} & 65.3 & 0.93 \\
 & \textit{PTS random} & 36.5 & 0.82 \\
QQP & CGMH & 22.5 & 0.72 \\
 & CGMH \cite{balahur2013constraint} & 18.8 & 0.72 \\
 & MCPG & 24.1 & \textbf{0.78} \\
 & PTS & \textbf{25.6} & 0.78 \\
 & \textit{PTS upper-b.} & 32.7 & 0.79 \\
 & \textit{PTS random} & 9.8 & 0.66 \\
\hline
\end{tabular}
\caption{Experiments summary. Higher values are better, best in bold. The PTS policy maximizes the MCPG criterion among the Pareto candidates. The PTS \textit{random} policy picks a random candidate in the Pareto set. The \textit{PTS upper-bound (upper-b.)} policy is "cheating" by taking a candidate that maximizes individual \textit{BLEU} against the target. These two former policies are given to assess the "quality" of the generated set of candidates.}
\label{tab:experiments}
\end{table}
References


