Neural-Driven Multi-criteria Tree Search for Paraphrase Generation

Anonymous Author(s)
Affiliation
Address
email

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 [14], privacy [20] or more recently adversarial learning [17].

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* [16, 15, 23, 24]. Regarding the *quality of the syntax*, the perplexity of a large-scale language model like GPT2 [22] 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 [12], 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*

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

algorithm (MCTS) [6, 7, 9]. This algorithm is famous for its successes on mastering the – highly combinatorial – game of Go [7, 13].

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) [11, 12] 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].

Source sentence :	he is speaking on june 14 .
PPDB rule	**Edited sentence**
is → is found	he is found speaking on june 14 .
is speaking → 's talking	he 's talking on june 14 .
speaking → speak now	he is speak now on june 14 .
14 → 14th	he is speaking on june 14th .

Table 1: PPDB rules samples applied to a source sentence

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 dataset (see section 4), the sentences are quite long and the median number of PPDB-XL rules that apply is around 450. After two steps of rewriting, the median number of candidates is around 10^5 and by iterative rewriting, we quickly reach a number of paraphrase candidates that is greater than 10^8.

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.

```
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.SCORE(paths.leaves)
            Backward node rewards along paths
            MAB.UPDATES_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

MCPG: Monte-Carlo Tree Search for Paraphrase Generation Following the idea of [9], 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 a randomized bandit policy called EXP3 [2] which proved to be robuster than UCB [1]. The MCTS algorithm being not designed for multi-objective

[1] PPDB is available on `http://paraphrase.org`, and a python interface is available here `https://github.com/erickrf/ppdb`

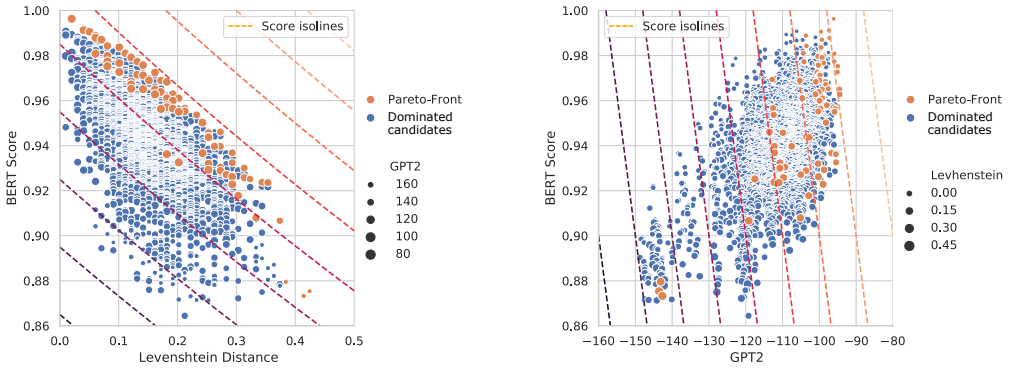

Figure 1: A cloud of candidates generated from a sample of the MSRPARAPHRASE. The optima of any positive combination of BERT score, normalized Levenshtein distance, and GPT2 perplexity belong to the Pareto front (orange dots). We plotted the projections of MCPG combined score with dashed isolines. The BERT score and Levenshtein distance being clearly anti-correlated, the balance between these two criteria is difficult to tune.

problems, we had to combine semantic similarity BERT_S, syntax correctness GPT2 and surface diversity Lev_S into a single criterion.

The simplest option was a linear scalarization but after some quantitative analysis of the scores distributions like in 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 Lev_S term by $\text{Lev}_\text{S} \cdot \text{BERT}_\text{S}$ which led us to the polynomial form $\alpha \cdot \text{BERT}_\text{S} + \beta \cdot \text{Lev}_\text{S} \cdot \text{BERT}_\text{S} - \gamma \cdot \text{GPT2}$. 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.

```
function PTS(input_sentence)
    candidates ← REWRITE(input_sentence)
    depth ← 1
    while time/space < budget do
        while time/space < layer-budget do
            batch ← SAMPLE(candidates)
            scored ← scored ∪ NN.SCORES(batch)
        end while
        layer_front_set ← PARETO-FRONT(scored)
        candidates ← REWRITE(layer_front_set)
        depth ← depth + 1
    end while
    return PARETO-FRONT(all scored nodes)
end function
```

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

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 [17] 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 standard machine translation metrics that compare the generated paraphrase to one or several references [10].

We report the surface metric BLEU and the semantic metric BERT[2]: the average BERT score of the generated sentence with respect to the **ground truth** sentences[3].

Corpus	Size	Len	PPDB@1	PPDB@3
MSCOCO	$2.4 \cdot 10^6$	11	119	$7.4 \cdot 10^4$
MSRPARA.	$7.8 \cdot 10^3$	23	454	$\mathbf{7.0 \cdot 10^6}$
OPUSPAR.	$4.2 \cdot 10^7$	6	113	$1.5 \cdot 10^4$
PAWS	$3.6 \cdot 10^5$	20	289	$1.3 \cdot 10^6$
QQP	$1.5 \cdot 10^5$	10	141	$1.3 \cdot 10^5$

Table 4: **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.

On Table 4 we give a summary of the datasets we used. On one hand we have large datasets like MSCOCO or OPUSPARCUS (OPUSPAR.) 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) [18] is an approach that uses Metropolis-Hastings sampling [3] 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 5 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.

Corpus	Model	*BLEU*	*BERT*
MSCOCO	CGMH	**17.3**	0.7
	MCPG	16.5	**0.71**
	PTS	17.0	0.64
	PTS *upper-b.*	17.6	0.64
	PTS *random*	6.4	0.56
MSRPARA.	CGMH	9.7	0.48
	MCPG	39.3	**0.81**
	PTS	**40.3**	0.80
	PTS *upper-b.*	49.9	0.82
	PTS *random*	28.3	0.74
OPUSPAR.	CGMH	7.6	0.58
	MCPG	**9.6**	0.67
	PTS	9.1	**0.68**
	PTS *upper-b.*	14.5	0.68
	PTS *random*	4.3	0.57
PAWS	CGMH	15.4	0.61
	MCPG	55.5	**0.93**
	PTS	**57.9**	0.92
	PTS *upper-b.*	65.3	0.93
	PTS *random*	36.5	0.82
QQP	CGMH	22.5	0.72
	CGMH [18]	18.8	-
	MCPG	24.1	**0.78**
	PTS	**25.6**	**0.78**
	PTS *upper-b.*	32.7	0.79
	PTS *random*	9.8	0.66

Table 5: **Experiments summary.** Higher value are better. Best values are marked in bold. The PTS policy maximizes the MCPG criterion among the Pareto candidates. The PTS *random* policy picks a random candidate in the Pareto set. The PTS *upper-bound* (*upper-b.*) policy is "cheating" by taking a candidate that maximizes individual BLEU against the target. These two former policies are given to assess the "quality" of the generated set of candidates.

5 Discussion

We experimented two search-based approaches for paraphrase generation. These approaches are pragmatic and flexible, we think that they mimics well the way human do paraphrase. When compared against CGMH, another search-based and weakly-supervised method, our algorithms proved to be faster and more efficient.

The use of local PPDB transformations however lead our models to produce paraphrases that are a bit too conservative, especially when starting from short sentences. We plan to refine the scoring with deep reinforcement learning techniques and enrich the rules with more sophisticated patterns like phrases permutations. A planning-then-realization hybridization like the one proposed by [21] and [19] could also be considered for further works.

[2]We used the scripts from `github.com/jhclark/multeval` and `github.com/Tiiiger/bert_score`. For the BERT score we used 'bert-base-uncased_L8_no-idf_version=0.1.2'.

[3]This usage of BERT score against **reference/ground-truth** must not be confused with the BERT score against source sentence ($BERT_S$) that we use in the MCPG score.

References

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. "Finite-time Analysis of the Multi-armed Bandit Problem". In: *Mach. Learn.* 47.2-3 (2002), pp. 235–256. DOI: 10.1023/A:1013689704352. URL: https://doi.org/10.1023/A:1013689704352.

[2] Peter Auer et al. "The Nonstochastic Multiarmed Bandit Problem". en. In: *SIAM Journal on Computing* 32.1 (Jan. 2002), pp. 48–77. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539701398375. URL: http://epubs.siam.org/doi/10.1137/S0097539701398375 (visited on 11/28/2019).

[3] Nicholas Metropolis et al. "Equation of State Calculations by Fast Computing Machines". en. In: *The Journal of Chemical Physics* 21.6 (Dec. 2004). Publisher: American Institute of PhysicsAIP, p. 1087. ISSN: 0021-9606. DOI: 10.1063/1.1699114. URL: https://aip.scitation.org/doi/abs/10.1063/1.1699114 (visited on 06/29/2020).

[4] Chris Callison-Burch, Philipp Koehn, and Miles Osborne. "Improved Statistical Machine Translation Using Paraphrases". In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA: Association for Computational Linguistics, June 2006, pp. 17–24. URL: https://www.aclweb.org/anthology/N06-1003 (visited on 10/28/2019).

[5] Sanda Harabagiu and Andrew Hickl. "Methods for Using Textual Entailment in Open-Domain Question Answering". In: *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia: Association for Computational Linguistics, July 2006, pp. 905–912. DOI: 10.3115/1220175.1220289. URL: https://www.aclweb.org/anthology/P06-1114 (visited on 10/25/2019).

[6] Levente Kocsis and Csaba Szepesvári. "Bandit Based Monte-carlo Planning". In: *Proceedings of the 17th European Conference on Machine Learning*. ECML'06. event-place: Berlin, Germany. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 282–293. ISBN: 978-3-540-45375-8. DOI: 10.1007/11871842_29. URL: http://dx.doi.org/10.1007/11871842_29 (visited on 11/26/2019).

[7] Sylvain Gelly and David Silver. "Combining Online and Offline Knowledge in UCT". en. In: (June 2007). URL: https://hal.inria.fr/inria-00164003 (visited on 11/28/2019).

[8] Chris Callison-Burch. "Syntactic Constraints on Paraphrases Extracted from Parallel Corpora". In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics, Oct. 2008, pp. 196–205. URL: https://www.aclweb.org/anthology/D08-1021 (visited on 10/21/2019).

[9] Jonathan Chevelu et al. "Introduction of a new paraphrase generation tool based on Monte-Carlo sampling". In: *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 249–252. URL: https://www.aclweb.org/anthology/P09-2063 (visited on 10/21/2019).

[10] Joseph Olive, Caitlin Christianson, and John McCary, eds. *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. en. New York: Springer-Verlag, 2011. ISBN: 978-1-4419-7712-0. DOI: 10.1007/978-1-4419-7713-7. URL: https://www.springer.com/gp/book/9781441977120 (visited on 06/30/2020).

[11] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. "PPDB: The Paraphrase Database". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 758–764. URL: https://www.aclweb.org/anthology/N13-1092 (visited on 11/27/2019).

[12] Ellie Pavlick et al. "PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 425–430. DOI: 10.3115/v1/P15-2070. URL: https://www.aclweb.org/anthology/P15-2070 (visited on 10/21/2019).

[13] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". en. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: https://www.nature.com/articles/nature16961 (visited on 11/28/2019).

[14] Zhao Yan et al. "DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 516–525. DOI: 10.18653/v1/P16-1049. URL: https://www.aclweb.org/anthology/P16-1049 (visited on 10/25/2019).

[15] Daniel Cer et al. "Universal Sentence Encoder for English". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 169–174. DOI: 10.18653/v1/D18-2029. URL: https://www.aclweb.org/anthology/D18-2029 (visited on 11/26/2019).

[16] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: 2018. URL: https://arxiv.org/abs/1810.04805.

[17] Mohit Iyyer et al. "Adversarial Example Generation with Syntactically Controlled Paraphrase Networks". en. In: *arXiv:1804.06059 [cs]* (Apr. 2018). arXiv: 1804.06059. URL: http://arxiv.org/abs/1804.06059 (visited on 09/11/2019).

[18] Ning Miao et al. "CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling". en. In: *arXiv:1811.10996 [cs, math, stat]* (Nov. 2018). arXiv: 1811.10996. URL: http://arxiv.org/abs/1811.10996 (visited on 03/09/2020).

[19] Yao Fu, Yansong Feng, and John P Cunningham. "Paraphrase Generation with Latent Bag of Words". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 13645–13656. URL: http://papers.nips.cc/paper/9516-paraphrase-generation-with-latent-bag-of-words.pdf (visited on 02/17/2020).

[20] Tommi Gröndahl and N. Asokan. "Effective writing style imitation via combinatorial paraphrasing". In: *CoRR* abs/1905.13464 (2019). URL: http://arxiv.org/abs/1905.13464.

[21] Amit Moryossef, Yoav Goldberg, and Ido Dagan. "Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation". en. In: *arXiv:1904.03396 [cs]* (May 2019). arXiv: 1904.03396. URL: http://arxiv.org/abs/1904.03396 (visited on 07/01/2020).

[22] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". en. In: (2019), p. 24. URL: https://openai.com/blog/better-language-models/.

[23] Sam Witteveen and Martin Andrews. "Paraphrasing with Large Language Models". In: *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Hong Kong: Association for Computational Linguistics, Nov. 2019, pp. 215–220. DOI: 10.18653/v1/D19-5623. URL: https://www.aclweb.org/anthology/D19-5623 (visited on 11/25/2019).

[24] Tianyi Zhang et al. "BERTScore: Evaluating Text Generation with BERT". In: *arXiv:1904.09675 [cs]* (Apr. 2019). arXiv: 1904.09675. URL: http://arxiv.org/abs/1904.09675 (visited on 09/04/2019).