Quality Controlled Paraphrase Generation ++

Capstone Project
Madhav Sankar Krishnakumar†
† University of California, Los Angeles
madhavsankar@g.ucla.edu

I. Abstract

Paraphrase generation is a widely studied natural language generation task in NLP. Moreover, it is also widely used in a number of downsteam NLP tasks. Therefore, most NLP tasks benefit from high quality paraphrases, which are semantically similar but linguistically different from the original sentence. Generating high-quality paraphrases is challenging as it becomes increasingly hard to preserve meaning as linguistic diversity increases. Moreover, it becomes even more difficult to control the linguistic diversity aspects. For instance, two paraphrases might have high lexical diversity if we replace words in the sentences with their synonyms but such a paraphrase has very low syntactic diversity as the underlying parse tree structure is identical. Recent works achieve good results by controlling specific aspects of the paraphrase, such as its syntactic tree. However, even these do not allow to directly control the quality of the generated paraphrase, and suffer from low flexibility and scalability. Here we propose QCPG++, a quality-guided controlled paraphrase generation model, that allows directly controlling the quality dimensions in terms of different diversity metrics. This is built on top of IBM's QCPG model [1] by adding more text diversity metrics from TextDiversity framework [2][1]. We also extend IBM's QP model that identifies points in the quality control space that can produce good paraphrase results for a given sentence. The code and data can be found in https://github.com/madhavsankar/quality-controlled-paraphrase-generation. The models are uploaded to huggingface: https://huggingface.co/madhavsankar.

II. Introduction

Paraphrases are sentences that convey the same meaning but have use different words or sentence structure. Therefore, paraphrase generation rewrites a given sentence using different words, syntax or other forms of textual diversity while preserving its semantic meaning [3]. It is an important technique in natural language generation and has been critical in many other downstream NLP tasks including question answering [4], summarization [5], data augmentation [6] and adversarial learning [7]. But the usability of the paraphrases differ based on the task. The common goal in most cases is that the the generated paraphrase needs to be as diverse as possible from the original text and these diverse paraphrases can avoid the blandness caused by repetitive patterns [8].

[1]The project is built on top of Fabrice Harel-Canada's TextDiversity Framework and a number of diversity metrics from the work has been re-introduced for completeness.

A recent approach aiming to produce high quality paraphrases is controlled paraphrase generation, which exposes control mechanisms that can be manipulated to produce diversity. While the controlled generation approaches have yielded impressive results, they require providing the model with very specific information regarding the target sentence, such as its parse tree [7] or the list of keywords it needs to contain [9]. However, for most downstream applications, the important property of the paraphrase is its overall syntactic or lexical diversity, rather than conforming to a specific syntactic or lexical form. The over-specificity of existing control based methods not only complicates their usage and limits their scalability, but also hinders their coverage. Thus, it would be desirable to develop a paraphrase generation model, which uses a simple mechanism for directly controlling paraphrase quality, while avoiding unnecessary complications associated with specific controls.

In [1], the authors propose QCPG, a Quality Controlled Paraphrase Generation model, that given an input sentence and quality constraints produces a target sentence that conforms to the quality constraints. These constraints are much simpler than previously suggested ones, such as parse trees or keyword lists, and leave the model the freedom to choose how to attain the desired quality levels.

QCPG++ builds on top of QCPG to add more dimensions of textual diversity, thus making the paraphrase generation more fine-grained. We pass a vector of quality constrains in the form of phonological, morphological, lexical and syntactic diversities apart from semantic similarity. These various diversity metrics are obtained the unified framework of TextDiversity [2]. This work completes the above said TextDiversity project as a comprehensive coverage of textual diversity metrics at several key levels — semantic, lexical, morphological, syntactical, and phonological and then uses them to improve the coverage of QCPG++.

Enabling the direct control of the quality dimensions, allows flexibility with respect to the specific requirements of the task at hand, and opens a range of generation possibilities: paraphrases of various flavors (e.g. syntactically vs. lexically diverse or phonologically diverse but similar along other diversities), quasi-paraphrases (with lower semantic similarity), and even non-paraphrases which may be useful for downstream tasks (e.g. hard negative examples of sentences that are linguistically similar but have different meanings [10], [11]).

III. Method

This section provides an overview of the approach. We first introduce the various Text Diversity metric exposed by the TextDiversity framework. This is followed by the QCPG++ and how it utilizes these diversities as input control values.

A. Dimensions of Diversity

1) Semantic Diversity: Semantic Diversity highlights the difference in meaning and hence is the major metric that needs to be minimized for paraphrase generation. Figure 1 shows examples of semantic diversity calculated for two different sets of text. In 2a, the word "right" is repeated, thus reducing lexical diversity (only 7 unique words compared to 9 unique words in 2b). However, because the polysemous nature of "right" is reflected in the contextual embeddings, we can observe very little off-diagonal similarities in the similarity matrix. The opposite is true for 2b, where for example, "massive", "enormous", and "colossal" all semantically overlap, thus lowering the effective number of concepts present in the set. We use two semantic similarity scores:

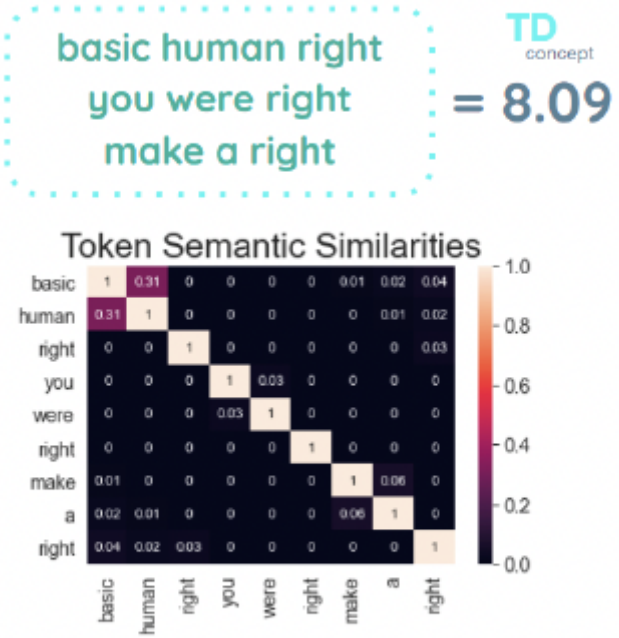
i. Bleurt Score: Given two sentences, s and s' the semantic similarity $q_{sym}(s, s')$ is defined as the Bleurt score [12], normalized using the sigmoid function to ensure a uniform range of values, [0, 1].

ii. DocumentSemanticDiversity: Understanding semantics at the document granularity makes it easier to identify paraphrases and redundant ideas regardless of syntax or lexical choices. Unlike traditional topic modeling [13] where one provides a parameter specifying a limited set of expected topics, we approximate topics via contextualized sentence embeddings output by SBERT [14]. Intuitively, these embeddings place the sentence at a specific point in semantic latent space and proximity to other points indicates related topicality [15] while avoiding the limitations of prior approaches [16], [17]. Given two sentences, s and s' the semantic similarity $q_{sym}(s, s')$ is defined as the pairwise cosine-similarity between the sentence embeddings of s and s'.

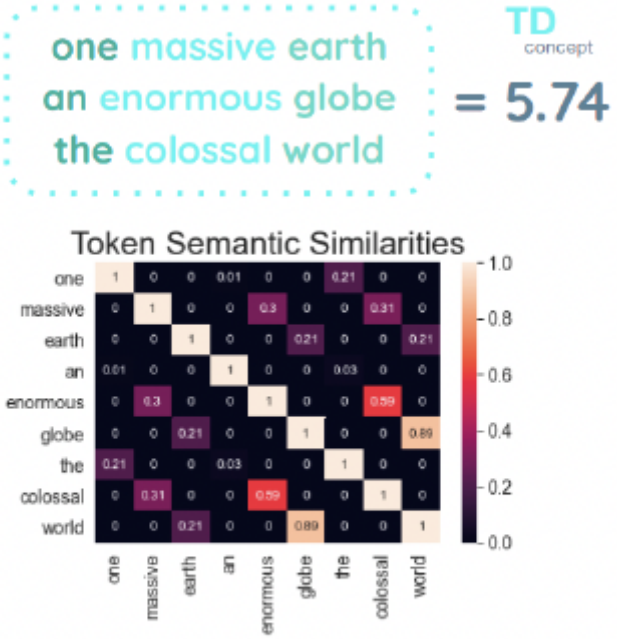
2) Lexical Diversity: Lexical Diversity is the measure of the degree to which two sentences have different word sets. Therefore, it is the distance between the bag of words of the underlying sentences.

Given two sentences, s and s', the lexical diversity $q_{lex}(s, s')$ is defined as the normalized character-level minimal edit distance between the bag of words. It is useful that this measure is independent of the word order to minimize the correlation between the various diversity input controls. Moreover, the token distances at character level enables to capture tokens that share the same lemma and is also found to be robust against noisy data.

3) Syntactic Diversity: Syntax is concerned with the arrangement of words and phrases to create well-formed sentences in a language. We use two different parse trees to measure syntactic diversity:



(a) **High Concept Diversity.** Despite lexical overlap (i.e. "right" is repeated twice), this set of text exhibits an effective number of 8.09 concepts.



(b) **Low Concept Diversity.** Despite lexical uniqueness (i.e. no repeated words), this set of text exhibits an effective number of 5.74 concepts due to redundant semantic content.

Fig. 1: Semantic Diversity.

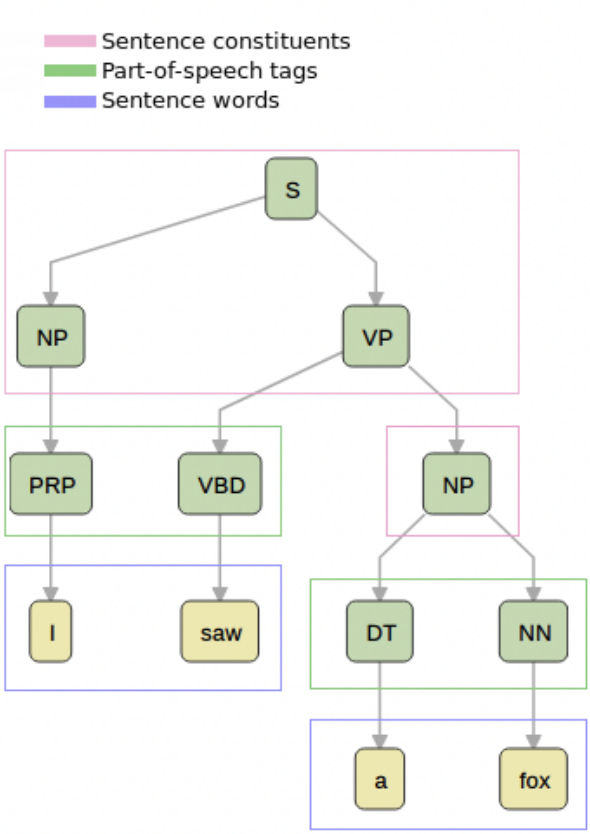


Fig. 2: Constituency Parse Tree.

i. Constituency Parse Tree: We propose to measure diversity at the constituency parse tree level [18] composed of nodes like noun phrases (NP) and verb phrases (VP).

Given two sentences, s and s', the syntactic diversity $q_{syn}(s, s')$ is defined as the normalized tree edit distance between the third level constituency parse trees of s and s' [7] after removing the tokens. Removal of token helps decouple the metric from lexical diversity.

Figure 2 depicts the constituency parse tree of the sentence "I saw a fox". We can observe how the words are grouped together in a hierarchical tree structure. For instance the "a fox" together is a Noun Phrase (NP) that consists of a determiner "a" and noun "fox".

ii. Dependency Parse Tree: We define dependency diversity where the relevant dependency parse trees [18] is composed of various relationship edges and types. Unlike constituency trees, dependency trees can represent non-adjacent and non-projective relationships in a sentence, which frequently appear in spoken language and noisy text [19].

Given two sentences, s and s', the syntactic diversity $q_{syn}(s, s')$ is calculated using a local degree profile (LDP) [20] as implemented by [21]. The LDP algorithm does not consider node or edge level attributes but quickly and efficiently generates a matrix embedding representing a batch of graphs and the syntactic diversity is given by the cosine similarity between the embeddings.

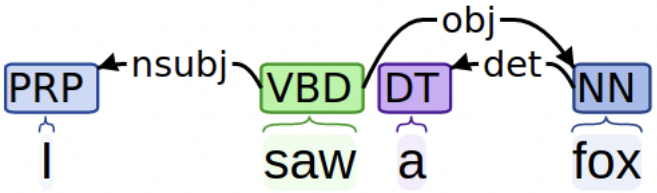


Fig. 3: Dependency Parse Tree.

Figure 3 depicts the dependency parse tree of the sentence "I saw a fox". We can observe how the dependencies between words are highlighted. One of the important aspects to be noted here is that dependencies can be between non-adjacent words which is not possible in the above constituent parse tree (refer Figure 2).

4) Phonological Diversity: Phonology is the branch of linguistics that deals with abstract systems of sounds. While reading text, humans engage in a process known as sub-vocalization, or reading with an "inner voice" where words are subconsciously annotated with phonological features that influence lexical access, comprehension, and cognitive load [22]. The sounds associated with words also impact human perception of beauty [23] and are at least partially predictive of lexical decline [24] – i.e. the timeless popularity of certain words while others fall into disuse. Maximizing phonological diversity during natural language generation seeks to hedge against monotony and increase the likelihood of hitting upon phrasings that enjoy more of the aforementioned benefits.

Given two sentences, s and s', the phonological diversity $q_{phon}(s, s')$ is defined as the rhythmic diversity between the two sentences. For our purposes, we regard rhythm as sequences of stressed and unstressed syllables and their associated weights—heavy and light, where the former are such that they "attract stress", while the latter only get stress if they happen to be in the right location in the string of syllables. Every sentence can be decomposed into a sequence of 4 rhythmic bases:

1) UL. Unstressed Light
2) UH. Unstressed Heavy
3) SL. Stressed Light
4) SH. Stressed Heavy

We implement this using the cadence package to extract stress features and the biopython package [25] to compute a pairwise sequence alignment and similarity score between rhythmic sequences. Figure 4 shows the rhythmic decomposition of sentences.

5) Morphological Diversity: Morphology is the study of how words are formed, analyzing their structures such as stems, root words, prefixes, and suffixes.

Given two sentences, s and s', the morphological diversity $q_{morph}(s, s')$ is defined as the diversity score between POS

Fig. 4: Rhythmic decomposition of sentences.

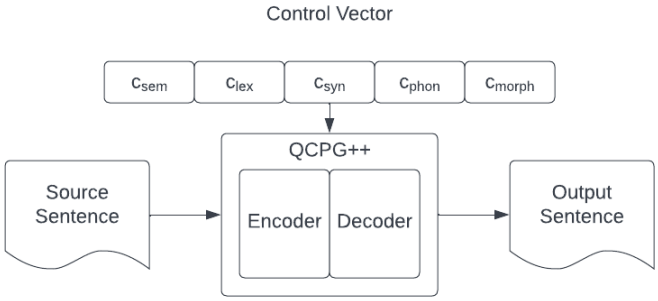


Fig. 5: QCPG++ Architecture.

tag sequences. Part of speech tags reflect the grammatical and morphological properties of words given their place and use in a sentence. We have 17 POS bases identified from the Universal Dependencies project [26]. To extract the POS tags, we use spacy [27] and then leverage the biopython package [25] to compute a pairwise sequence alignment and similarity score between POS tag sequences.

B. QCPG++ Model

The main component of our solution is a quality controlled paraphrase generation model (QCPG++), which is an encoder-decoder model trained on the task of controlled paraphrase generation. Given an input sentence s and a control vector $c = (c_{sem}, c_{syn}, c_{lex}, c_{phon}, c_{morph})$, the goal of QCPG++ is to generate an output paraphrase QCPG++(s, c) that conforms to c. We first preprocess the training data to find the textual diversity for the various training set pairs (s, s'), given by $q(s, s') = ((q_{sem}(s, s'), q_{syn}(s, s'), q_{lex}(s, s'), q_{phon}(s, s'), q_{morph}(s, s'))$. Each of the metric can be one of many possible metric options from the TextDiversity framework. We then train QCPG++ using the training set pairs (s, s'), by setting c to be q(s, s'), and maximizing $P(s'|s, c = q(s, s'))$ over the training set via the autoregressive cross entropy loss. In the inference stage, we can pass any given sentence and a vector of control values and the model generates an output sentence that adheres to the control values. Figure 5 shows the architecture.

C. Quality Predictor, QP

As expected it is not always possible to generate a paraphrase for all values of quality controls. For instance, a sentence like "United States has a GDP of 10 Trillion" may not be amenable to generate lexically diverse paraphrases as it contains a lot of named entities and numbers that cannot be replaced like "United States", "GDP" and "10 Trillion". Therefore, if we pass a quality vector with high lexical diversity for this sentence, it would result in poor quality paraphrases. Therefore, for effective use of QCPG++, we should first find the control vector range that would generate good paraphrases. Moreover knowing the typical control values that can produce good paraphrase could be vital for most downstream tasks.

For this purpose, we make use of IBM's Quality Predictor, QP [1] and extend it to make use of TextDiversity metrics. So given the source sentence s, QP produces the control vector r(s) that optimizes the expected paraphrase quality. QP makes couple of assumptions:

- The quality distribution $p(q|s)$ for all paraphrases obtained from source sentence s is normally distributed with mean $q_0(s)$.
- The difficulty to generate a paraphrase is decided by $p(q|s)$ and hence it is easiest for QCPG++ to generate a good quality paraphrase for s if the quality vector is $q_0(s)$. It becomes increasingly difficult as we move away from it. So given a control vector c(s), the difficulty can be expressed as $c(s) - q_0(s)$. We define this as the offset, o.

The Quality Predictor is a regressor produces the approximate value of the above said $q_0(s)$, for given sentence s. The model is trained for given pairs of sentence-paraphrase pairs s and t. We calculate the diversities between s and t to generate c(s, t). We assume this c(s, t) to be $q_0(s)$ and the model tries to predict the control vector c(s, t) for given s and t.

So the complete inference step when given a sentence is shown in Figure 6. As shown in Figure 5, the input to the paraphrase generation model, QCPG++, is composed of two elements: a source sentence s, and a quality vector $c = (c_{sem}, c_{syn}, c_{lex}, c_{phon}, c_{morph})$, which controls the quality of the generated paraphrase. Selecting appropriate values of c is crucial for obtaining high-quality paraphrases. The quality predictor model, QP, helps select suitable input quality vectors, by predicting the typical quality, $q_0(s)$, of the paraphrases of s. Lets call this approximation of $q_0(s)$ as r(s). The control vector c is the sum of r(s), and an offset vector o, which indicates the extent to which the requested quality deviates from the typical value.

IV. Implementation

A. Dataset

Paraphrase generation dataset can be obtained in multiple ways. One of the popular ways is to use back-translation, i.e. translate a given sentence to another language and then back-translate to obtain a paraphrase. Apart from this there are existing datasets that were originally created for other tasks, which can be preprocessed to be used as a paraphrase generation dataset. We have used MSCOCO dataset for our model. We have also trained on ParaBank2.0 and this model

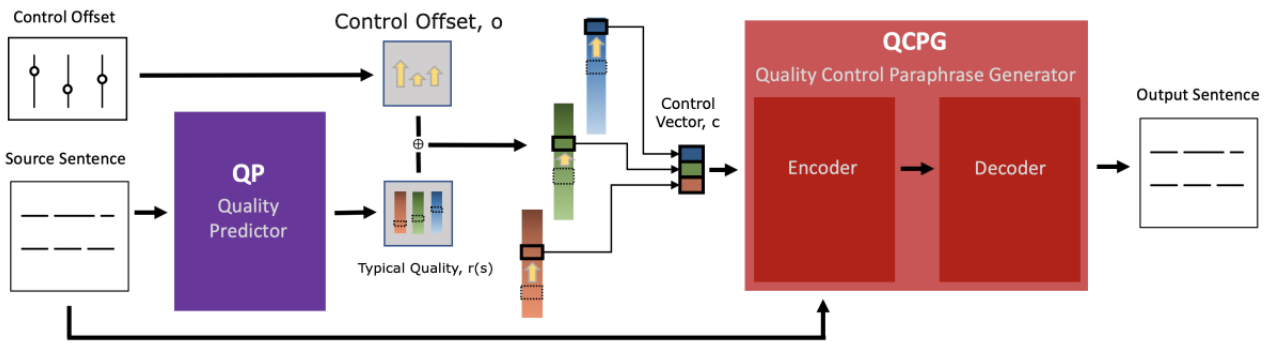


Fig. 6: Solution Architecture.

is used on another downstream task of Data Augmentation as part of another project.

MSCOCO: This dataset consists of 123K images, where each image contains at most five human-labeled captions [28]. We consider different captions of the same image as paraphrases and thus generate 10 paraphrase pairs for each image. Therefore, this leads to 1.23M paraphrase pairs, which are divided across training (900k), validation and test data.

ParaBank2.0: A dataset containing clusters of sentential paraphrases, produced from a bilingual corpus using negative constraints, inference sampling, and clustering [29]. The dataset is composed of avarage of 5 paraphrases in every cluster and close to 100 million pairs in total.

If the expected data is composed mainly of questions, then we can make use of the following dataset:

WikiAnswers: The WikiAnswers corpus contains clusters of questions tagged by wiki-answers.com users as similar. There are 30,370,994 clusters with 25 question in each on average. In total, the corpus contains over 70 million question pairs [30].

B. Implementation Details

Preprocessing: Preprocessing step first divides the data into train, validation and test. We have 900k training sentence-paraphrase pairs obtained from MSCOCO and around 14k validation and test pairs. We then calculate the various diversities for the sentence-paraphrase pairs using Google Colab. We append these various TextDiversity metrics as columns to the MSCOCO dataset.

1) QCPG++: **Base Model**: A Google Text-To-Text Transfer Transformer (T5) model [31] is used as the base model. T5-base is a pretrained language model that was trained on the Colossal Clean Crawled Corpus (C4) dataset and hence has learnt language structure, grammar, and semantics. As mentioned, it is a Language Model that looks at historical parts of sentence and predicts the next word in the sentence. The model was trained as a masked language model where words or phrases are masked and attempted to learn as targets. The main achievement of the model over other popular models like BERT is that is a text-to-text transfer model, which means it is a unified framework model which takes input and output as texts and can be easily adapted to solve a number of NLP tasks using transfer learning.

Training: The preprocessed data is used for training on the Google Colab GPU. We pass the 5 values of textual diversity metrics conforming to the phonological, morphological, lexical, syntactic and semantic diversity. These values are quantized and concatenated to the head of the input sentence and passed as an input to the T5 model. We maximize $P(s'|s, c = q(s, s'))$ over the training set via the autoregressive cross entropy loss. We use the SacreBLEU [32] implementation of the BLEU metric as our optimization metric.

We intend to repeat the training with several learning rates and choose the one that achieves the highest dev set performance.

Inference: We pass a source sentence and control vector as input to the trained T5 model. The control input vector to QCPG++ is quantized at every dimension into 20 equally spaced values ranging from 0 to 100. Each value is assigned to a special saved-token. The five tokens corresponding to the quantized values of the control vector c, are concatenated to the head of the input sentence, and together used as input to the model. The model generates an output sentence that conforms to the control vector.

Models: We trained two models. Model A consists of all TextDiversity metrics and Model B is built on top of QCPG by adding new diversity dimensions.

- Model A:
  - Semantic Similarity: DocumentSemanticDiversity
  - Syntactic Diversity: DependencyDiversity
  - Lexical Diversity: Character-level edit distance
  - Phonological Diversity: RhythmicDiversity
  - Morphological Diversity: POSSequenceDiversity.
- Model B:
  - Semantic Similarity: Bleurt Score
  - Syntactic Diversity: ConstituencyDiversity
  - Lexical Diversity: Character-level edit distance
  - Phonological Diversity: RhythmicDiversity
  - Morphological Diversity: POSSequenceDiversity.

2) QP: **Base Model**: An Electra base model [33], which uses a sample-efficient pre-training task called replaced token

detection. In this approach, some tokens are replaced with alternatives sampled from a small generator network and then a discriminative model is trained that predicts whether each token in this given corrupted input was replaced or not.

Training: The given source sentence and the five diversity metrics from the preprocessed data are passed to the Electra base model, which is finetuned with MSE loss to predict the typical quality values.

Inference: When the fine-tuned Electra model is given a source sentence, it predicts the typical quality values along the five diversity measures.

V. Results

A. Model A

```
Training Samples: 900000
Eval Samples: 14000

Text Diversity Metrics:
Semantic: DocumentSemanticDiversity
Syntactic: DependencyDiversity
Lexical: Character-level edit distance
Phonological: RhythmicDiversity
Morphological: POSSequenceDiversity.
```

1) QCPG++:

```
Baseline: t5-base
Learning Rate: 1e-4
Epochs: 6

i. Dataset: MSCOCO
[Used for all the evaluation in this paper]
Train Loss: 1.3403
Dev Loss: 1.811
Dev BLEU: 11.0279

ii. Dataset: ParaBank2.0
[For future work]
Train Loss: 0.4351
Dev Loss: 1.0986
Dev BLEU: 34.3115
```

2) QP:

```
Baseline: electra-base-discriminator
Dataset: MSCOCO
Learning Rate: 5e-5
Epochs: 3

Train Loss (MSE): 0.0127
Dev Loss (MSE): 0.0136
```

B. Model B

```
Dataset: MSCOCO
Training Samples: 900000
Eval Samples: 14000

Text Diversity Metrics:
Semantic: Bleurt Score
Syntactic: ConstituencyDiversity
Lexical: Character-level edit distance
Phonological: RhythmicDiversity
Morphological: POSSequenceDiversity
```

1) QCPG++:

```
Baseline: t5-base
Learning Rate: 1e-4
Epochs: 4

Train Loss: 1.4309
Dev Loss: 1.765
Dev BLEU: 11.7859
```

2) QP:

```
Baseline: electra-base-discriminator
Learning Rate: 5e-5
Epochs: 1

Train Loss (MSE): 0.0145
Dev Loss (MSE): 0.0138
```

VI. Evaluation

A. Effectiveness of the Quality Dimensions

The test data contains 14000 sentence-paraphrase pairs. We first calculate the various diversity metrics for each of these pairs, $q_{original}$. Then we pass these metric values as the Quality Dimensions for QCPG++. Both models generate corresponding paraphrases. Now we can calculate the diversity metrics for these paraphrases, q_{model}. We can calculate the RMSE between $q_{original}$ and q_{model}. This highlights how well the model's output confirms to the passes Quality Dimension values. This can be repeated for QCPG model as well.

As shown in table I, we can see that QCPG++ produces more accurate results for all the metrics that it is fine-tuned on. This is especially true when we check the DocumentSemanticDiversity value. As we add more dimensions, we can see that the output semantic similarity can be more accurately adjusted.

B. Responsiveness to the Quality Dimensions

We can use the test data and set the control vector, c to the corresponding diversity values for the given paraphrase. Now we can modify just one of the control variable at a time, keeping the rest constant and generate paraphrases. We can compare the diversity metrics of the generated paraphrase, q vector to the expected c. We can then average out the results obtained for the various test sentences and measure how responsive the paraphrases are to the quality dimensions. We have used Model A for the analysis as results should be similar for both models.

Metric	QCPG	Model A	Model B
Lexical	**0.0649**	**0.0639**	**0.0726**
Syntactic (Dep)	0.1252	**0.1215**	0.122
Syntactic (Cons)	**0.0849**	0.1313	**0.0836**
Phonological	0.1156	**0.099**	**0.1016**
Morphological	0.1291	**0.0971**	**0.0966**
Semantic (Doc)	0.1388	**0.0813**	0.1368
Semantic (Bleurt)	**0.1549**	0.1774	**0.1449**
BLEU	9.5594	9.4079	9.238

TABLE I: RMSE of the metrics when compared to the given test data. Bold rows highlight which metric was part of the training columns. Underlined highlights which model provides the most effective quality controls (lowest RMSE).

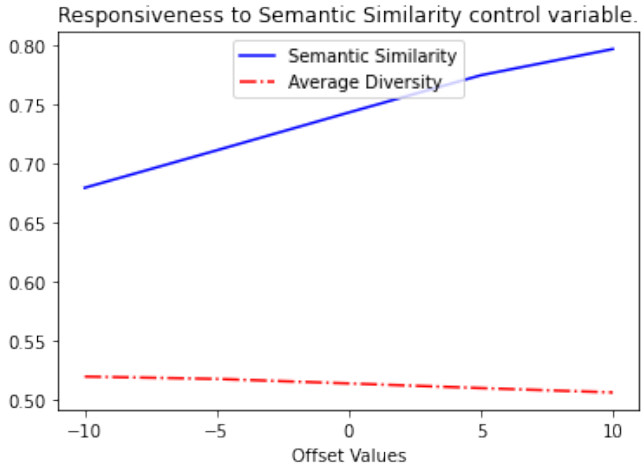


Fig. 7: Responsiveness to Semantic Similarity control variable.

1) Semantic Similarity: Given the control vector, c to be the corresponding diversity values of the test dataset paraphrases, we add an offset, o to the semantic similarity value in c. For instance, if the given test sentence is "A group of people wine tasting at a table." and the given paraphrase is "A group of people sitting at a table with glasses of wine.". We can calculate the diversity metrics between these 2 sentences to get c. For instance $c = (c_{sem}, c_{syn}, c_{lex}, c_{phon}, c_{morph})$ will be (72.9, 97, 27.5, 40, 40.7). Now we can add offset values of -10, -5, 0, 5 and 10 to the semantic similarity to obtain c_{sem} values of 62.9, 67.9, 72.9, 77.9 and 82.9. We can use QCPG++ to generate paraphrases with the new quality control vector and evaluate the generated paraphrase diversity metrics.

With increasing offset values, we are controlling the QCPG++ to generate paraphrases with increasing semantic similarity but with the same diversity. As can be observed in Figure 7, the results support the claim and QCPG++ is highly responsive to the semantic similarity dimension with minimal effect on the diversity dimensions.

2) Diversity Metrics: Similar to Semantic Similarity, we can add an offset to all the Diversity control variables as well and observe the effect on the QCPG++ paraphrases obtained. As can be observed in Figure 8, the diversity metrics increases with increasing offset, leading to a corresponding decrease in the BLEU evaluation metrics, as BLEU can be seen as a measure of similarity.

Note that as syntactic diversity of the test data was on an average at around 97%, a +10% offset would not make sense.

C. Best Possible Paraphrases for given diversity control vector

We define the best paraphrase as the one with the highest BLEU score (similarity) while maintaining the diversity as per the control vector. To find the best possible paraphrase we need to run inference on QCPG++ for various values of semantic similarity offsets starting from a base value. For the given test data, we can use the diversity metrics of the test paraphrase as the base. If this is not available, we can arrive at this base control vector using QP model.

As the offset increases, BLEU score increases but after a particular point, it is not possible to generate paraphrases with those semantic similarity control variables. We have observed the best BLEU score for offset of 20 and the score is 14.584895 as shown in Figure 9. For comparison, the QCPG model results for the same test data using the IBM's trained QP model was 11.803176.

We have used the above method of using test paraphrase diversity metrics as the control vector as our test data has paraphrases. Hence we are sure that paraphrases exist at those control vectors. For a new given sentence, we will have to use the QP model to arrive at a plausible control vector and then pass the same to QCPG++.

D. Comparing the TextDiversity metrics

We can train models using different TextDiversity metrics - Models A and B respectively. We set the control vector as the diversity metrics of the test data and then finetune the semantic similarity control. As shown in 9, the best BLEU for Model A is 14.58. On the other hand, as shown in 10, the best BLEU for Model B is 17.83. As the two models differ only in semantic and syntactic diversity measures, the main reason for the difference could be because of Constituency vs Dependency Parse Tree Syntactic Diversities. From the given data it was observed that Dependency Diversity was always very high in the range of 95-100% making it very restrictive. This could have affected the quality results.

E. Comparing the datasets

We can train the same QCPG++ model with TextDiversity metrics on both MSCOCO and ParaBank2.0 datasets. We compare the results obtained in II. We used glue-mrpc [34], [35] dataset for this purpose. We first filter out the paraphrases

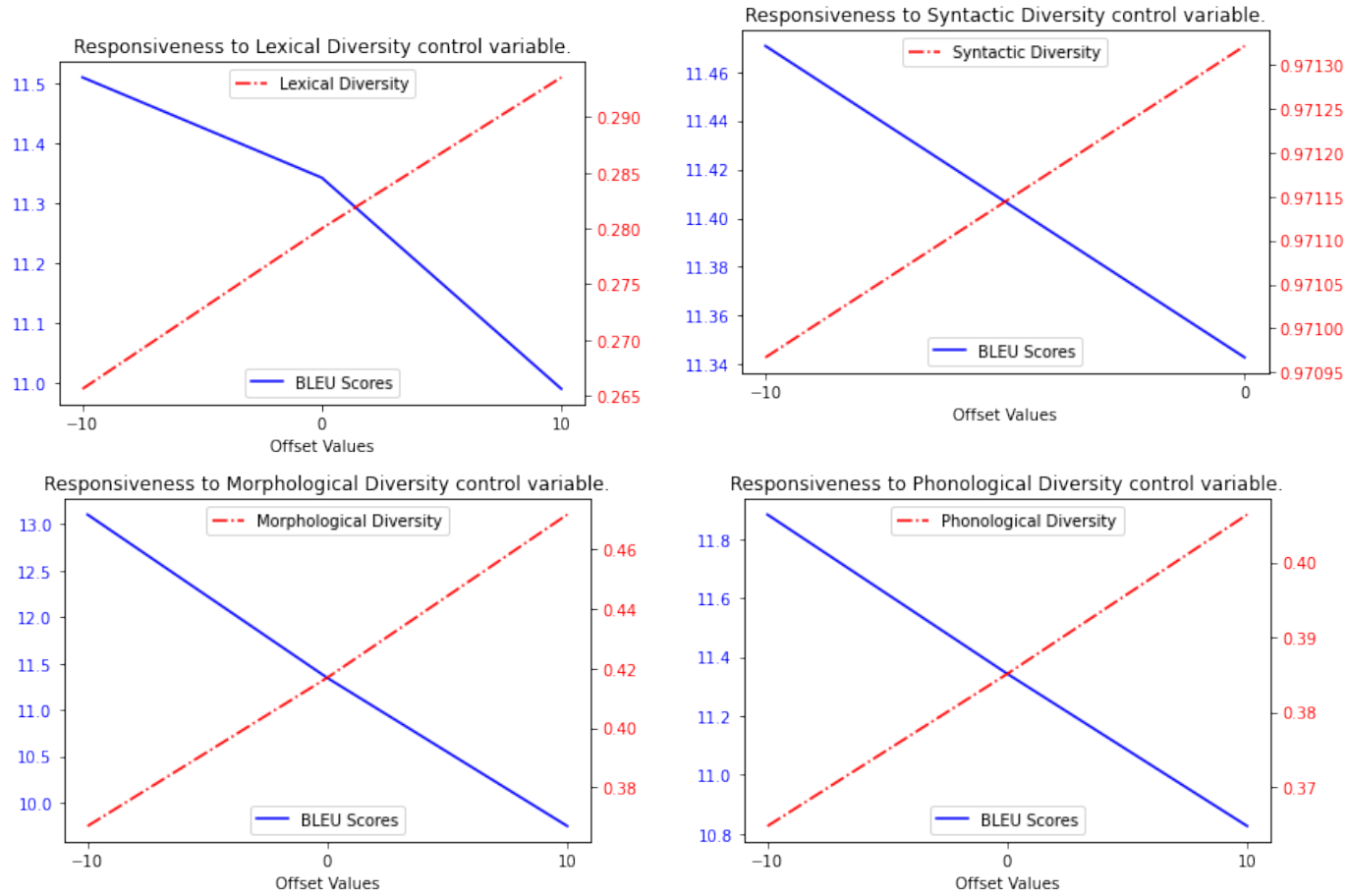


Fig. 8: Responsiveness to Diversity control variables.

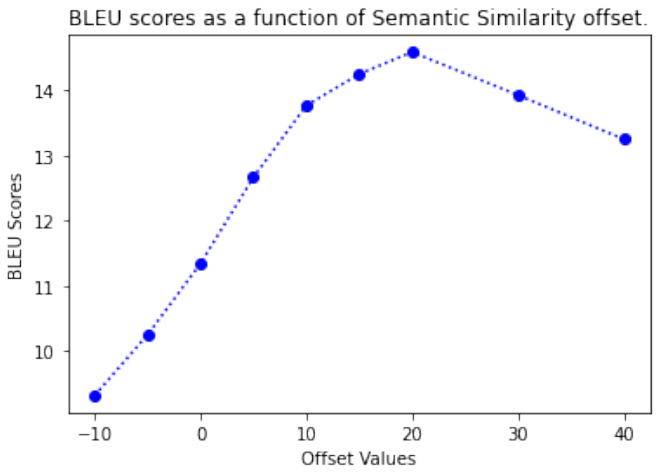


Fig. 9: BLEU Score as a function of Semantic Similarity control variable for Model A.

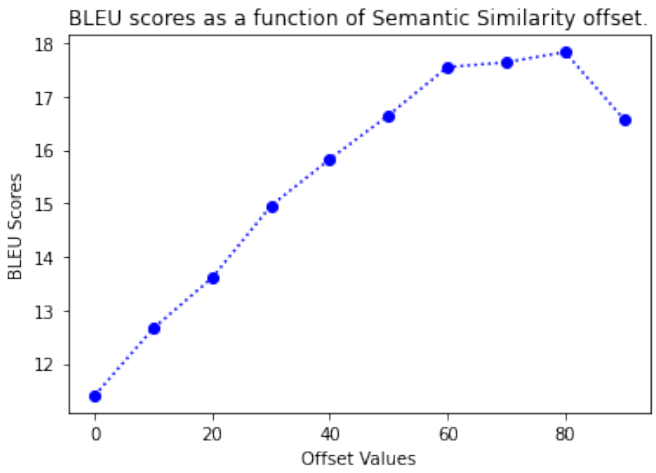


Fig. 10: BLEU Score as a function of Semantic Similarity control variable for Model B.

from the validation dataset and calculate the diversity metrics of the paraphrases. Then we pass these diversity metrics to our model as the control variables and try to generate paraphrases. We can then calculate the diversity metrics of the generated paraphrases and compare how well they are able to generate paraphrases for this new dataset.

As we can observe, ParaBank2.0 model is able to generate much better results than MSCOCO. The diversities are better

aligned to the input diversities and final paraphrase BLEU score is also higher. This can be because of the fact that MSCOCO dataset comprises of image captions and hence contains very few complete sentences and almost no questions. Therefore, it is specialized only for caption phrases. On the other hand, ParaBank2.0 is a more diverse dataset with complete sentences and hence it manages to extend well to a new dataset. Therefore, ParaBank2.0 based QCPG++ model will be a better fit for a generic downstream task.

Metric	Control	MSCOCO	ParaBank2.0
Lexical	0.1352	0.2917	0.1636
Phonological	0.2632	0.4741	0.3167
Morphological	0.2903	0.5273	0.3401
Syntactic	0.8266	0.9442	0.8775
Semantic	0.9055	0.5275	0.8914
BLEU	-	**4.8234**	**17.9163**

TABLE II: Average Diversity metrics.

VII. Conclusion and future work

In this paper, we propose a controlled paraphrase generation model, that generates paraphrases with desired quality along various textual diversity controls. We demonstrate the level of control that can be achieved on the model. We prove that high quality paraphrases can be obtained by fine tuning the results and also that the model is extremely responsive to the control variables.

A possible direction for future research is exploring methods, such as reinforcement learning, for further improving the ability of the model to satisfy the quality requirements. Also, we have trained only on captions, so the model is expected to perform poorly on longer sentences or questions. Re-training on more diverse datasets can help alleviate this problem.

REFERENCES

[1] E. Bandel, R. Aharonov, M. Shmueli-Scheuer, I. Shnayderman, N. Slonim, and L. Ein-Dor, "Quality controlled paraphrase generation," *arXiv preprint arXiv:2203.10940*, 2022.

[2] F. Harel-Canada, "Textdiversity: A unified framework for measuring and generating diverse language," 2022, to appear.

[3] R. Bhagat and E. Hovy, "What is a paraphrase?" *Computational Linguistics*, vol. 39, no. 3, pp. 463–472, 2013.

[4] A. Fader, L. Zettlemoyer, and O. Etzioni, "Open question answering over curated and extracted knowledge bases," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1156–1165.

[5] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.

[6] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv preprint arXiv:1809.08887*, 2018.

[7] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," *arXiv preprint arXiv:1804.06059*, 2018.

[8] K. Inui, J. Jiang, V. Ng, and X. Wan, "Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[9] D. Zeng, H. Zhang, L. Xiang, J. Wang, and G. Ji, "User-oriented paraphrase generation with keywords controlled network," *IEEE Access*, vol. 7, pp. 80 542–80 551, 2019.

[10] M. Guo, Q. Shen, Y. Yang, H. Ge, D. Cer, G. Hernandez Abrego, K. Stevens, N. Constant, Y.-H. Sung, B. Strope, and R. Kurzweil, "Effective parallel corpus mining using bilingual sentence embeddings," in *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 165–176. [Online]. Available: https://aclanthology.org/W18-6317

[11] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 4512–4525. [Online]. Available: https://aclanthology.org/2020.emnlp-main.365

[12] T. Sellam, D. Das, and A. P. Parikh, "Bleurt: Learning robust metrics for text generation," *arXiv preprint arXiv:2004.04696*, 2020.

[13] I. Vayansky and S. A. Kumar, "A review of topic modeling methods," *Information Systems*, vol. 94, p. 101582, 2020.

[14] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[15] N. Peinelt, D. Nguyen, and M. Liakata, "tbert: Topic models and bert joining forces for semantic similarity detection," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 7047–7055.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[17] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 113–120.

[18] D. Jurafsky and J. H. Martin, "Naïve bayes classifier approach to word sense disambiguation," *Computational Lexical Semantics*, 2009.

[19] M. R. A. H. Rony, L. Kovriguina, D. Chaudhuri, R. Usbeck, and J. Lehmann, "Rome: A robust metric for evaluating natural language generation," *arXiv preprint arXiv:2203.09183*, 2022.

[20] C. Cai and Y. Wang, "A simple yet effective baseline for non-attributed graph classification," *arXiv preprint arXiv:1811.03508*, 2018.

[21] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM, 2020, p. 3125–3132.

[22] M. L. Slowiaczek and C. Clifton, "Subvocalization and reading for meaning," *Journal of Verbal Learning and Verbal Behavior*, vol. 19, no. 5, pp. 573–582, 1980. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022537180906283

[23] D. Crystal, "Phonaesthetically speaking," *English Today*, vol. 11, no. 2, p. 8–12, 1995.

[24] D. Francis, E. Rabinovich, F. Samir, D. Mortensen, and S. Stevenson, "Quantifying cognitive factors in lexical decline," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1529–1545, 2021.

[25] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski *et al.*, "Biopython: freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[26] J. Nivre, M.-C. De Marneffe, F. Ginter, Y. Goldberg, J. Hajic, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira *et al.*, "Universal dependencies v1: A multilingual treebank collection," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, pp. 1659–1666.

[27] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017, to appear.

[28] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2556–2565.

[29] J. E. Hu, R. Rudinger, M. Post, and B. Van Durme, "Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6521–6528.

[30] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.

[31] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[32] M. Post, "A call for clarity in reporting BLEU scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 186–191. [Online]. Available: https://www.aclweb.org/anthology/W18-6319

[33] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[34] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[35] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2019, in the Proceedings of ICLR.

VIII. Appendix

A. Responsiveness to the Quality Dimensions

We fine tune each quality dimension by keeping the rest of the dimensions as the same value as the metric predicted in the test data. We assume the quality parameters as the diversity metric of the given test data and apply varies offset on them.

1) Semantic Similarity: Offset = -10:

```
set_diversity          0.289535
seq_diversity          0.499252
syn_diversity          0.324878
bleu_score             9.322824
bleurt_score           0.277784
phon_diversity         0.391749
morph_diversity        0.420967
semantic_similarity    0.679139
syn_dep_diversity      0.974286

RMSE:
semantic_similarity  :  0.09213771848560805
set_diversity  :  0.0659377295917991
syn_dep_diversity  :  0.1191823524600314
phon_diversity  :  0.0995725461111405
morph_diversity  :  0.09639508324074336
bleurt_score  :  0.16708929493417288
syn_diversity  :  0.1295044159640583
bleu_score  :  7.818781636641774
seq_diversity  :  0.09030533854418812
```

Offset = -5:

```
set_diversity          0.285367
seq_diversity          0.487872
syn_diversity          0.321665
bleu_score            10.251656
bleurt_score           0.313470
phon_diversity         0.388872
morph_diversity        0.420312
semantic_similarity    0.710925
syn_dep_diversity      0.974016

RMSE:
semantic_similarity  :  0.07774930786719676
set_diversity  :  0.06445465102946325
syn_dep_diversity  :  0.12272808706634031
phon_diversity  :  0.0986271686739251
morph_diversity  :  0.09653994490499199
bleurt_score  :  0.166569226255981
syn_diversity  :  0.13154790998126983
bleu_score  :  8.510726056846748
seq_diversity  :  0.0933548032028193
```

Test Data default (offset = 0):

```
set_diversity          0.280000
seq_diversity          0.475732
syn_diversity          0.320042
bleu_score            11.342888
bleurt_score           0.351114
phon_diversity         0.385101
morph_diversity        0.416813
semantic_similarity    0.742973
syn_dep_diversity      0.971321

RMSE:
semantic_similarity  :  0.0813948790456692
set_diversity  :  0.06397267299957513
syn_dep_diversity  :  0.1215377313619848
phon_diversity  :  0.09905506733794946
morph_diversity  :  0.09715125211505114
bleurt_score  :  0.17747668704337852
syn_diversity  :  0.13129380638555263
bleu_score  :  9.407900692775437
seq_diversity  :  0.0994998447779193
```

Offset = +5

```
set_diversity          0.273697
seq_diversity          0.460673
syn_diversity          0.317973
bleu_score            12.662787
bleurt_score           0.394864
phon_diversity         0.379799
morph_diversity        0.413119
semantic_similarity    0.774553
syn_dep_diversity      0.970840

RMSE:
semantic_similarity  :  0.10134140729474161
set_diversity  :  0.06615699890720199
syn_dep_diversity  :  0.12510089493813648
phon_diversity  :  0.10081351017450182
morph_diversity  :  0.09795996769033616
bleurt_score  :  0.20491873909832953
syn_diversity  :  0.13385637212478776
bleu_score  :  10.901257051091765
seq_diversity  :  0.11028617069971945
```

Offset = +10

```
set_diversity          0.267705
seq_diversity          0.448753
syn_diversity          0.315230
bleu_score            13.764193
bleurt_score           0.426880
phon_diversity         0.376756
morph_diversity        0.407660
semantic_similarity    0.796784
syn_dep_diversity      0.970547

RMSE:
semantic_similarity  :  0.14248668902193828
set_diversity  :  0.06984149993364774
syn_dep_diversity  :  0.12223692498682334
phon_diversity  :  0.10349531371360671
morph_diversity  :  0.09921774095624591
```

```
bleurt_score  :  0.25036418770787083
syn_diversity  :  0.13627405900153472
bleu_score  :  12.315727861821694
seq_diversity  :  0.12347644628289427
```

2) Lexical Diversity: Offset = -10

```
set_diversity           0.265609
seq_diversity           0.471015
bleu_score             11.510949
semantic_similarity     0.747582

RMSE:
semantic_similarity  :  0.08335085976262227
set_diversity  :  0.08061449916685852
bleu_score  :  9.720016376990369
seq_diversity  :  0.10548143963048576
```

Offset = 0

```
set_diversity           0.280000
seq_diversity           0.475732
bleu_score             11.342888
semantic_similarity     0.742973

RMSE:
semantic_similarity  :  0.0813948790456692
set_diversity  :  0.06397267299957513
bleu_score  :  9.407900692775437
seq_diversity  :  0.0994998447779193
```

Offset = +10

```
set_diversity           0.293550
seq_diversity           0.481733
bleu_score             10.989443
semantic_similarity     0.738738

RMSE:
semantic_similarity  :  0.08023273688012007
set_diversity  :  0.07192073737188874
bleu_score  :  9.03991372622984
seq_diversity  :  0.09811855287721787
```

3) Morphological Diversity: Offset = -10

```
bleu_score         13.104714
morph_diversity     0.367071

RMSE:
morph_diversity  :  0.11811839160697106
bleu_score  :  11.109605751426285
```

Offset = 0

```
bleu_score             11.342888
morph_diversity         0.416813

RMSE:
morph_diversity  :  0.09715125211505114
bleu_score  :  9.407900692775437
```

Offset = +10

```
bleu_score          9.745691
morph_diversity     0.471449

RMSE:
morph_diversity  :  0.10760239856591261
bleu_score  :  8.091531032406602
```

4) Syntactic Diversity: Offset = -10

```
bleu_score           11.470625
syn_dep_diversity     0.970967

RMSE:
syn_dep_diversity  :  0.12470975125909926
bleu_score  :  9.65463748521861
```

Offset = 0

```
bleu_score             11.342888
syn_dep_diversity       0.971321

RMSE:
syn_dep_diversity  :  0.1215377313619848
bleu_score  :  9.407900692775437
```

Offset = +10

```
bleu_score           13.737135
syn_dep_diversity     0.958849

RMSE:
syn_dep_diversity  :  0.14962587725415166
bleu_score  :  12.042334123297389
```

5) Phonological Diversity: Offset = -10

```
bleu_score        11.882409
phon_diversity     0.364744

RMSE:
phon_diversity  :  0.10629039119221839
bleu_score  :  10.071852869145086
```

Offset = 0

```
bleu_score             11.342888
phon_diversity          0.385101

RMSE:
phon_diversity  :  0.09905506733794946
bleu_score  :  9.407900692775437
```

Offset = +10

```
bleu_score        10.826873
phon_diversity     0.406396

RMSE:
phon_diversity  :  0.09926805425185906
bleu_score  :  9.016516292137753
```

B. Experiment with QP

Try the QP models for the corresponding QCPG or QCPG++ model and generate paraphrases using the obtained control values. As these values just ensure that good paraphrase sentences exists and doesn't guarantee best paraphrase results, these comparisons do not carry much value.

Metric	Test	QCPG	Model A	Model B
Lexical	0.281840	0.268988	0.280000	0.267869
Phonological	0.397592	0.379900	0.385101	0.380221
Morphological	0.427093	0.408133	0.416813	0.406293
Syntactic (Dep)	0.974789	0.970718	0.971321	0.974275
Syntactic (Cons)	0.341856	0.326554	0.320042	0.311328
Semantic (Doc)	0.708275	0.774616	0.742973	0.779463
Semantic (Bleurt)	0.292738	0.350551	0.351114	0.366043
BLEU	8.587361	11.803176	11.342888	11.127385

TABLE III: Average Diversity metrics.