Universitat Politècnica de Catalunya

Master in Artificial Intelligence

Paraphrase generation via round-trip machine translation

Human Language Engineering (MAI-HLE)

Authors:
Jordi Armengol
Marta Barroso

Course 2020-2021

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Abstract

The task of automatically generating paraphrases is relevant for a variety of reasons. First, they are interesting per se as a proof task for Natural Language Processing (NLP) models and linguistics. Second, rephrasing a sentence is a common task for all kind of writers. Last, but not least, it could be used for data augmentation in NLP, which is more challenging than in vision (since we cannot apply spatial transformations such as rotation or cropping). In this work, we explore the well-known technique of paraphrasing via round-trip machine translation from different perspectives. Specifically, we study the effect of using different families of languages, speech recognition and synthesis systems, and multiple translation cycles. Apart from manually inspecting the generated paraphrases for an intrinsic qualitative analysis, we evaluate them with different intrinsic quantitative metrics, and use them as data augmentation in a sentence classification task, as an extrinsic evaluation.

Contents

1 Introduction and Contributions **3**

2 Background and Related Work **3**

3 Methods and Experiments **4**
3.1 Paraphrasing Methods 4
3.2 Evaluation 5

4 Experimental framework **6**

5 Results and discussion **7**
5.1 Qualitative analysis 7
5.2 Quantitative analysis 8

6 Conclusions and Future Work **11**

A Implementation **13**
A.1 Input 14
A.2 Output 14
A.3 Translator 14
A.4 Paraphraser 15
A.5 Evaluator 16

1 Introduction and Contributions

The task of automatically generating paraphrases is relevant for a variety of reasons. First, they are interesting per se as a proof task for Natural Language Processing (NLP) models and linguistics. Second, rephrasing a sentence is a common task for all kind of writers. Last, but not least, it could be used for data augmentation in NLP, which is more challenging than in vision (since we cannot apply spatial transformations such as rotation or cropping).

In this work, we explore the well-known technique of paraphrasing via round-trip machine translation from different perspectives. This method is defined as follows. Assume a machine translation system $A \rightarrow B$, for translating from the language A to the language B, and a machine translation system $B \rightarrow A$, for the reverse direction[1]. If we have a sentence in the language of interest (A), a, we can first translate it into B with the A $\rightarrow$ B system. Then, we could translate the translated sentence, b, back into A via the $B \rightarrow A$ system. Since translation is potentially noisy, when trying to recover the original sentence, a, one will obtain $a\prime$, with $a \approx a\prime$, and potentially $a \neq a\prime$. If the translation is bad enough such that the original sentence cannot exactly be recovered, nut good enough not to lose the original meaning, $a\prime$ is an automatically generated paraphrase of a. For being noisy, the translation does not necessarily have to be inexact. The mere diversity of natural languages, especially if they are from different families, can generate translations with the exact same meaning but with different words.

Specifically, we study the effect of using different families of languages. We also generalize the round-trip translation to inter-modal transformations with speech recognition and synthesis systems, and multiple translation cycles. Apart from manually inspecting the generated paraphrases for an intrinsic qualitative analysis, we evaluate them with different intrinsic quantitative metrics, and use them as data augmentation in a sentence classification task, as an extrinsic evaluation.

In Section 2, we describe the background and related literature. Then, in Section 3, we describe our proposed methodology. Finally, in sections 5 and 6, we present the obtained results and arrive to conclusions, respectively.

2 Background and Related Work

The task of automatically generating paraphrases is well-studied in the NLP literature. Furthermore, the specific methodology of doing so via round-trip machine translation is also well-known since at least the Statistical Machine Translation (SMT) era [1].

Regarding alternative methods, in [2] they propose generating paraphrases with stacked LSTMs with residual connections. They show that stacking of residual LSTM layers is useful for paraphrase generation but it needs to be remarked that it may not perform equally well for machine translation because not every word in a source sequence needs to be substituted for paraphrasing. Residual connections help retain important words in the generated paraphrases.

In [3], authors suggest joint copying and restricted generation for creating paraphrases. In particular, they define a model that consists of two components: a copying decoder and a restricted generative

[1] Obviously, a multilingual system with both directions could play both roles.

decoder. To combine the two decoders and determine the final output, they train a predictor to predict the writing modes (copying and rewriting). Finally, the approach in [4] consists of a deep generative framework with sequence-to-sequence models (LSTM) for generating paraphrases.

As far as the round-trip machine translation method itself is concerned, apart from the works in the SMT era we said [5], more recent works have kept deepening on the topic, with a great deal of creativity. It has been shown that *backtranslation*[2] can be effectively used as data augmentation in Neural Machine Translation (NMT) [6]. More recently, the exact same technique but using vision models (image captioning and synthesis) has been shown to be effective to generate multiple paraphrase sentences. In [7], the authors introduce a novel way of image-based paraphrasing that allow to incorporate additional knowledge from image to the translation process. Furthermore, they implement an automatic paraphrase generation model, and use it with a multi-expert approach within NMT. Following the same technique, [8] proposed to construct neural paraphrase models which initiate expert models and utilize them in NMT. They diffuse the image information by using image-based paraphrasing without using the image itself.

Recently, some researchers started focusing on improving robustness of NMT models when translating noisy texts. For instance, [9] propose new data augmentation methods to extend limited noisy data. Further, they manage to improve NMT robustness to noise while keeping the models small. Additionally, they explore the effect of utilizing noise from external data in the form of speech transcripts and show that it could help robustness.

Like [2], [3], and [4], we want to automatically generate paraphrases. Unlike them, but like [1], [7], [8], we want to use the round-trip method. The same way as [9], we are also going to experiment with speech models. Similarly to [9] (and even [6]), we propose to use these paraphrases as data augmentation. However, in our case we will target sentence classification (instead of machine translation). All in all, we base our work on the existing related work we mentioned, which has been very productive in this topic, but combining text machine translation of different language families with speech systems, and experimenting with multiple cycles. In addition, we introduce a new intrinsic metric for evaluating paraphrases, unlike previous works (such as [4], which basically use BLEU score [10], the most widely used MT metric).

3 Methods and Experiments

In this section, we will first describe the proposed methods, and then specify how we plan to validate them.

3.1 Paraphrasing Methods

We base our work on the well-known round-trip machine translation paraphrasing method, described in Section 1. We introduce, though, some perspectives and extensions.

Translation paths By *translation path* we mean a series of translations that start in a given language and end in another one. If the path starts and ends in the same language, it is a *cyclic*

[2]Backtranslation is a procedure whereby sentences that were previously translated into another language are re-translated to the original one. This can be used for generating synthetic parallel sentences from monolingual corpora.

translation path. For building paraphrases, we need a cyclic translation path starting from and finishing in the language of interest. This can be obtained as follows:

- Round-trip machine translation: In vanilla round-trip machine translation, we use the method described in Section 1. If the language of interest is A, and we have translation systems for the directions $A \rightarrow B$ and $B \rightarrow A$, we can translate sentences from A into B, and then back into A, to hopefully obtain paraphrases with a certain quality.
- Intermediate (or pivot) machine translation: For increasing text diversity, we could translate from A into B, then from B into C, and finally from C into A. This can be generalized to many different paths, with multiple languages in between.
- Multi-cycle machine translation: The cycles defined in the vanilla round-trip machine translation and intermediate machine translation can be repeated N times, which also could increase text diversity.

Translators With the translation-based paraphrasing methods we described, we can use any machine translation. We propose using systems for the following cases:

- Different families: If the used languages belong to the same family, the recovered sentences might be identical or at least too close to the original sentences, which would make the paraphrases worthless. For this reason, we study the effect of using languages from different families, especially if the languages are considerably different.
- Speech: We generalize the round-trip translation to inter-modal transformations with speech recognition and synthesis systems. Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) can be considered as special cases of translation, and we can apply the same paraphrasing methods we saw. We note that the same method could be applied with image synthesis and captioning, or with speech translation (instead of the mere transcription). Our motivation for using different modalities is that we believe that the recovered sentences will be more diverse, since there will be more noise.

3.2 Evaluation

For analyzing and validating the generated paraphrases, we evaluate them both qualitatively and quantitatively. In the case of the quantitative analysis, we do so both for intrinsic and extrinsic evaluations.

Qualitative analysis As a qualitative analysis, we suggest to manually inspect a randomly sampled set of the generated paraphrases. This analysis will not be enough for extracting conclusions, but it can allow a better understanding of the proposed system, and point to future research lines.

Quantitative analysis We suggest the following metrics for the quantitative analysis:

- Intrinsic evaluation: We collect a number of metrics, including 1. BLEU [10], a well-known metric in machine translation that used to compare the n-gram similarity between a generated sentence and a target reference; 2. Edit distance (and normalized edit distance), as a character-level distance measure; 3. Jaccard distance, a set-level distance (in this case applied to words); and 4. Cosine similarity of sentence embeddings. Character and word-based distances should be as high as possible (since we want the paraphrases to, indeed, use different words)

at least up to a point (if they are too different, the paraphrase might have lost the original meaning). The embedding cosine similarity should be as high as possible too, in this case without any nuances (the higher, the better, without restrictions). We define a new metric, `Jaccard-Embedding Factor`, as the product between the Jaccard distance and the cosine similarity of the sentence embeddings of a given paraphrase and the original sentence. Since both metrics range from 0 to 1, so does our derived metric. An ideal paraphrase would have the maximal Jaccard distance and cosine similarity, that is, `Jaccard-Embedding Factor` = 1, so we want to maximize this metric. For the intrinsic evaluation, we compute the mentioned metrics for each individual paraphrase, and report basic statistics (mean, standard deviation) for each paraphrase set.

- Extrinsic evaluation: As extrinsic evaluation, we suggest using the generated paraphrases as data augmentation in a sentence classification task. Specifically, we take an open-source implementation, and tweak it to include data augmentation. We compare the results with and without data augmentation.

4 Experimental framework

We implement the paraphrasing methods from scratch in Python, focusing on readability and extendability (making use of class inheritance, for instance), as detailed in Appendix A.

Due to time and computational constraints, for the experiments we only use the vanilla round-trip machine translation and an execute once the multiple cycles variant, (even if the implementation of other methods works), and we leave the execution of the other implemented methods as future work. Regarding the translators used as the back-end for the paraphrasing, we discard the FAIR models for the experiments, being too slow, and instead use the Marian ones. Specifically, we use the Marian systems for the pairs corresponding to German, Icelandic, and Russian, being representative of different families. All these models are based on the Transformer [11] architecture, a purely attentional architecture considered the state-of-the-art in neural machine translation. In the case of the Speech, we use the mentioned Tacotron[12] Text-To-Speech and Silero[3] Speech-To-Text models[4], even if computationally expensive, because we are very keen on studying inter-modality.

For the intrinsic evaluation, we use NLTK's[5] word tokenizer, BLEU, and Jaccard distance implementation. We take the sentence embeddings from the implementation of SentenceBERT [13], a set of BERT-like [14] models supervisedly fine-tuned for producing high-quality sentence embeddings in the special `[CLS]` token[6].

For the extrinsic evaluation, we adapt an open-source PyTorch [15] implementation of a sentence classifier[7], using the data from a sentence classification competition in Kaggle[8]. This dataset consists of around 8k tweets that have to be classified into tweets about a real natural disaster and tweets that are not about a real natural disaster. We train the same model, based on 1D convolutional

[3] `https://pytorch.org/hub/snakers4_silero-models_stt/`

[4] Both based on convolutional layers.

[5] `https://www.nltk.org/`

[6] The reason why we use SentenceBERT, instead of vanilla BERT, is that the sentence embeddings obtained in BERT are not especially useful per se, unless fine-tuned. For further explanation, please refer to the SentenceBERT article [13].

[7] `https://github.com/FernandoLpz/Text-Classification-CNN-PyTorch`

[8] `https://www.kaggle.com/c/nlp-getting-started`

layers and pooling, for 10 epochs, and report accuracy in the test set. We use the same split as the original implementation, 75% for the train set and 25% for test, and only inject paraphrases from the train set, once the split has already been done.

Appendix A has more details about the implementation of both the intrinsic and extrinsic evaluations.

5 Results and discussion

In this section, we provide the results of the evaluation.

5.1 Qualitative analysis

We refer to the description of human evaluation of machine translation in [16], and try to evaluate the generated paraphrased in terms of their *adequacy* (i.e., whether the original meaning is respected), and *fluency* (correctness of the paraphrases as English sentences themselves). We use these criteria defined for machine translation evaluations, but for evaluating the paraphrases (at the end of the day, they are also generated by machine translation systems). Here, we do not assign numerical values to the paraphrases, but just comment the adequacy and fluency that we perceive. Also, we evaluate whether the generated paraphrases are too close to the original sentence.
We first randomly sample 10 sentences from the Tweets dataset. Then, we make the following observations for the first sentence, regarding the generated sentences by the paraphraser systems described above (here, we only include the vanilla round-trip ones):

- En-De: `@flowri did you marinate it or was it an accident?`. The sentence is grammatically correct, fluent and maintains the same meaning than the original sentence but the words of the sentence are almost equal to the original one. Thus we do not consider that this sentence is a proper paraphrase. We believe that the main reason for which this happens is the high similarity between English and German.

- En-Is: `Did you finish it, or was it an accident?`. The sentence is grammatically correct and fluent, but the meaning has been altered. The context of the original sentence is related to cooking, more precisely to marinade. This information is lost when the verb is substituted by "finish". Moreover, the nickname is lost as well.

- En-Ru: `@Flowri docked or was it an accident?`. In this case we observed a similar situation to the one using Icelandic. The verb has been modified by another that is not equivalent. Although the sentence is correct grammatically, the meaning is not satisfied.

- Speech: `floray were you marinating it or was it in accident`. Since the language is the same, only the medium (from text to speech and vice-versa) we observe almost no change between the original sentence and the paraphrase. The interrogation punctuation is lost. We suspect that the model Tacroton does not detect questions when converting from text to speech. In general, punctuation is usually lost in this system.

We have observed some cases in which the paraphrases are equal for all the languages. It is the case with short sentences that have a simple structure, i.e only one verb, no verbal complements, no punctuation and numbers or dates. This phenomena is observed for the sentence `I am a wreck`. However, there are a few cases in which the paraphrase is not correct either because it is incomplete,

because it is not grammatically correct or because the meaning is not the same as the original phrase. For the sentence `Attack on Titan game on PS Vita yay! Can't wait for 2016` we obtain:

- En-De: `Attack on Titan game on PS Vita yay! Can't wait 2016`. The sentence is the same as the original one. This is a typical problem when both languages are similar.
- En-Is: `I can't wait for 2016.` Although the sentence is correct, it is incomplete since the first part of the sentence is missing.
- En-Ru: `The attack on Titan's game at PS Vita Yay!`. We observe the same situation but this time is missing the second part.
- Speech: `attack on tghtan game on vita ya can't wait for twenty sixteen`. When the sentence includes punctuation, own names and numbers we can see that this model has difficulties to perform the translation. Then, we consider that part of information is lost as well.

In general, we observe that, indeed, the round-trip machine translation can generate paraphrases. In the case of the German system, the sentence is not unlikely to be identical to the original one. More than paraphrasing, the German systems serves as a sort of *normalization* system. For instance, "I'm" is converted to "I am". In the case of Icelandic and Russian, the original system is sometimes lost. In the case of speech, punctuation and casing is always lost. Also, sometimes the original words are replaced by random words. As per the quantitative analysis that we will see in Section , we could infer that the kind of information that is lost in the Speech system (e.g., punctuation) is not relevant, and that the diversity introduced by the sort of random replacement of words (caused by the speech generation and transcription) will be useful for data augmentation.

5.2 Quantitative analysis

Intrinsic evaluation Table 1 shows the results of the intrinsic quantitative metrics we collected. Indeed, the metrics show that paraphrases coming from the German model, being a language close to English, are the ones the most similar with the original sentences. Nevertheless, these makes them potentially worse paraphrases, due to using too similar words to the original ones. When compensating the word-level string similarity with our custom metric, Jaccard-Embedding-Factor, we observe that the paraphrases coming from the German model are the worse ones. The ones coming from the Icelandic, Russian, and Speech models are considerably better.

As per our metric, the ones coming from the Speech model are the best ones, even if with a small distance with respect to Icelandic and Russian, which makes it difficult to extract conclusions between these three ones. What we can say, though, is that in the case of paraphrases from the Russian model, being closer to German than Icelandic, the generated sentences are, indeed, closer to the original ones, when observing both the string and embedding distances. However, in Jaccard-Embedding-Factor, sentences from the Icelandic model compensate its decreased embedding similarity with bigger string-level distances.

The case of the paraphrases coming from the Speech model is curious. On the one hand, the language belongs to the same family (obviously; it is also English). On the other hand, the change of modality introduces noise. We observe that these paraphrases are the ones the most *balanced*. That is, if we observe the individual metrics, in all of them it is never the *worse*[9]; it always performs

[9] As per our interests in obtaining paraphrases; none of these metric is necessarily better or worse per se.

competitively. Finally, it has the best mean Jaccard-Embedding-Factor.

Model	bleu	emb. cos. sim.	norm. edit dist.	jaccard	jaccard-emb.-factor
German	0.77 (0.17)	0.88 (0.11)	6.18 (18.87)	0.38 (0.22)	0.32 (0.17)
Icelandic	0.50 (0.20)	0.70 (0.19)	12.37 (28.90)	0.64 (0.17)	0.43 (0.13)
Russian	0.58 (0.20)	0.78 (0.15)	12.36 (30.87)	0.59 (0.18)	0.45 (0.13)
Speech	0.65 (0.20)	0.71 (0.16)	9.97 (5.50)	0.67 (0.21)	**0.46 (0.16)**

Table 1: Intrinsic quantitative metrics for the vanilla round-trip machine translation paraphrasers: BLEU (BLEU score), Emb. Cos. Sim. (SentenceBERT Sentence Embedding Cosine Similarity, Norm. Edit Dist. (Normalized Edit Distance), Jaccard (Jaccard distance), and Jaccard-Emb.Factor (Jaccard-Embedding-Factor, our custom metric for maximizing both word-level string distance and cosine similarity) for paraphrases generated with German, Icelandic, Russian, and Speech models. We provide the mean and standard deviation (between parenthesis) of these metrics. As per all collected metrics, paraphrases coming from the German model, being a language close to English, are the ones the most similar with the original sentences. These makes them potentially worse paraphrases, due to using too similar words to the original ones. In this evaluation, we use Jaccard-Embedding-Factor for selecting the best paraphrases, the ones coming from the Speech model.

For studying the effect of using multiple round-trip machine translation cycles, we take the best paraphraser as per the intrinsic metrics in the vanilla round-trip experiments, that is, the Speech one, and use it with 2 cycles. That is, En $\rightarrow$ En_speech $\rightarrow$ En $\rightarrow$ En_speech. Table 2 shows the same metrics but comparing these two systems. We can see that, as expected, the transcription quality degrades (BLEU, Edit distance). Contrary to our expectations, though, this word (or string) level difference comes at the expense of a considerably less similarity in terms of embedding cosine similarity, so one does not compensate the other. Thus, the resulting Jaccard-Embedding-Factor is worse when using multiple cycles. The multiple cycles do not seem to be worth the computation.

Model	bleu	emb. cos. sim.	norm. edit dist.	jaccard	jaccard-emb.-factor
Speech	0.65 (0.20)	0.71 (0.16)	9.97 (5.50)	0.67 (0.21)	**0.46 (0.16)**
Speech N=2	0.62 (0.20)	0.66 (0.17)	10.47 (5.54)	0.69 (0.21)	0.44 (0.15)

Table 2: Intrinsic quantitative metrics comparing paraphrasing methods: BLEU (BLEU score), Emb. Cos. Sim. (SentenceBERT Sentence Embedding Cosine Similarity, Norm. Edit Dist. (Normalized Edit Distance), Jaccard (Jaccard distance), and Jaccard-Emb.Factor (Jaccard-Embedding-Factor, our custom metric for maximizing both word-level string distance and cosine similarity). We take the best vanilla round-trip machine translation paraphraser, the Speech one, and then compare it with a paraphraser also based on Speech but with 2 translation cycles. As per the Jaccard-Embedding-Factor, the vanilla round-trip method outperforms the one with 2 cycles.

Extrinsic evaluation Table 3 shows the results of the extrinsic evaluation, using the generated paraphrases as data augmentation in the sentence classification task from the Kaggle competition we mentioned. Using the generated paraphrases as data augmentation does not generally seem to help. In fact, the performance degrades, except the case of the ones coming from the Speech models. We are unsure about the reasons of this general bad performance. Perhaps, many sentences are too

close to the original ones and just add noise.

We note, though, that for easing the comparison, all the runs were executed with the exact same hyperparameters as the baseline. This means that configurations with more data, and therefore, potentially needing more time to converge, were trained with the exact same number of epochs than the baseline, which might not be optimal for the settings with data augmentation. We did found one important detail, which is that shuffling is key, at least for paraphrase-based data augmentation. In the first run of the extrinsic evaluation, we obtained worse results. After inspecting the code, we found that paraphrases were set in the same batch as their respective original sentences, which made batches less diverse. Shuffling the data *after* the data augmentation (but, obviously, before the train-valid-test split, which is always the same) slightly improved the results of the systems with data augmentations (the ones reported in Table 3 are with shuffling after data augmentation). We hypothesize that this is the case because of the increased in-batch diversity.
Remarkably, the order of the results obtained in the extrinsic evaluation is *the exact same order that the one obtained in the intrinsic evaluation*, in Table 1, that is, German, Icelandic, Russian, and Speech, from worse to better results. This hints at the potential of our proposed metric for evaluating paraphrases without having to run an extrinsic evaluation. Obviously, being a single task, and with only 4 kinds of generated paraphrases, we cannot extract many conclusions, but the results are promising.

Model	test accuracy
Baseline	0.735
German	0.661
Icelandic	0.678
Russian	0.686
Speech	**0.748**

Table 3: Test accuracy in the extrinsic evaluation (the sentence classification task from a Kaggle competition based on tweets we said) for the vanilla round-trip machine translation paraphasers: Using the generated paraphrases as data augmentation does not seem to help (in fact, the performance degrades), except the case of the ones coming from the Speech models. Remarkably, the order of the results obtained in the extrinsic evaluation is *the exact same order that the one obtained in the intrinsic evaluation*, in Table 1, that is, German, Icelandic, Russian, and Speech, from worse to better results.

As in the intrinsic evaluation, we now take the best vanilla round-trip machine translation paraphraser as per the extrinsic evaluation (again, the Speech one), and study the effect of using 2 cycles instead of just one, this time in the extrinsic evaluation. Table 4 shows the extrinsic evaluation results comparing these two systems. As in the intrinsic evaluation, we observe that using 2 cycles (instead of just one) does not seem to be useful, contrary to our expectations. The resulting 2-cycle system performs almost identically as the baseline (the system without data augmentation), even though is still better than the non-speech vanilla round-trip paraphrasers we experimented with. Thus, the extra computation of using more than one cycle is not worth it.

What we observed before about the capabilities of Jaccard-Embedding Factor metric to predict the performance in the extrinsic evaluation still holds, in the sense that Jaccard-Embedding Factor of the Speech system with 2 cycles already under-performed with respect to the Speech system with only one cycle. So, the order in both evaluations is still the same. However, if we compare

paraphrasers with different languages and methods, the order does no longer hold (Russian is better than Speech with 2 cycles in Jaccard-Embedding-Factor, but worse in the extrinsic evaluation). We believe that this is indicative that the proposed metric can be useful, but we do not have enough data points to produce reliable statistical analysis, and it should be further studied.

Model	test accuracy
Baseline	0.735
Speech	**0.748**
Speech N=2	0.734

Table 4: Test accuracy in the extrinsic evaluation for different paraphrasing methods): We take the best vanilla round-trip machine translation paraphraser, the Speech one, and then compare it with a paraphraser also based on Speech but with 2 translation cycles. As per the extrinsic evaluation, the vanilla round-trip method outperforms the one with 2 cycles.

6 Conclusions and Future Work

To sum up, we have implemented from scratch a Python library for generating paraphrases with the well-known round-trip translation method. We have generalized the method to multiple cycles and different translation paths, and considered inter-modal transformations (speech) as a special case of machine translation. We have evaluated the generated paraphrases in a considerably complete evaluation framework, also defined by us. We have considered both qualitative and quantitative analysis, the latter both with intrinsic and extrinsic evaluation settings.

Perhaps the most interesting finding of the evaluation is that our proposed intrinsic metric, Jaccard-Embedding-Factor, evaluates the tested systems with the exact same order as the extrinsic evaluation metric, when using the same method (vanilla round-trip). The results are not yet conclusive, but this hints at the potential of our metric for avoiding the whole computation of an extrinsic metric, which is usually expensive. This relation between Jaccard-Embedding-Factor and the extrinsic evaluation results is less clear when introducing a different paraphrasing method.

Shuffling the training data after data augmentation with the generated paraphrases has improved the results, due to increased in-batch diversity. We conclude that paraphrases coming from closer languages are more similar to the original ones, yet not necessarily better as paraphrases per se (being too similar). We have found the paraphrases coming from the Speech model to be the one with the results the most consistent across intrinsic metrics. Furthermore, these paraphrases are the ones with the highest Jaccard-Embedding-Factor, and the ones the most useful in the sentence classification task, outperforming the non-augmented baseline. Using multiple cycles was not worth it, at least in our experiments, but it did not substantially degrade the generated paraphrases.

Personally, we have learned the importance of studying phenomenons, especially when they are counter-intuitive, and the difficulty and value of providing understanding of them. In contrast, works that base their contributions on developing new methods that obtain slightly higher accuracies are usually the most acclaimed. But in our case, for instance, when we found that adding paraphrases was actually counterproductive in some cases, we were not sure how to proceed. We inspected the code, and the results, and found some potential explanations. But these do not suffice, and we are still unsure about them. At least, in the case of the shuffling we found a logical cause of

the problem and a concrete solution.

As future work, we suggest different lines of work. First, it would be worth executing the more complex paraphrasing methods we implemented, even if the experiments will take a considerable amount of time. Apart from that, it would be interesting to conduct experiments on more languages (especially if from different families), and modalities (image, video). We believe to be necessary a deeper understanding of the reasons why the generated paraphrases from the German, Icelandic, and Russian models degrade the performance of the sentence classification system in the extrinsic evaluation. Evaluating the paraphrases as data augmentation in more tasks, instead of just only sentence classification with a single dataset, could further validate our results. Finally, we note that an improved version of our system could be used as a general data augmentation tool for NLP systems.

A Implementation

The project contains the following files and directories:

- `input/`: contains the dataset `tweets.csv`.
- `output/`: contains different foldersswith the results of each experiment. The name of the folder describes the translation method used, the name and the languages of the translators, the dataset used, the timestamp and the commit of the repository. Each of these folders contains:
  - `args.json`: contains information about the translation method used, the translator names and the path of the dataset.
  - `eval-`$<$ *timestamp* $><$ *commit* $>$`.json`: contains the results for the intrinsic and the extrinsic evaluation. For each sentence, the intrinsic evaluation displays the paraphrase and the result of the metrics: `bleu_score`, `normalized_original_sentence`, `normalized_paraphrase`, `embedding_cosine_similarity`, `edit_distance`, `normalized_edit_distance`, `jaccard` and `jaccard_embedding_factor`. At the end, it also shows the description (the total number of paraphrases, the mean, the standard deviation, the minimum value, the values of the percentiles and the maximum value) of each metric.
  - `eval-`$<$ *timestamp* $><$ *commit* $>$`.log`: contains the accuracy and the loss values for each epoch and the final accuracy of the sentence classification task.
  - `paraphrase.log`: log file of the paraphrase generation. In particular it contains information about the time that the translation takes for each cycle (i.e the time to translate from English to another language and vice versa).
  - `paraphrases.json`: list of lists where in each position appears the original sentence and its paraphrases. In addition, this folder contains a .json (`paraphrases_four_sources_2021-01-07-1735.json`) with the combination of all the paraphrases generated using different languages and converting the text to speech and vice versa.
- `paraphraser/`: contains the classes for generating and evaluating paraphrases.
- `extrinsic_evaluation/`: contains the classes that implements the sentence classification.
- `tacroton_pytorch/`: contains the implementation of the Google's Tacotron TTS system with PyTorch. This directory is not included directly, but downloaded with the `get-third-party.sh` script.
- `evaluate.py`: main program that evaluates the paraphrases of a given folder or directly a .json file.
- `paraphrase.py`: main program that generates paraphrases given a translator method, the names of the translators and the input dataset. As a result generates a folder with the results located at `output`.
- `run_extrinsic_baseline.py`: executes the extrinsic evaluation baseline (without data augmentation) with `tweet.csv`.
- `build_report.py`: Utility for generating Latex tables and randomly sampling paraphrases for the qualitative analysis.

- `get-third-party.sh`: download the implementation of Tacrotron TTS from the github repository `https://github.com/ttaoREtw/Tacotron-pytorch`.
- `setup.sh`: defines the setup of the project creating the vitual environment, installing the requirements and downloading the implementation of Tacroton TTS.
- `requirements.txt`: includes all the libraries used in the project.

In the following sections we are going to explain in depth the main modules of the project.

A.1 Input

We have used the dataset *Disaster Tweets*[10] from Kaggle for experimentation, stored in the file `tweets.csv`. This dataset contains a set of X tweets that about disasters that can be real or not. In particular, it has the following columns:

- `id`: unique identifier for each tweet.
- `text`: text of the tweet.
- `location`: location the tweet was sent from, may be empty.
- `keyword`: particular keyword from the tweet, may be empty.
- `target`: it has a value of 1 if the tweet is talking about a real disaster or 0 otherwise.

A.2 Output

This directory contains one directory for each of the executions. Inside each of them, there is the json file (`paraphrases.json`) resulting from executing the paraphraser along with the description of the parameters used (`args.json`) and the log file (`paraphrase.log`). The parameters that define an execution of the paraphraser are:

- `method`: describes the paraphraser method. As it was mentioned before there are three possible methods: round-trip, intermediate, and multi-cycle.
- `translator`: describes the names of the models used.
- `input`: the input file, useful for debug purposes. Initially the code was tested with smaller datasets.

A.3 Translator

We have a define a collection of pretrained machine translation models that generate translations from different sources and languages. In particular we can classify the translator in three categories according to the source of the translation:

- From text to text: we use multi-lingual translation Transformer-based models from Fairseq[11] [17] and MarianMT[12] [18] frameworks. Both models have English pairs with German, and Russian. In addition, Fairseq has English-Tamil pairs, and MarianMT has models for English-Icelandic.

[10] `https://www.kaggle.com/c/nlp-getting-started/data`
[11] `https://github.com/pytorch/fairseq/tree/master/examples/translation`
[12] `https://huggingface.co/Transformers/model_doc/marian.html`

- From speech to text: Silero Speech-To-Text[13] models are robust to a variety of dialects, codecs, domains, noises, lower sampling rates. The models consume a normalized audio in the form of samples (without any pre-processing except for normalization to -1...1). Although there are several languages available, we focus on using only English speech systems.
- From text to speech: Tacotron-pytorch[14] is an open-source implementation of Google's Tacotron generative Text-To-Speech model with pytorch. As the original paper mentions [19], the system consists of a text analysis frontend, an acoustic model and an audio synthesis module. Given (text, audio) pairs, the model is trained completely from scratch with random initialization. In addition, since Tacotron generates speech at the frame level, it is substantially faster than sample-level autoregressive methods.

A general class is implemented with the basic methods of a Translator. This class contains the method `build` that instantiates the translator according to its name and also a method `translate_sentences` that normalizes spaces and removes urls before calling the private translation method (_translate_sentences) of the pretrained model. In addition, the translator also saves a list of tuples with the direction of the translations. In the case of Text-To-Text models, a direction is defined by the names of the source and destination language while for speech to text and vice versa the direction is defined by default as ('en_speech', 'en') and ('en', 'en_speech') respectively.

For each translator, we define a `_translate_sentences` method, and the `directions` available:

- `FAIRHubTranslator`: given the name of the Transformer model, the class downloads the model from the hub `pytorch/fairseq`. The `_translate_sentences` just calls the private method `translate` of the Transformer and returns the translations.
- `MarianHFTranslator`: given the source and destination languages we build the name of the Transformer model since the following structure is always maintained `Helsinki-NLP/opus-mt-{src_lang}-{tgt_lang}`. Once we have the name of the model, we obtain the tokenizer and the model itself. The `_translate_sentences` converts the text to tokens, translate them and after converts them to text again.
- `SileroASR`: the English model is downloaded from the repository. In the `_translate_sentences` method, audio files are grouped into batches and input to the corresponding speech to the model. At the end the decoder returns the generated text.
- `TacotronPyTorch`: the pretrained model is downloaded from the checkpoint according to the configuration defined in `tacotron_pytorch/config/config.yaml`. The `_translate_sentences` method preprocesses the sentences by normalizing spaces, converting numbers and abbreviations to text and later converting the text to characters. At the end, the model outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech. The speech is saved in `.wav` in a temporal directory (`tmp`). By default, audio files are removed after having been transformed back to text.

A.4 Paraphraser

The Paraphraser instantiates a paraphraser according to the three methods mentioned before and generates paraphrases for each tweet. A paraphraser is characterized by a set of translators where

[13] `https://pytorch.org/hub/snakers4_silero-models_stt/`
[14] `https://github.com/ttaoREtw/Tacotron-pytorch`

the destination language from a given translator is equal to the source language of the next translator.

The number of generated paraphrases depends on the method and the number of translators defined in the configuration of the experiment. We have implemented a general class that includes the basic methods of a paraphraser from which the rest of the paraphrases inherit. This class contains the following methods:

- `paraphrase`: given a string that represents a sentence it generates its paraphrase.
- `_paraphrase_sentences`: given a set of sentences returns a dictionary with the list of paraphrases for each sentence. This method is implemented for each type of paraphraser.
- `paraphrase_sentences`: given a set of sentences, it removes white spaces at the beginning and at the end of each sentence. It calls the previous private method to return the dictionary with the paraphrases.
- `_translate_path`: given the sentences, the source and destination languages plus the intermediate languages, it translates the sentences in batches from the source language into the target language passing through the intermediate language/languages. This intermediate language can be only one language or a set of languages which is indicated with the word *all*. By default, for the experimentation English is defined as the source and the destination language English, since is the language we are targeting. As a result, the method outputs one paraphrase for each translator. For instance, taking into account that the source language is always English, if we have four translators: en-fr, en-es, en-ru and en-is and vice versa, the method will output four paraphrases per sentence.
- `build`: it instantiates the paraphraser according to the method defined.

A.5 Evaluator

We said that the quantitative analysis is divided into two processes: intrinsic evaluation and extrinsic evaluation. These two processes inherit from the class `Evaluator`, which implements a method that returns the instantiation of both evaluators. The classes `IntrinsicEvaluator` and `ExtrinsicEvaluator` implement the main method `evaluate_paraphrases`. In the case of the intrinsic evaluation, it returns the result of applying a set of metrics: BLEU score, edit distance, normalized edit distance, Jaccard distance and the cosine similarity of the sentence embeddings of a given paraphrase and the original sentence.

For the extrinsic evaluation, we use a repository that implements sentence classification with CNNs. The content of the repository is located in `extrinsic_evaluation` and it contains the following files:

- `configuration.py`: it contains the configuration of the preprocessing, model and training parameters.
- `model.py`: it contains the description of the model. The architecture of the model is composed of 4 convolutional layers which generate 32 filters each, then each one of these filters is passed through the max pooling function whose outputs are subsequently concatenated. Finally, the concatenation is passed through a fully connected layer.

- `preprocessing.py`: it is in charge of loading the data and generating training and testing datasets. Also if the `augment` flag is activated then performs data augmentation. As preprocessing steps, it removes special symbols, tokenizes the sentences, build the vocabulary (size of `num_words`) and adds padding to the sentences that does not fulfill the required length (`seq_len`).
- `run.py`: implements the training, testing phases and accuracy calculation.

References

[1] F. Josef and H. Ney, "Statistical machine translation," 06 2001.

[2] A. Prakash, S. A. Hasan, K. Lee, V. Datla, A. Qadir, J. Liu, and O. Farri, "Neural paraphrase generation with stacked residual LSTM networks," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, (Osaka, Japan), pp. 2923–2934, The COLING 2016 Organizing Committee, Dec. 2016.

[3] Z. Cao, C. Luo, W. Li, and S. Li, "Joint copying and restricted generation for paraphrase," *CoRR*, vol. abs/1611.09235, 2016.

[4] A. Gupta, A. Agarwal, P. Singh, and P. Rai, "A deep generative framework for paraphrase generation," *CoRR*, vol. abs/1709.05074, 2017.

[5] H. Somers, "Round-trip translation: What is it good for?," 01 2005.

[6] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Berlin, Germany), pp. 86–96, Association for Computational Linguistics, Aug. 2016.

[7] J. Effendi, S. Sakti, K. Sudoh, and S. Nakamura, "Enhancing neural machine translation with image-based paraphrase augmentation," 2019.

[8] J. Effendi, S. Sakti, K. Sudoh, and S. Nakamura, "Multi-paraphrase augmentation to leverage neural caption translation," 2018.

[9] Z. Li and L. Specia, "Improving neural machine translation robustness via data augmentation: Beyond back translation," *CoRR*, vol. abs/1910.03009, 2019.

[10] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, July 2002.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[12] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: A fully end-to-end text-to-speech synthesis model," *CoRR*, vol. abs/1703.10135, 2017.

[13] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *CoRR*, vol. abs/1908.10084, 2019.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[16] M. S. Maučec and G. Donaj, "Machine translation and the evaluation of its quality," in *Recent Trends in Computational Intelligence* (A. Sadollah and T. S. Sinha, eds.), ch. 8, Rijeka: IntechOpen, 2020.

[17] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[18] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. Fikri Aji, N. Bogoychev, A. F. T. Martins, and A. Birch, "Marian: Fast neural machine translation in C++," in *Proceedings of ACL 2018, System Demonstrations*, (Melbourne, Australia), pp. 116–121, Association for Computational Linguistics, July 2018.

[19] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. Saurous, "Tacotron: Towards end-to-end speech synthesis," pp. 4006–4010, 08 2017.