University *of Ljubljana*
Faculty *of Computer and Information Science*

Paraphrasing sentences

Luka Boljević, Matjaž Bizjak, Mitja Vendramin

Abstract

In this paper, we explore the task of automatic sentence paraphrasing - given an input sentence, the model returns its paraphrase. Many methodologies and models for this problem already exist, however, the most popular ones are usually for English, or mainstream languages such as German, French, and so on. As a paraphrasing model for Slovene doesn't seem to exist, we propose a simple methodology for fine-tuning a pretrained T5 model on publicly available Slovene corpora. We showcase the performance of our fine-tuned T5 models, as well as evaluate them manually, based on readability, relevance, fluency and number of changes in the generated paraphrases, and automatically, using BLEU and ROUGE metrics.

Keywords
Paraphrase, T5, translation

Advisors: Slavko Žitnik

Introduction

In this project, we will explore the task of paraphrasing sentences. Given an input sentence, our goal is to generate a different sentence that conveys the same meaning. Paraphrasing is an essential tool for effective communication, as it helps to improve clarity and simplify complex ideas. In terms of NLP, it can be used for tasks such as text summarization and generating diverse text for user-facing systems like chat bots. While paraphrasing sentences may seem like a straightforward task, it can be challenging to achieve accurate and natural-sounding results, especially for non-native speakers or those with limited language fluency.

A lot of research has been done and various models have been proposed, but a lot of them are pretrained on existing English datasets that contain sentence or question pairs, such as QQP (Quora Question Pairs), WikiAnswers, ParaNMT and Twitter URL paraphrasing corpora. Thus, such models are usable only for English. We propose a simple methodology that can be used for sentence paraphrasing in Slovene.

Our proposed methodology is the following. Using the publicly available Slovene corpora ccKres [1] and ccGigafida [2], we aim to obtain paraphrases using back-translation. In other words, the dataset i.e. individual sentences from the corpora will be translated to English, and then back to Slovene, using a publicly available Slovene to English back-translator. This will usually result in a slightly altered version of the original Slovene sentence, which we can take to be its paraphrase. Using these sentence pairs, we fine-tune T5 models already pretrained on Slovene (more on that later), so that it will be able to produce a paraphrased version of an input sentence.

Once we have the fine-tuned models, we will evaluate them using the popular ROUGE and BLEU metrics, as well as manually, in terms of readability, relevance, fluency, and number of changes in the generated paraphrases. The number of changes, or simply "change", refers to how much the original sentence was actually altered, i.e. how well does the generated paraphrase capture the notion of being a "paraphrase".

Related work

In the article Unsupervised Paraphrasing with Pretrained Language Models [3], Niu et al. propose a paraphrase generation model called "Dynamic Blocking". While training models for paraphrasing is typically supervised, their Dynamic Blocking model uses a transfer learning approach that enables pretrained language models to generate paraphrases in an unsupervised setting. This can alleviate the trouble of obtaining a large amount of labeled data. They used two different 'types' of automatic evaluation. The first one was BERT-iBLEU, which is a harmonic mean of BERTscore and one minus self-BLEU. The metric proved to be significantly better correlated to human evaluation than traditional metrics. The second type of automatic evaluation was using iBLEU, BLEU and ROUGE. Human evaluation was done by comparing this model with an already existing one - the human evaluators compared paraphrases from both models to see which one the evaluators liked more.

A novel DNPG model that generates paraphrases was proposed in article [4]. DNPG is a transformer-based model that can learn and generate paraphrases of a sentence at different levels of granularity, i.e. different levels of detail or specificity. They also developed an unsupervised domain adaptation method for paraphrase generation, based on DNPG. They tested their model on Quora Question Pairs (human-labeled) and WikiAnswers (automatically labeled) datasets. They used BLEU, ROUGE, iBLEU, and compared their DNPG model with 4 existing neural-based models. Human evaluation was done by six human assessors that each evaluate 120 groups of paraphrase candidates, given input sentences of course. Each group consists of the output paraphrases from 4 models, including the authors'. The evaluators then ranked the 4 candidates from 1 (best) to 4 (worst) based on their readability, accuracy, and surface dissimilarity to the input sentence.

The article A Deep Generative Framework for Paraphrase Generation [5] explains a method of paraphrase generation using deep generative models with sequence-to-sequence models. Gupta et al. use their own encoder and decoder architecture to generate sequences, which also works for paraphrases - one sequence generates multiple phrases. The human evaluation in this article consisted of multiple human evaluators scoring a subset of results on relevance and readability.

The article Paraphrase Generation with Deep Reinforcement Learning [6] has a different approach. Li et al. propose an approach consisting of a generator and evaluator. The generator is first trained to generate paraphrases, which are then evaluated by the evaluator. The generator is then fine-tuned by reinforcement learning where the reward is determined by the evaluator. For the evaluator learning they propose two different methods - supervised learning and inverse reinforcement learning. The human evaluation is done very similarly as in [5]. The authors use two metrics - relevance and fluency.

In the article [7] Kazemnejad et al. talk about retrieval-based method for paraphrase generation with Micro Edit Vectors for fine grade control over the editing process. The method is composed of a Retriever and Editor. The retriever selects a pair from the training corpus which is most similar to the input sequence. Editor is further split into the Edit provider, which computes Micro Edit Vector (MEV), and Edit performer, which rephrases the input sequence by utilizing the computed vector. The method was tested on QQP dataset and the Twitter URL paraphrasing corpus. Automatic comparison was made using BLEU, ROUGE and METEOR methods, while manual testing involved six annotators.

Article [8] proposed a Syntax Guided Controlled Paraphraser (SGCP), an end-to-end framework for syntactic paraphrase generation. It's a controlled text generation that adapts the sentence using various constraints. SGCP is comprised of 3 parts, namely sentence encoder, syntactic tree encoder and a syntactic paraphrase decoder. The authors used ParaNMT-small and QQP-Pos datasets. Automatic valuation was made using BLEU, METEOR, ROUGE-1, ROUGE-2, ROUGE-L, Syntactic transfer (TED-R and TED-E) and Model-based evaluation, while manual evaluation consisted of three judges.

Dataset

Slovene corpora

As mentioned in the introduction, we decided to use publicly available Slovene corpora ccKres [1] and (a subset of) ccGigafida [2] for fine tuning our models. Both corpora are sampled from larger, not publicly available corpora, namely Kres and Gigafida respectively (names are hyperlinked). ccKres contains about 10 million words, while ccGigafida contains around 100 million words. ccKres contains 9376 documents, while ccGigafida contains 31722 documents, which include various politics and history related texts, news taken from the internet, and so on. The corpora provide information about the source, year, type, title and author of each text, but for the purpose of this paper only the text itself was used.

Preprocessing the dataset

The first step after setting up the required tools and downloading the ccKres and ccGigafida datasets is preprocessing the said datasets. The dataset has text organised in such a way, that the text files include paragraphs. The first step was to divide these paragraphs into lone sentences. Before obtaining the sentences, we clean up the file contents a bit, so that, for example, all left braces are transformed into "(", all right braces into ")", different versions of a dash are transformed into "-", and so on.

After the cleanup, the file contents are split into sentences using NLTK's `sent_tokenize` sentence tokenizer. In each sentence, we also remove any non ASCII characters, except č, ć, š, ž and đ. We ignore sentences with less than 4 words, and more than 50 words (as, for example, around 97% of sentences in ccKres have less than or equal to 50 words). We then save each preprocessed sentence to a different line in the output file. This is partly so the data is split in the same way as in the original corpora, and partly due to the fact that translation we will shortly mention has a limit of 5000 characters per query. Saving the preprocessed sentences like this, in different files, makes it easier to make batches for the query.

Generating paraphrases for training

The preprocessed files are then used as an input phrases. We generate their paraphrases with the use of a translator and back-translation. For this purpose, we chose the Slovene_NMT translator[1]. To translate, we take the input phrases and send them through an API call to the translator. We first translate each sentence to English, save the output and then translate it back to Slovene to generate the paraphrases.

Manual evaluation of translations

A part of the translations were manually evaluated, again, based on readability, relevance and fluency. We picked 30 random samples and each of us evaluated them based on readability, relevance and fluency with a score from 1 to 5 where

[1] Slovene_NMT translator is publicly available on GitHub.

1 equals the worst and 5 equals the best. Our scores were then averaged - the average score for all three categories was above 4.85. We essentially scored all of the translations with a 5 - all translations were very readable, but there were some problems with relevance and fluency. For example: "*Ponjo boste morali priti sami.*" translated to "*Moraš ga vzeti v roke.*" which has a completely different meaning. The sentence "*Čebulo sesekljamo, popražimo na dveh žlicah olja, dodamo meso, solimo in popramo ter dušimo deset minut.*" translated to "*Čebulo sesekljamo, popražimo dve žlici olja, dodamo meso, sol in poper ter dušimo še deset minut.*". The mistake isn't as big as the previous example but there is a slight problem with the meaning.

Otherwise, all sentences returned by the translator were good. Any mistakes or problems were relatively minor and didn't impact the evaluation drastically.

Final training dataset

Now we have original (preprocessed) sentences, and their generated paraphrases using back-translation. For fine tuning, we make sentence pairs and tokenize them, which is a process of converting words to integers. As mentioned in the introduction, we will fine tune pretrained T5 models on Slovene - in particular, we use two models available on `cjvt`[2] repository on HuggingFace. The two models are "cjvt/t5-sl-small"[3] and "cjvt/t5-sl-large"[4].

Each model published on HuggingFace comes with its tokenizer, so we tokenize the sentence pairs using the ones from t5-sl-small i.e. t5-sl-large. Since tokenization is deterministic, we only need to tokenize the pairs once.

We made the raw preprocessed datasets, as well the tokenized sentence pairs available on OneDrive.[5].

Methods

Training the first model

The models from "cjvt" repository on their own aren't capable of generating paraphrases, so we need to train the models, or in other words fine tune them for this task. We used the *tranformers* library for this purpose. It offers the *Seq2SeqTrainingArguments* class with which we set our training arguments. We decided to first train t5-sl-small on (essentially) the entire ccKres corpus for 5 epochs, with an initial learning rate set to $2e^{-4}$ and weight decay of 0.01. It's also important to set the `fp16` argument to `False`, otherwise the training will not be successful. This ensures that the model doesn't use 16 bit precision for floats. The last argument we set was batch size, which we initially ambitiously set to 128. We lowered this number to 64, otherwise the training would result in an out of memory error by the GPU. Next we set up a data collator, which is also included in the library. The *DataCollatorForSeq2Seq* class will prepare the batches using a list of dataset elements as input. We can now create the trainer object using *Seq2SeqTrainer* class and start the training. We named this first pretrained model "t5-small-cckres". All of our pretrained models are available on OneDrive[6].

Further training

In the previous section, we described how we trained our first model. We also decided to train the small model once more, this time with a larger dataset. We also trained this model for 5 epochs, but instead of ccKres, we used 1.5M sentences from ccGigafida. The other hyperparameters were exactly the same: batch size 64, weight decay 0.01, and initial learning rate set to $2e^{-4}$. We named this pretrained model "t5-small-ccgiga".

We further fine tuned t5-sl-large twice. Both of those models were trained for 1 epoch, used batch size 16, and weight decay of 0.01. The first model we trained, which we named "t5-large-1", was trained on 1.5M sentences from ccGigafida, with initial learning rate set to $2e^{-5}$. The other one, which we named "t5-large-2 was trained on 1M sentences, with a higher initial learning rate - $2e^{-4}$. Again, all of our models are available on OneDrive.

The losses for t5-small-cckres and t5-small-ccgiga slowly but continuously decreased during the 5 epochs. Near the end of the 5th epoch, the losses for both models stabilized and didn't decrease further. The loss for t5-small-cckres ended up around 1.14, while the loss for t5-small-ccgiga ended up around 0.9.

The two large models (t5-large-1 and t5-large-2) experienced something similar. The loss for t5-large-1 dropped to around 0.81, 0.82 around 70% of the way through the epoch, and it didn't decrease further. This was the main reason why t5-large-2 was made. As explained, it used a bit less training data and a higher initial learning rate, to try and further minimize the loss, which we successfully accomplished. The loss for t5-large-2 at the end of the epoch was around 0.67, and it showed potential of decreasing further.

Results

As already mentioned, we automatically evaluated each of our models using the BLEU and ROUGE metrics. The results are visible in Table 1.

We also performed human evaluation on the trained models with the results visible in Table 2. In the same OneDrive folder where we published the models, you can find the file "Paraphrases for human evaluation.txt", where we list the 30 randomly sampled sentences we got those results from.

Discussion

Looking at the results of human evaluation in Table 2, we can see that readability, relevance and fluency got quite a high score. All of the generated paraphrases were essentially perfect, with minimal grammatical or semantic errors. However, there is a huge problem with the number of changes

[2]Center za jezikovne vire in tehnologije Univerze v Ljubljani

[3]cjvt/t5-sl-small pretrained model is available here.

[4]cjvt/t5-sl-large pretrained model is available here.

[5]Our final dataset can be found here.

[6]Link to trained models is here.

Model	Output method	Metric	Score
t5-small-cckres	Greedy	ROUGE1	0.73
		ROUGE2	0.57
		BLEU	0.52
	Beam	ROUGE1	0.74
		ROUGE2	0.57
		BLEU	0.52
	TopK	ROUGE1	0.72
		ROUGE2	0.54
		BLEU	0.50
t5-small-cciga	Greedy	ROUGE1	0.74
		ROUGE2	0.58
		BLEU	0.52
	Beam	ROUGE1	0.75
		ROUGE2	0.58
		BLEU	0.54
	TopK	ROUGE1	0.72
		ROUGE2	0.55
		BLEU	0.51
t5-large-1	Greedy	ROUGE1	0.73
		ROUGE2	0.56
		BLEU	0.50
	Beam	ROUGE1	0.74
		ROUGE2	0.57
		BLEU	0.53
	TopK	ROUGE1	0.70
		ROUGE2	0.52
		BLEU	0.47
t5-large-2	Greedy	ROUGE1	0.75
		ROUGE2	0.58
		BLEU	0.54
	Beam	ROUGE1	0.75
		ROUGE2	0.59
		BLEU	0.55
	TopK	ROUGE1	0.71
		ROUGE2	0.53
		BLEU	0.49

Table 1. Results for all four models with ROUGE and BLEU scores on three different methods of generation. As stated in the Methods section, t5-large-1 was trained on 1.5M sentences with initial learning rate set to $2e^{-5}$, while t5-large-2 was trained on 1M sentences, with initial learning rate set to $2e^{-4}$.

in the actual paraphrase - most of the generated sentences are too similar to the inputs. The problem can actually be traced back to the step of generating the paraphrases, i.e. to the back-translation part. While the back-translated sentences are essentially spot on, there isn't *that* much variation as we would expect in a paraphrase. This is actually the problem that persisted with all 4 of our models, so we skipped manually evaluating the others, as we deemed it unnecessary.

The problem with this could be addressed in a few ways. First, we could potentially solve this issue by translating the original dataset multiple times to generate much more diverse phrases. Another potential solution is to take an already existing English dataset intended for paraphrasing, such as QQP or WikiAnswers, and simply translate that dataset to Slovene.

	Readability	Relevance	Fluency	Change
t5-large-2	4.92	4.89	4.93	2.01

Table 2. Results of the human evaluation based on readability, relevance, fluency, and number of changes in the generated paraphrases. The number of changes, or simply "change", simply tells us how much the input sentence was actually altered. We each evaluated the quality of paraphrases generated from 30 randomly sampled sentences, and the results were averaged per category. Each category could be scored with a score from 1 to 5, where 1 equals the worst and 5 equals the best. For "change", 1 means "no to very little change", and 5 means "substantial amount of change, human level paraphrase". All 30 paraphrases were generated using the "beam" generation method.

Having this problem in mind, we can sort of understand why the ROUGE and BLEU metrics are so high in Table 1. Looking just at the numbers, the results are incredibly good, but unfortunately, as our paraphrases often aren't actually paraphrases, we can't give that much weight to the results. If anything, the results tell us that the models are definitely capable of being fine tuned for paraphrasing, but we need to be more mindful of our data.

Conclusion

In this project, we had a look at automatic sentence paraphrasing for Slovene. Our proposed pipeline was to obtain paraphrases from publicly available Slovene corpora using back-translation, and then fine tune T5 models pretrained on Slovene for the actual paraphrasing. We fine tuned 4 models, and performed automatic and manual evaluation.

We trained the models with varying degrees of success. The models often generated paraphrases that are too similar or even exactly the same as the input, which is a problem that can be traced all the way back to the back-translation step. Namely, the back-translated sentences weren't as diverse as we would want for training a good paraphrasing model. In this sense, the difference between different models i.e. small and large was negligible. The only difference was the time needed to train the model. Similarly, the size of the training dataset also didn't have much of impact in the end, something which we didn't expect to be the case.

Our models aren't entirely bad though. Although the generated paraphrases are unfortunately often similar to the inputs, the model is capable of switching the word order around, or replacing a few words. Every now and then, it is capable of generating a human level paraphrase - but, not as often as we want to. The actual output also depends on the generation method used - greedy, beam, top-k, top-p and so on.

Besides addressing the issues with the dataset itself, one of the ideas to improve the model(s) is using many paraphrases for a single input. Instead of using one back-translated sentence as the expected output, we could assign the input sentence with multiple back-translated sentences.

References

[1] Logar, Nataša and Erjavec, Tomaž and Krek, Simon and Grčar, Miha and Holozan, Peter. Written corpus ccKres 1.0, 2013. Slovenian language resource repository CLARIN.SI.

[2] Logar, Nataša and Erjavec, Tomaž and Krek, Simon and Grčar, Miha and Holozan, Peter. Written corpus ccGigafida 1.0, 2013. Slovenian language resource repository CLARIN.SI.

[3] Niu, Tong and Yavuz, Semih and Zhou, Yingbo and Keskar, Nitish Shirish and Wang, Huan and Xiong, Caiming. Unsupervised Paraphrasing with Pretrained Language Models, 2020.

[4] Li, Zichao and Jiang, Xin and Shang, Lifeng and Liu, Qun. Decomposable neural paraphrase generation. *arXiv preprint arXiv:1906.09741*, 2019.

[5] Gupta, Ankush and Agarwal, Arvind and Singh, Prawaan and Rai, Piyush. A Deep Generative Framework for Paraphrase Generation, 2017.

[6] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[7] Amirhossein Kazemnejad, Mohammadreza Salehi, and Mahdieh Soleymani Baghshah. Paraphrase generation by learning how to edit from samples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6010–6021, Online, July 2020. Association for Computational Linguistics.

[8] Kumar, Ashutosh and Ahuja, Kabir and Vadapalli, Raghuram and Talukdar, Partha. Syntax-Guided Controlled Generation of Paraphrases. *Transactions of the Association for Computational Linguistics*, 8:330–345, 06 2020.