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

Paraphrasing sentences

Luka Boljević, Matjaž Bizjak, Mitja Vendramin

Abstract

We will write the abstract for the final report.

Keywords

Paraphrase, ...

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 corpus ccKres [1], we aim to obtain paraphrases using back-translation. In other words, the dataset i.e. individual sentences will be translated to English, and then back to Slovene. 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 will fine-tune a pretrained T5 model so that it will be able to produce a paraphrased version of an input sentence.

Related work

In the article Unsupervised Paraphrasing with Pretrained Language Models [2], 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, which can be cumbersome. For automatic evaluation they used two different evaluations. 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, and compare paraphrases from both to see which one the evaluators liked more.

A novel DNPG model that generates paraphrases was proposed in article [3]. 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 [4] explains a method of paraphrase generation using deep generative models with sequence-to-sequence models, where given an input sequence, it generates a paraphrase. Gupta et al. use their own encoder and decoder architecture

to generate sequences, which also works on paraphrases not previously learned - one sequence generates multiple phrases. The human evaluation in this article consisted of multiple human evaluators scoring some subset of results on relevance and readability.

The article Paraphrase Generation with Deep Reinforcement Learning [5] 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 [4]. The authors propose two metrics - relevance and fluency.

In the article [6] Kazemnejad et al. talk about retrieval-based method for paraphrase generation with Micro Edit Vectors for fine grade control over editing process. The method is composed of Retriever and Editor. The retriever selects a pair from the training corpus which is most similar to the input sequence. Editor is further split to 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 [7] 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.

Methods

The methods used are described in this section.

Preprocessing the dataset

The first step after setting up the required tools and downloading the ccKres dataset is preprocessing the said dataset. 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 ")", 7 different versions of a dash are transformed into "-", and so on. After the cleanup, file contents are split into sentences using Python's `nltk.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 (around 97% of sentences in ccKres have less than or equal to 50 words). We save each sentence to a different line in the output file.

Generating paraphrases for training

The preprocessed files are then used as an initial set of phrases. We use these to generate additional phrases with the use of a translator and back-translation. We chose the Slovene_NMT translator[1], and translation is done in such a way that we take the output files from the preprocessing step 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.

A part of the translations were then manually evaluated. We picked a couple of samples and evaluated them based on readability, relevance and fluency with a score from 1 to 5 where 1 equals the worst and 5 equals the best. Almost all sentences were translated perfectly. The readability was good for all of them, but the relevance and fluency had some problems. 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 some problem with the meaning.

Otherwise all sentences returned by the translator were good. Even the ones with mistakes or problems had minor problems and didn't impact the results.

Training a model

Now we have a folder with original sentences, and a folder with generated paraphrases. Next we have to make pairs of them and tokenize them. We first load the sentences in two arrays. It's important that the second array matches the first in indices - so each element X on index Y in array one has the corresponding paraphrase in array two on index Y. Then, we have to tokenize the sentences. This is a process of converting words to integers. Each model comes with it's tokenizer. We will be using `cjvt/t5-sl-small` model from HuggingFace[2], so we use the included tokenizer. After the pairs are formed and tokenized, we can begin training our model. We trained the model (for the time being) for one epoch. After just one epoch the model is capable of generating sentences, but it still needs a little bit more tuning to correctly generate the paraphrases, especially if the sentence is longer.

Future work and ideas

We of course plan to extend and improve the current work and report for the final submission. Here are some of our plans listed in no particular order: (1) Further refine the model, using more data and training for longer or/and with different

[1] The translator is publicly available on GitHub.

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

arguments, (2) Make the final model and dataset available, (3) Further clean up the repository, add more in README (4) Compare the fine tuned `t5-sl-small` model to a fine tuned multilingual T5 model, for example `google/mt5-small`, (5) Also compare the fine tuned `t5-sl-small` to a "model" that generates paraphrases by replacing some words with synonyms, (6) Evaluate the final fine tuned `t5-sl-small` model, both manually and automatically (i.e. using manual and automatic metrics), (7) And so on.

Results

Discussion

Conclusion

Acknowledgments

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] 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.

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

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

[5] 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.

[6] 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.

[7] 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.