Senior Project Report

Department of Computer Science

Calvin University

Title: Alternative Wordings

Author: Noah Madrid

Date: 5-10-21

Mentor: Kenneth Arnold

Background and Problem

While writing, variety in vocabulary and sentence structure is vital in staving off monotony and retaining reader interest. While there are always multiple sentence structures which can be used to communicate the same idea, switching between various sentence structures can often be difficult even for experienced writers. A tool which can generate these various sentence structures could therefore be useful for all writers as they experiment with the style of their text. Creating a tool with the ability to switch the overall structure of a sentence while still retaining meaning, requires a system which can understand the abstract meaning of a sentence, something which can now be done with machine learning translation models. This task, often called paraphrasing, is defined as using an alternative surface form to express the same semantic content (Mallinson et al., 2017). Therefore, a successful paraphrase system will not change the meaning of a sentence, only the way that meaning is expressed. A common method of generating paraphrases within machine learning today is through round-trip translation. Round-trip translation entails translating text to a foreign language (often referred to as the pivot language) and translating the result back to the original language. Through translating text to a foreign language, the semantic content of a sentence is abstracted out. This abstracted semantic meaning can then be used to form new sentences by translating back to an original language (Mallinson et al., 2017).

Over the summer of 2020, Calvin University student April Volzer, working with Professor Arnold, created a tool to generate paraphrases of a given sentence based on round-trip translation. Using Vue.js, a JavaScript front end toolkit, for the user interface and Flask, a Python based web-development framework, for the backend, this tool employed a pre-trained translation model from Hugging Face's Transformers, a Python natural language processing library. Sentence alternatives were generated through round-trip translation. Paraphrase sentence structure diversity was obtained by generating a multitude of possible ways a paraphrase could start, and then forcing these possible starting phrases during translation back to English (Figure 1).

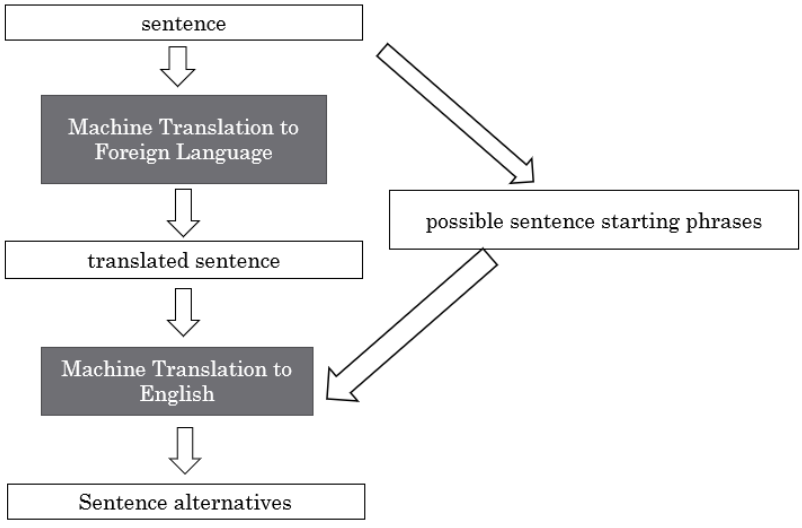

Figure 1, Round-trip translation with prefix forcing

This alternative wording generator allowed users to switch single words in a sentence with other likely words for the same context and generates sentence structure alternatives with some limitations (Figure 2).

Figure 2, An example result of the original system built by April Volzer.

One of the limitations of April's system was that the user had limited control over the paraphrases shown to them. Sentences with specific starting phrases could be expanded to see other sentence alternatives, but, besides an initial phrase, it was not possible to choose the order phrases were generated. Switching two phrases later in a sentence could not be accomplished and the order of phrases after the initial phrase was not guaranteed. This limitation was not a UI limitation but rather a system limitation. Generating alternatives to a sentence could only be done through specifying a single constraint determining the start of the sentence. In order to specify multiple constraints and have control over later phrases, our project wanted to add a drag and drop feature in which all phrases in the sentence could be arbitrarily reordered to generate paraphrases. To add such a feature, a new system of generating paraphrases would have to be researched and developed. The question which directed our research was, what kind of system would allow phrase reordering to direct paraphrase generation?

While the domain of paraphrase generation within machine learning is large and rich, there is limited work on combining the fields of paraphrase generation and controlled text generation. Past work on the controllable paraphrase generation has included, for example, using a "syntactic exemplar" sentence which provides syntactic guidance to the paraphrase generation while still maintaining the original meaning. This type of generation poses the problem of how to effectively obtain and choose exemplar sentences for generation and would be difficult to use for a prospective drag and drop system. Another method of controllable paraphrase generation involves giving a model both a sentence to paraphrase and reordering indices to guide generation to a specific order. Finally, a third method we propose is combining lexically constrained decoding with round-trip paraphrase generation. Lexically constrained decoding is a relatively new search method using a grid beam search to ensure the decoder of a model will find output

that includes specific lexical constraints. Through using the phrases within a sentence as the ordered constraints in lexically constrained decoding, a rudimentary drag and drop system can be created.

Approach

As our project is based on April Volzer's original system, our approach continues using a Vue.js front end along with a Flask back end. As Hugging Face's Transformers does not yet support lexically constrained decoding, we were forced to find a new machine learning library capable of lexically constrained decoding. This library was also required to have pre-trained translation models, as we did not have the time or resources to train our own model. The only library which currently fulfills both of those conditions is the Fairseq machine learning sequence modeling toolkit built by Facebook AI.

As the model for round-trip translation we chose Fairseq's pretrained mBART created by Facebook AI researchers. MBART is a sequence-to-sequence denoising auto-encoder pretrained on large-scale monolingual corpora in many languages. MBART also uses a sentence-piece Byte-Pair Encoding model for tokenization (Liu et al., 2020). Out of all the pre-trained models within Fairseq, mBART was chosen due to both its high translation quality and its multilingual capability. As mBART supports translation to and from multiple languages, a round-trip translation only requires a single model to be initialized rather than two (one to translate to a foreign language and one to return to the original language). For our round-trip translation we chose Dutch as our pivot language. Much of the quality of paraphrases through any round-trip paraphrase method relies on the ability of the model to not lose information as it translates from English to a foreign language and back again. If the model's translation to a foreign language is faulty, the paraphrases generated in the translation back to English will also be faulty. To limit the loss of meaning from this process, using languages similar to English as a pivot language is desirable. While we experimented with other languages, we ended up choosing Dutch as our pivot language. Its similarity to English allows translations to maintain a large degree of meaning as can be seen in pivot language comparisons in Figure 3.

Original Sentence	Pivot Language	Constraints	Expected Result	Actual Result
Researchers found that heart attacks can be caused by stress.	Dutch	[Heart attacks, researchers, stress]	Heart attacks, researchers have found, can be caused by stress.	Heart attacks, researchers have found, can be caused by stress.
Researchers found that heart attacks can be caused by stress.	German	[Heart attacks, researchers, stress]	Heart attacks, researchers have found, can be caused by stress.	Heart attacks researchers found can be caused by stress.
Researchers found that heart attacks can be caused by stress.	French	[Heart attacks, researchers, stress]	Heart attacks, researchers have found, can be caused	Heart attacks researchers have found can be caused by stress.
Researchers found that heart attacks can be caused by stress.	Chinese	[Heart attacks, researchers, stress]	Heart attacks, researchers have found, can be caused	Heart attacks are researchers found to be caused by stress.
Researchers found that heart attacks can be caused by stress.	Dutch	[Heart attacks, stress, researchers]	Heart attacks can be caused by stress, researchers found.	Heart attacks, stress is what researchers have found to be the cause of heart attack
Researchers found that heart attacks can be caused by stress.	Dutch	[Heart attacks can, stress, researchers]	Heart attacks can be caused by stress, researchers found.	Heart attacks can be stress-induced, as researchers have found.
Researchers found that heart attacks can be caused by stress.	Dutch	[Heart attacks can be caused by stress, researchers have found.]	Heart attacks can be caused by stress, researchers have found.	Heart attacks can be caused by stress, researchers have found.
Researchers found that heart attacks can be caused by stress.	Dutch	[heart attacks, researchers, stress]	Heart attacks can be caused by stress, researchers have found.	Well, heart attacks, researchers have found, can be caused by stress.
The window was broken by me.	Dutch	[me, the window]	I broke the window.	The window was broken by me, window was broken.
The window was broken by me.	Dutch	[I, the window]	I broke the window.	I broke the window.

Figure 3, A table of sample model outputs demonstrating the impact of constraints and pivot languages on model generation.

An integral element of our current system is our method of choosing constraints. At the present, our system identifies the noun chunks within a sentence using the spaCy natural language processing library and treats them as the constraints. These noun chunks are highlighted for the user and they can be dragged and dropped around the sentence. If a noun chunk is moved to a new relative position to another the model is called with the reordered noun

chunks given as constraints (Figure 4).

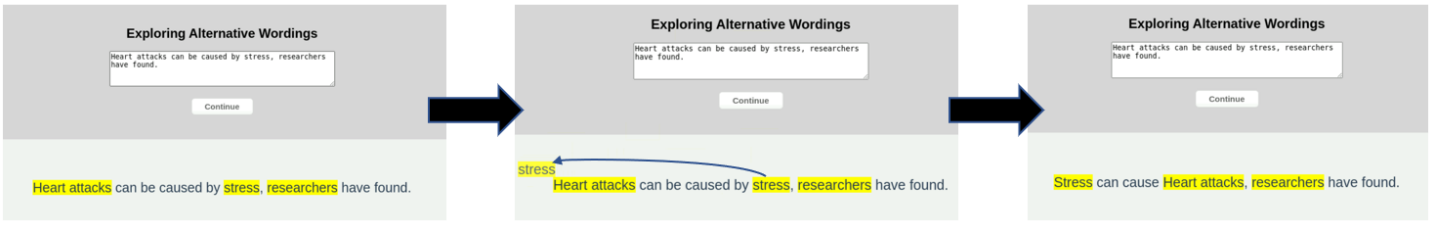

Figure 4, example of the current drag and drop UI experience. The original constraint list in this case would be [Heart attacks, stress, researchers] while upon the drag and drop action it would change to [Stress, heart attacks, researchers].

Results, Analysis, and Discussion

Choosing constraints to use from a sentence is perhaps the most difficult task in developing a system using lexically constrained decoding. The current method of using noun chunks as constraints is very simplistic and does not include every phrase that could feasibly be moved in a sentence. For example, in the example "Yellowstone Nation Park was established by the US government in 1972 as the world's first legislated effort at nature conservation" the phrase "in 1972" should clearly be draggable but is left out when using only noun chunks. While we had a system in place for generating prefixes for a sentence, this would not work for generating constraints without major adjustments. When it comes to constraints, giving the model more lexical information within the constraints generally leads to better output. In the extreme case, one can give the model the desired paraphrase sentence itself as a constraint giving the model a high degree of lexical information and nearly guaranteeing the model will generate the correct paraphrase. While more detailed the constraints given to the model the more the model is guided in the search process, but this same guidance can cause the model to fail to paraphrase if the constraint is invalid. One way a constraint could be invalid is if it needs to be modified to fit in a sentence reordering. For example, in the sentence "The window was broken by me" a possible reordering for this sentence would switch from passive to active voice resulting in "I broke the window". In this case the only phrase which is unaltered by the reordering which could be given as a constraint would be "the window" as both the verb and pronoun are altered. To correctly generate this sentence, a preprocessing step would be required to convert either the pronoun (to "I") or the verb (to "broke") before using it as a constraint. Other factors to keep in mind while constructing constraints is capitalization and punctuation. As constraints preserve both the capitalization and punctuation within them due to mBART's tokenizer, through capitalizing the first word of a phrase one can influence the model to generate it first and through adding a period to the end of a constraint one can guide the model to generate that word last. The opposite is also true. Not capitalizing the first word in a constraint phrase ensures it is unlikely to be generated at the start of a sentence (examples of these constraint issues can be viewed in Figure 3).

Overall, this project succeeded in making an improvement to the original system. Our system allows for an arbitrary number of constraints compared to a single constraint at the start of a sentence. The drag and drop UI implementation succeeds in giving a user agency over

reordering a sentence in a much more direct way than the previous system and provides added flexibility.

Problems Encountered

While researching and developing a drag and drop system the project ran into a variety of obstacles. The first issue faced within the project was getting up to speed on an existing codebase. This required learning Vue.js, PyTorch (a machine learning library), Flask, and Hugging Face's Transformers as well as deciphering how these components all fit together within the codebase. This process took longer than expected and led to less time extending the project than expected. Secondly, the project lost a team member halfway through, requiring a reevaluation of the project goals and scope.

Lexically constrained decoding was not the first approach we researched. A large amount of the project work time entailed experimenting with various systems to reach the goal of a drag and drop. Initially, we assumed we would have to train a model similar to Tanya Goyal and Greg Durrett's implementation in "Neural Syntactic Preordering for Controlled Paraphrase Generation", but as we experimented with adding reordering information to a model we realized that this process would probably take more resources and time than we had. After discovering lexically constrained decoding, we chose it mainly due to its reliance on already trained models.

After deciding on lexically constrained decoding to implement our drag and drop system, we had to port a large amount of our backend code to the Python sequence modeling toolkit Fairseq from Hugging Face's Transformers. Fairseq is mainly intended to be used with the command line, but due to our backend being written in Python and the level of control we required over the model, we had to directly interact with the Python code base. There is very little instruction on the repository of how to use Fairseq directly through Python especially in comparison to Hugging Face's Transformers which led to difficulties in correctly using the model. The lexically constrained decoding within Fairseq is also a very new introduction to the repository with little documentation leading to further difficulties such as strange system behavior at times.

Possible Future Work

A possible extension to our work would be to switch back to Hugging Face's Transformers repository to leverage their wider variety of models. This would require either porting Fairseq's lexically constrained decoding to Transformers or waiting until it is ported by someone else. A further extension to our work which could increase the performance of the model is leveraging zero-shot paraphrase generation using a multilingual language model such as is introduced in the paper "Zero-Shot Paraphrase Generation with Multilingual Language Models" (Guo et al., 2019). Zero-shot paraphrasing works by essentially translating from English to English, skipping the intermediary pivot language, which means that there would be less loss of sentence meaning with this method than by conducting a round-trip translation. As often writers may want to see all other alternatives to a sentence rather than specifying a specific one through drag and drop, finding a way to integrate the current backend system with the original

system would allow more functionality for any user. Finally, much more work needs to be done on deciding which constraints to use and how to alter phrases for specific orderings.

References

Goyal, T., & Durrett, G. (2020). Neural syntactic preordering for controlled paraphrase generation. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 238–252. https://doi.org/10.18653/v1/2020.acl-main.22

Guo, Y., Liao, Y., Jiang, X., Zhang, Q., Zhang, Y., & Liu, Q. (2019). Zero-shot paraphrase generation with multilingual language models. *ArXiv:1911.03597 [Cs]*. http://arxiv.org/abs/1911.03597

Hokamp, C., & Liu, Q. (2017). Lexically constrained decoding for sequence generation using grid beam search. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1535–1546. https://doi.org/10.18653/v1/P17-1141

Hu, J. E., Rudinger, R., Post, M., & Van Durme, B. (2019). Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. *ArXiv:1901.03644 [Cs]*. http://arxiv.org/abs/1901.03644

Kumar, A., Ahuja, K., Vadapalli, R., & Talukdar, P. (2020). Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, *8*, 330–345. https://doi.org/10.1162/tacl_a_00318

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., & Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *ArXiv:2001.08210 [Cs]*. http://arxiv.org/abs/2001.08210

Mallinson, J., Sennrich, R., & Lapata, M. (2017). Paraphrasing revisited with neural machine translation. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 881–893. https://www.aclweb.org/anthology/E17-1083

(Vectorized) Lexically constrained decoding with dynamic beam allocation. (n.d.). GitHub. Retrieved May 12, 2021, from https://github.com/pytorch/fairseq