Rhetorical Figures as Machine Paraphrasing Constraints

Anonymous ACL submission

Abstract

We use rhetorical figures to organize the natural language processing problem domain of organizing under constraints. We believe that this organizer has significant implications for computer assisted style editing. In addition to proposing this novel framework, we worked on an algorithm for paraphrasing sentences so that they contain alliterating or rhyming words at a user-specified rate. We developed a novel architecture for paraphrasing under frequency-based constraints. We leveraged a high performing language model to produce a sample of seed paraphrases that were fed into a rules-based word-replacement algorithm. This allowed our algorithm to explore a larger part of the constrained paraphrasing landscape compared with previous work (Allen; Dras, 1997). Our algorithm consistently produced output that satisfied the given constraints. However, it did poorly at preserving the semantics of input prose. Near term work would leverage better sentence embeddings to improve performance. Additional future work would involve solving other structural and semantic constraint paraphrasing problems, and developing paraphrasing algorithms for joint rhetorical figure constraints.

1 Introduction

This paper investigates a subset of problems within the domain of paraphrase generation under constraints. Past works in this area include (1) paraphrasing prose into rhyming poetry (Allen), (2) constraining paraphrases by sentence length or complexity (Dras, 1997) and (3) metaphor generation.[1] However, there has been no apparent recognition that these problems fall into a general class of related tasks or that these tasks can be mostly indexed by linguistic schemes known as rhetorical figures.

[1] See Stowe et al. (2020) for a review.

Rhetorical figures are patterns of speech that have been recorded and taxonomized by writers and orators. For definitions of specific rhetorical figures, see Lanham (1991).

Harris and DiMarco (2009) developed a taxonomy of rhetorical figures (see Figure 1) that we have modified to reflect the kinds of problems explored in this paper (see Figure 2). In Figure 2, the family of figures labeled "Repetition" is more syntactic and less semantic than most other kinds of rhetorical figures. As a result, this family is an ideal starting place for exploring ways of paraphrasing arbitrary sentences according to each figures' constraints (because it is less likely that semantic considerations will preclude the existence of acceptable paraphrases).

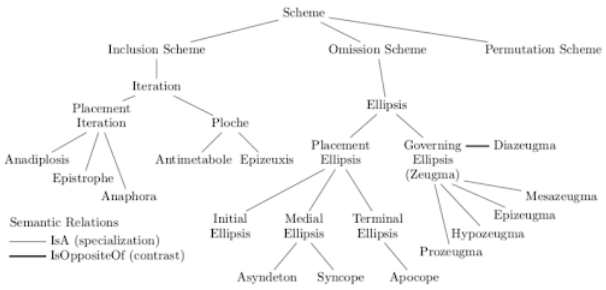

Figure 1. A seed ontology of rhetorical schemes

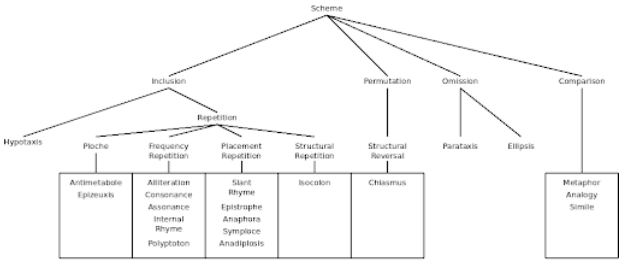

Figure 2. An ontology of rhetorical figures, organized to emphasize families of schemes that share similar constraints.

2 Previous Research

2.1 A Rules Based Approach

Allen addressed a paraphrasing problem that shares

strong similarities with repetition based rhetorical figures. The goal of Allen's project was to paraphrase prose as rhyming poetry. Therefore, the problem has two constraints. First, the prose must be converted into a sequence of lines where the end of each line rhymes with the end of the preceding or following line. Next, the rhyming lines must share the same or nearly the same number of syllables. Finally, the paraphrase must preserve the meaning of the original prose.

Allen uses a highly rules-based approach to this problem. First, for each word in the prose, he uses WordNet and VerbNet to build a list of synonyms and hypernyms in descending order of similarity. Each word in each word-substitute list, is checked against the words in the original prose for rhyme matches using the Carnegie Mellon Pronunciation Dictionary. Next, the candidate rhymes are ranked according to the product of a rhyme quality score and a meaning preservation score. Several of the top rhyming pairs are saved. For each candidate rhyming pair, the associated lines are cached.

Next, a paraphrasing algorithm attempts to make each pair of lines conform to the constraint that they must be the same length (or very close) in syllables. First, Allen uses the Stanford Parser to tag each word and clause with its parts of speech. Next, each pair of lines can be rearranged by moving any combination of prepositional phrases to the beginning of their parent clauses. All of these rearrangements are found and passed to the next step. Rearrangements are scored according to how few swaps are made.

In the next step, the algorithm ranks phrases according to their disposability. Some parts of speech are more likely to be disposable than others. For example, adverbs and adjectives can usually be deleted without substantially changing the meaning of a phrase. Paraphrases scored according which parts of speech were deleted and how many deletions were made.

Next, each rhyme split is ranked according to a full score that is computed by multiplying (1) rhyme quality, (2) synonym quality, (3) rearrangement quality and (4) truncated phrase quality (all scores are inverses of discrete quantities). The best pair is chosen and printed. Any rearrangements from the best pair are applied to the original text. The algorithm is then re-applied to the text, beginning with the next word in the original text. The algorithm greedily picks rhyme pairs with no consideration for the quality of future rhymes.

One major limitation of the approach taken by Allen is that the algorithm could only explore a very small slice of the space of paraphrases. This criticism applies generally to rules-based paraphrasing approaches. In the methods section, we will revisit this issue and explore one method for exploring a wider range of possibilities when paraphrasing sentences.

2.2 Statistical Approaches

Dras (1997) takes a more principled approach to the problem paraphrasing under constraints. He frames the task as a continuous optimization problem. Dras' model starts off with Decision Variable, a binary value, p_ij which determines whether a paraphrase should continue to be made to the text. To minimize the amount of reluctant changes, an Objective Function is used to measure the total summation of changes made to the original text by calculating the change, c_{ij}, per paraphrase

$$z = \sum c_{ij} \cdot p_{ij}$$

The Objective Function could be transferred to measure length constraints by limiting the change to a constant value. As shown in the following equation, the amount of changes made by the constraint is determined by k_1.

$$\sum w_{ij} \cdot p_{ij} \leq k_1$$

where w_{ij} = change to length of sentence i caused by paraphrase

ij k_1 = required change to the length of text in words; and $k_1 \leq 0$ (Dras, 1997)

Building on top of the base of the Objective Function, Dras defined a readability constraint

$$\frac{W + \sum w_{ij} \cdot p_{ij}}{S + \sum s_{ij} \cdot p_{ij}} \leq k_2$$

that is,

$$\sum (w_{ij} - k_2 \cdot s_{ij})\, p_{ij} \leq k_2 S - W$$

s_{ij} = change to number of sentences in the text by paraphrase

ij W = total number of words in original text

S = total number of sentences in original text k_2 = required average sentence length $k_2 \geq 0$ (Dras, 1997)

which enforces an average sentence length by tracking not only the amount of changes made per paraphrase to a specific sentence length, but also tracking the number of total sentences in the text remaining after each paraphrase. Afterwards, comparing the ratio of total words in the text to the total number of sentences in the text with a determined average sentence length constraint. n

Now since it is possible to track different ratios of changes made to a text this can be expanded to interpret a variety of constraints. Dras defines a lexical density constraint which manipulates the ratio of closed class words to total words.

$$\frac{F + \sum f_{ij} \cdot p_{ij}}{W + \sum w_{ij} \cdot p_{ij}} \geq k_3$$

that is,

$$\sum \left(f_{ij} - k_3 \cdot w_{ij}\right) p_{ij} \geq k_3 W - F$$

where

f_{ij} = change to number of function words caused by paraphrase

ij F = total number of function words in original text k_3 = required proportion of function words to total words; $0 \leq k_3 \leq 1$ (Dras, 1997)

Another possible use for this ratio constraint is to add alliteration to a text by controlling different types of ratios. One possible way to incorporate alliteration could be to limit the number of "odd words" (words that do not begin with the same letter as the rest of the sentence) to total words in a sentence.

Lastly, to remedy the issue of conflicting paraphrases, Dras limits the amount of paraphrases that could be made to the sentence.

$$\sum_j p_{ij} \leq 1$$

In general, rhetorical figures of repetition do not require strict length constraints like those explored in Dras (1997). However, it is often desirable for paraphrases to simplify and preserve the length of the original text. Framing length as an optimization problem—rather than a strict constraint—could improve the performance of our paraphrasing algorithm by introducing flexibility in this area. Moreover, there are two other highly syntactic rhetorical figures in the ontology we are considering for this paper: hypotaxis and parataxis. Both of these rhetorical figures may be largely characterized by the length and complexity of a paraphrase relative to the original text (parataxis would reduce length and complexity while hypotaxis would increase them).

More generally, statistical approaches to paraphrase generation have the potential to substantially improve on rules-based paraphrasing methods by exploring a larger portion of the problem space. In the past, generative language models have been used for unconstrained paraphrasing by leveraging foriegn language translation algorithms to mutate input prose. For example, Quirk et al. (2004) used a monolingual implementation of three linguistic translation algorithms to generate paraphrases. They discovered that phrase replacement was often superior to word replacement. As a result, using a machine learning algorithm for finding equivalent phrases could improve on a rules-based approach without destroying the interoperability of the algorithm. Additionally, we believe that the cosine distance between word embeddings would outperform WordNet synonym and hypernym rankings as a word-meaning preservation score for replaced key words. Moreover, generative language models use continuous quantities for scoring word similarities, phrase similarities and n-gram expectations. As a consequence, these scores can be integrated into an optimization problem similar to the one defined in Dras (1997). Therefore, our team decided to augment Allen's search-and-replace approach with a generative language model and a more principled approach to constraint optimization.

2.3 Machine Learning Approaches

There are many high performing unconstrained paraphrasing algorithms in the domain of machine learning. We aimed to find an algorithm that was capable of producing diverse paraphrases while retaining the semantics of the original input text. After considering several options within the realm of machine learning, we found that transformer models which would be most applicable for our needs in paraphrase generation were BERT, BART, and Pegasus.

BERT(Bidirectional Encoder Representations from Transformers) was a revolution in terms of creating context based language models. According to (Devlin et al., 2018), the key to BERT's outstanding performance is its bidirectional encoder.

As opposed to other language models, such as OpenAI's GPT3 (which can only decode from left to right), BERT can gather context in both left and right directions. It goes about this by randomly "masking" certain tokens from a given input, forcing the model to guess at the original token from collecting data on the other tokens.

The main issue with BERT was the fact that a bidirectional model generally suffered at generation tasks due to limitations at total sentence prediction. Our first solution was to use Facebook's BART(Bidirectional AutoRegressive Transformers), which built on the bidirectional masking of BERT with a few more modes of context derivation and a similar autoregressive decoder as used by GPT3. These unidirectional models had much better success with sentence generation; combining this with the bidirectional encoder meant greater contextual awareness within better paraphrases. (Lewis et al., 2019) explains the mechanism and performance of BART in relation to BERT, and several other BERT-like models.

Unfortunately, BART's main issue with respect to paraphrasing was its lack of novel or unique enough paraphrases. This led us on to looking at Pegasus(, Pre-training with Extracted Gap-sentences for Abstractive SUmmarization), Google's cutting edge Encoder-Decoder transformer model. (Jingqing Zhang, 2020) combined the masked language encoder with a novel concept they called Gapped Sentence Generation. Faithful to its name, it creates large gaps midsentence to predict. This essentially layers a normal, BERT-like encoder with a GSG encoder to grab even more context than before. After tweaking both the sampling and beam search parameters, and finding a pretrained and fine tuned model, we were able to generate unique, faithful, unconstrained paraphrases.

3 Method

We developed an algorithm for paraphrasing input phrases into prose that (1) alliterate, (2) have assonance or (3) have internal rhyme. We defined a function that takes an English sentence with n words, and a parameter p, and—if it can find a solution—returns a sentence where approximately np words alliterate, are assonant or rhyme. The function also takes several additional parameters that will be discussed in turn and then recapped together.

We aimed to overcome the respective limitations of rules-based and machine learning approaches by combining the two methods. Our algorithm starts by generating a sample of unconstrained paraphrases from the original phrase using a free paraphraser (the sample size is currently given by the user). We used the pre-trained and fine tuned Google Pegasus (Jingqing Zhang, 2020) model to generate our sample paraphrases. However, any adequate paraphrase generator could be used in its place.[2] The main consideration is the variety and quality of the paraphrases generated.

Next, the algorithm takes each sample sentence (including the original) and uses a rules-based search algorithm to find word replacements that force the phrase to conform with the given frequency constraint. We call this portion of the algorithm the Procrustean paraphraser because it forces sentences to conform to the constraint, often with unnatural sounding results (hence, why sampling plays an important role in producing sound paraphrases).

To begin, the Procrustean paraphraser takes an input string and breaks it down into a token string list using a mapping module. The internals of this breakdown is performed by a separate language module that uses an internally defined tokenizer that returns favorable separations for computations (the English language module within the project uses NLTK RegexTokenizer). Afterwards, this list of tokens is processed through a rule application module. Frequency based rules (i.e. alliteration, assonance, rhyme, etc) run token strings through a series of pre-processing steps. The first pre-processing step is compiling a list of synonyms for each token through the language module using a corpus (the English module uses NLTK Wordnet). Next, the phonemes of the original token and each of its synonyms are calculated, through a phoneme corpus defined in the language model, and tabulated within a dictionary indexed by the most relevant phonemes (i.e. for alliteration its the first phoneme, for rhyme its the last phoneme, etc). When a token is tabulated within the dictionary as a value of a relevant phoneme, its index is also recorded. The phoneme-collection dictionary's type should be of 'Dict string, Dict int, string'. This ends the pre-process step. During rule processing, any phonemes that do not comply

[2] We are already considering the use of an algorithm developed by Goyal and Durrett (2020). It has shown promise in expanding the diversity of paraphrases without significant loss of quality.

with the specifications of the rhetorical rule and user-preferences are tossed. For example, vowel phonemes are not evaluated for alliteration. This final culled dictionary is the output of the algorithm. This ends the frequency based rule application algorithm; as of this point, only frequency algorithm has been defined. After the rule processing, the input is then reconstructed in a analyzation module. This module takes outputted dictionary from the rule application step and begins to pre-process the data. In this step, all synonyms are compared to the original token in the context of the original input. This is done using a word-sense disambiguator (the English module uses NLTK WSD) and corpus similarity functionality. Non-similar words, words that don't pass a certain score threshold, are thrown; and the most similar words for each index within a candidate phoneme are chosen and their similarity scores are added together. The candidate phoneme with the highest similarity score is chosen and used for reconstruction. Indices, where values of the original input can be replaced, are modified and the input is put back together into string form. This string is the return value of the Procrustean paraphraser.

Finally, the algorithm takes the candidate paraphrases and evaluates them on the basis of similarity and quality. A distance matrix is generated between the candidate sentences using the Levenshtein metric. Additionally, a distance vector is produced that quantifies the distance of each candidate sentence from the original sentence using the covariance distance of sentence embeddings (sentence embeddings were computed as the average of Google word2vec word embeddings). Next, the sentences are assigned a cluster key using the scipy hierarchical clustering package.[3] Lastly, the sentences are grouped according to Levenshtein cluster and each group is ranked according to similarity to the original sentence using the correlation distance. From each cluster, the highest ranking sentence is selected for output. Thus one sentence is produced per cluster.

To recap, the algorithm starts by using a *free paraphraser* to generate a *sample* of *seed paraphrases*. Next, the *Procrustean paraphraser* forces each member of the sample (including the original sentence) to conform to the *frequency constraint*. Finally, the *paraphrase curator* filters the candidates according to faithfulness and uniqueness.

The function takes the following parameters:

1. An input sentence as a string.
2. A frequency proportion (as discussed above).
3. A similarity threshold (threshold of word2vec similarity of accepted replacement words in the Procrustean paraphraser).
4. An integer that determines the maximum number of suggestions to be offered.
5. A sample size parameter that determines the size of the preprocessing sample generated by the free paraphraser.

4 Data and Evaluation

Eight excerpts from classic works[4] were used as input. Output was manually coded as UG (ungrammatical), NP (grammatical, but not a paraphrase) VP (a valid paraphrase).

Many of the valid paraphrases only made minor modifications. For example, this excerpt from Forster (1969)

> Most of life is so dull that there is nothing to be said about it, and the books and talk that would describe it as interesting are obliged to exaggerate, in the hope of justifying their own existence (Forster, 1969).

had this among its suggetsions:

> Most of life is dull and **b**oring and the **b**ooks and talk that would describe it as interesting are obliged to exaggerate, in the hope of justifying their own existence.

This valid paraphrase was subtle—with the proportion set to 0.2—but arguably did improve the prose. Mostly grammatical non-paraphrases often confused actors in sentences. For example,

> The Mole was a good listener, and Toad, with no one to check his statements or to criticize in an unfriendly spirit, rather let himself go (Grahame, 1908).

became

[3] The maximum number of clusters is given by the user. This parameter determines how many sentences.

[4] You can see the quotes included in the supplementary Colab Notebook.

> The Mole was a good listener and even though he had no one to check **h**is statements or **h**arsh on **h**im, **h**e let **h**imself go.

Ungrammatical sentences were sometimes quite bizarre. For example,

> A certain pride, a certain awe, withheld him from offering to God even one prayer at night, though he knew it was in God's power to take away his life while he slept and hurl his soul hellward ere he could beg for mercy(Joyce, 2008).

became

> **C**hemical element knew it was in God 's executive **c**lemency to take away his life **c**old spell **c**hemical element slept , but **c**hemical element **c**ouldn't offer a prayer **c**hemical element night because of his pride and awe.

It appears as if byzantine prose are especially confounding to the algorithm. Sentence length was used as a proxy for hypotaxis in a binomial regression, with ungrammatical statements as the response. Unfortunately, the test could not confirm a relationship (p=0.88) due to insufficient sample size (n=16). A larger sample was not gathered because the weak performance of the model suggests further improvements would be a better use of time.

The outcomes for each method are summarized in Table 1. The assonance method is not included because it failed to produce any paraphrases for any of the inputs.

Method	Total	UG	NP	VP
Alliteration	3.00 ± 0.74	0.75 ± 0.61	1.13 ± 0.44	1.13 ± 0.58
Rhyming	3.38 ± 0.98	2.00 ± 0.64	1.13 ± 0.58	0.25 ± 0.32

Table 1. Means with 95 percent confidence intervals are presented for each outcome for each method. The confidence interval for Rhyming-VP should be max(0.25 ± 0.32,0) but this was excluded for typesetting reasons.

The alliteration method performed significantly better at producing valid paraphrases compared to the rhyming method (t = 2.59, df = 11, $p < 0.05$).

5 Conclusion

Our team developed a promising new paradigm for organizing and solving the challenges of style editing in an environment where author-computer collaboration is increasingly replacing author-editor partnerships during writing projects.

We believe that this framework has significant implications for computer assisted style editing. The current generation of automated style editors function like augmented spell-checkers even though they have the potential to do so much more. For instance, Grammarly asks authors merely to fill out a brief form on the intended style of a document. It then returns a variety of rudimentary suggestions on spelling, grammar, word choice and sentence complexity. The upper bound of its utility is reached by the amateur user. Contrast this with Adobe PhotoShop. PhotoShop is merely a collection of digital image manipulation algorithms wrapped in a canvas-and-palette user interface. It is festooned with an overwhelming array of buttons, switches, tabs, sidebars and menus. Yet by forcing the user to compromise with the computer, the system achieves tasks that neither human nor machine can accomplish alone.

Now imagine PhotoShop for prose. What would it be like? We believe that, at its core, it would be driven by a vast array of R-style functions that solve constrained paraphrasing problems. Individually these functions would be underwhelming, but together they could form a powerful tool-set. The system would include extensive menus and function documentation; and it would allow functions and function outputs to be inserted directly into the document text. Additionally, since rhetorical figures can be organized by use-cases, we believe that a simple markup language for noting the author's intentions throughout a document could enable sophisticated collaboration between the user and a recommender system that would help guide choices about which functions to use when.

Our first paraphrasing algorithm did quite poorly at discovering valid paraphrases. However, we believe that the architecture we developed can be substantially improved by incrimental modifications.

6 Future Work

6.1 Problems and Solutions

One major problem with the algorithm we developed was that the rules-based word replacement protocol used a list of synonyms as a starting point. Moreover, word replacements were scored according to their similarity to the original word. As a consequence, the context of each word was not considered at any point. This caused the word replacement algorithm to produce some very strange sug-

gestions.[5] Indeed, the very name of this segment of our algorithm—*the Procrustean paraphraser*—sums up its performance problems. While we attempted to compensate for this deficit by sampling unconstrained seed paraphrases as a preprocessing step, there are only so many alternative phrases that will, by chance, be amenable to the context-free method of word replacement we used. Therefore, the biggest missing piece in our algorithm is an n-gram or attention based method for picking replacement words in the Procrustean paraphraser. The most promising approach would be to use a Transformer architecture such as BART (see Lewis et al. (2019) for more details). Transformer architectures are capable of incorporating a sizable word context window into their predictions about which words belong in a given location.

Unfortunately, even with enhanced word suggestion methods, (statistical) rules-based paraphrasing methods still generally have the inherent limit of being unable to leap across the space of possible phrases in the way that pure machine learning based methods do. In addition to the seed paraphrase strategy discussed previously, one way to reduce this limitation would be to develop multiple Procrustean paraphrasers. For example, a greedy algorithm could be separately implemented on the initial sample and the results from both methods could be pooled before clustering and selection.

Additionally, a better free paraphraser would produce better samples. Goyal and Durrett (2020) are developing a promising approach called neural syntactic preordering. Their language model produces more diverse suggestions than others while preserving quality. The SOW-REAP structure of (Goyal and Durrett, 2020)'s transformer would allow for more grammatically unique paraphrasing, something that normal transformers struggle in producing. By taking advantage of the neural parse tree sandwiched between encoder layers, SOW-REAP is able to acknowledge and restructure clauses. It could be effective in actually generating more grammatically based rhetorical figures with the right corpus and sensitivity. This sentiment can also be applied to the phoneme corpus used within the algorithm. CMUdict unfortunately never disclosed the full list of potential output phonemes causing the assonance module to cull certain correct results because they are not identified as vowel phonemes.

[5] See the suggestion for Joyce (2008) in the the Data and Evaluation section.

Lastly, the paraphrase curator could be improved in a variety of ways. First, a sentence faithfulness cutoff could be imposed on the candidate paraphrases before they are clustered and selected. This would ensure that poor suggestions do not rise to the top simply because they got clustered separately from a larger group of superior candidates. Moreover, our sentence embeddings could be significantly improved. We currently use the average of word2vec word embeddings. One easy improvement would be to implement doc2vec for our sentence embeddings.

6.2 Where to next?

First, there is substantial work left in the domain of frequency based rhetorical figures. Rhetorical figures that combine sentence structure with frequency such as anaphora[6], antistrophe[7], anadiplosis[8] would be susceptible a slightly modified version of our current method. Word replacements would merely need to be restricted to one part of a sentence or a sequence of parts of clauses. Moreover, epizeuxis[9], polysyndeton[10] could be implemented using a word insertion algorithm. Additionally, the valance or tone of a paraphrase could be modulated by integrating a metric derived from projecting word and sentence embeddings onto an emotion space obtained through dimensional scaling of thin emotion words.

Additionally, structure based rhetorical figures such as isocolon (parallel structure), chiasmus (mirrored structure) could be implemented simply by biasing a machine learning model with a corpus full of isocolon or chiasmus examples. Our team discovered such a corpus for Chiasmus, curated by Dubremetz (2017). Metaphors and puns could be similarly trained into a model, and then suggestions could be curated on the basis of semantic similarity in the case of the former and phonetic similarity in the case of the latter. There is already substantial work on metaphor generation (Stowe et al., 2020), which could be subsumed under this larger framework.

Purely structural rhetorical figures such as hy-

[6] repeating the first word or phrase of a sentence

[7] repetition of the same word or phrase at the end of several clauses or sentences

[8] repetition of a word at the end of one clause to begin the next

[9] repeating a word or phrase for emphasis

[10] repetition of conjunctions between each clause

potaxis[11] and parataxis[12] could be obtained by breaking or merging sentences and, in the case of parataxis, feeding it into a reluctant paraphraser a la Dras (1997).

Another major domain that is already being independently explored is that of argument mining and argumentation schemes (Walton et al., 2008). While it is not possible to paraphrase any given paragraph as an argument scheme, paraphrasing could play a role in helping authors converge on arguments. Existing argument mining methods (Aker et al., 2017) could be used to suggest likely arguments throughout an author's draft. Moreover, some sequences of sentences would be sufficiently similar to an argumentation scheme that automated paraphrasing could be used to make it conform to recognizable argumentative language. These suggestions could be given wherever available. Rule-based paraphrasing techniques could also be used to insert argument metadata with manual supervision.

Lastly, one major advantage of organizing these diverse concepts in one framework is the ability to explicitly represent the ways they co-occur. Really strong examples of rhetorical figures are almost always hybrids. For example, consider how this quote integrates chiasmus and three separate instances of alliteration:

> Formerly, **w**hen religion **w**as **s**trong and **s**cience **w**eak, **m**en **m**istook **m**agic for **m**edicine; Now, **w**hen **s**cience is **s**trong and religion **w**eak, **m**en **m**istake **m**edicine for **m**agic. —Thomas Szaz

One major problem with algorithms that paraphrase under only one constraint at a time is that they cannot be composed in order to layer rhetorical figures. They would un-do each other. As a consequence, functions for every useful joint rhetorical will have to be explicitly defined. Fortunately, this is not as Sisyphean as it sounds. In the domain of frequency-based rhetorical figures, multiple frequency parameters can easily be included in our existing algorithm. For example, two alliteration parameters could be given, and the algorithm could simply be modified to select two non-overlapping subsets of word replacements that meet the specified criterion. This could just as easily be applied to any combination of alliteration, assonance, internal, rhyme polyptoton[13] and more. Moreover, each frequency parameter could be defined as a range to reduce the chances of producing no candidates meeting the constraints. For structural constraints, anchors could be defined so that certain words would be subject only to entangled replacement. That is, certain words would be linked and if one word is replaced, so is the other. For example, the anchored phrase

> If [advertising]$_1$ had a little more [respect]$_2$ for the [public]$_3$, the [public]$_3$ would have a lot more [respect]$_2$ for [advertising]$_3$. (Dubremetz, 2017)

would have three entangled pairs of words. Similarly, argument scheme metadata could be used to automatically diagram an argument. The prose could be divided into modules. Then, paraphrasing efforts would be constrained to retain logical relationships between the modules. For example, the logical connectors in this argument from slippery slope could be anchored.

> One participant argued that with all the different minority groups, [[once you accept]] one kind of religion as legitimate, [[you are going to have to accept]] many other kinds of religious groups as having a legitimate right to have prayers or religious services in the classroom. This participant said: "It's a Pandora's box. You know that Satanism is a religion too!" (Walton, 1995)

Ultimately, argumentation schemes would be harder to preserve than obvious structure such as chiasmus or isocolon. However, if future paraphrasers are better at preserving the semantics of their input sentences, this issue would be greatly diminished.

As with each individual rhetorical figure, paraphrasing with joint rhetorical figures poses some major yet manageable challenges. Moreover, paraphrasing with joint rhetorical figures could finally take constrained paraphrasing from the "toys" aisle of natural language processing models and place it on a par with speech recognition and other essential services.

[11] byzantine sentences with clauses within clauses

[12] simple prose

[13] repeating the root of a word

References

Ahmet Aker, Alfred Sliwa, Yuan Ma, Ruishen Lui, Niravkumar Borad, Seyedeh Ziyaei, and Mina Ghobadi. 2017. What works and what does not: Classifier and feature analysis for argument mining. In *Proceedings of the 4th Workshop on Argument Mining*, pages 91–96.

Luke Allen. Converting prose into rhyming poetry.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mark Dras. 1997. Reluctant paraphrase: Textual restructuring under an optimisation model. *arXiv preprint cmp-lg/9707001*.

Marie Dubremetz. 2017. mardub1635/corpus-rhetoric.

Edward Morgan Forster. 1969. *A passage to India*. Pearson Education India.

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. *arXiv preprint arXiv:2005.02013*.

Kenneth Grahame. 1908. The wind in the willows.

Randy Harris and Chrysanne DiMarco. 2009. Constructing a rhetorical figuration ontology. In *Persuasive Technology and Digital Behaviour Intervention Symposium*, pages 47–52. Citeseer.

Mohammad Saleh Peter J. Liu Jingqing Zhang, Yao Zhao. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777v3*.

James Joyce. 2008. *A Portrait of the Artist as a Young Man*. OUP Oxford.

Richard A Lanham. 1991. *A handlist of rhetorical terms*. University of California Press Berkeley.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Chris Quirk, Chris Brockett, and William B Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 142–149.

Kevin Stowe, Leonardo Ribeiro, and Iryna Gurevych. 2020. Metaphoric paraphrase generation. *arXiv preprint arXiv:2002.12854*.

Douglas Walton. 1995. A pragmatic theory of fallacy. tuscaloosa.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation schemes*. Cambridge University Press.

A Individual Contributions

The proposal. The original idea for the project was contributed by Oliver. He created a taxonomy of the most common rhetorical figures, based on their structural properties. The taxonomy clustered figures of speech based on the kinds of constraints they impose and the kinds of algorithms that might solve the respective constraints. He presented his taxonomy to the group and strategies for paraphrasing under each kind of constraint were fleshed out more. The proposal document was drafted by Oliver, and reviewed by the team.

The design. The design of the main algorithm was a full team effort. Oliver put forward a mathematical method for parameterizing frequency-based constraints, which Mervin took as a starting point for devising a rules-based algorithm for generating alliterating paraphrases. Mervin devised the initial data flow model. Oliver subsequently elaborated on Mervin's design with the idea of generating a sample of unconstrained paraphrases as a pre-processing step. Jethro devised a strategy for generating paraphrases that are biased towards the structure of chiasmus. Additionally, he tracked down a corpus of chiasmus figures that could be used for training a model.

The alliteration paraphraser. Jethro did the research required to discover and implement several transformer-based unconstrained paraphrasing algorithms. He got the Pegasus paraphraser working so that it could be connected to the rules-based paraphraser. Mervin coded the entire rules-based algorithm (steps 1 and 2), with input from Oliver on how to implement the constraints (an alternative to the greedy algorithm that was originally implemented had to be devised). Mervin's code constituted the bulk of the coding effort. Oliver coded the sentence selection algorithm and wrote the final function that connects the three modules, with extensive input from Jethro. Oliver did the manual paraphrase rating and quality analysis.

The chiasmus paraphraser. While the chiasmust paraphraser was not ready in time, Jethro did extensive research on the topic. Sunni was assigned to write a script that would prepare the Chiasmus corpus for use as a training set, but the work was never fully completed.

The literature review. Oliver wrote the introduction and the section on Allen. Sunni wrote the section on Dras (1997). Jethro wrote the section on Transformers. Oliver edited the final draft. The

team approved the draft.

The presentation. Oliver wrote the presentation, and delivered the section on the project's significance. Sunni edited and delivered the section on rhetorical figures. Jethro edited and delivered the section on the algorithms. Mervin edited and delivered the section on future directions.

The final report. The Introduction and Previous Research sections was taken from the literature review assignment. However, the Machine Learning Approaches subsection was entirely re-written by Jethro. In the methods section, Jethro, Mervin and Oliver each summarized the parts of the algorithm they coded. The Data Evaluation and Conclusion sections were written by Oliver. The Future Work section was written by Oliver and Jethro. Sunni edited the paper for flow.