NL-Augmenter 🐉→🐊
A Framework for Task-Sensitive Natural Language Augmentation

Kaustubh Dhole*†, Varun Gangal†, Sebastian Gehrmann†, Aadesh Gupta†, Zhenhao Li†, Saad Mahamood†, Abinaya Mahendiran†, Simon Mille†, Ashish Shrivastava†, Samson Tan†, Tongshuang Wu†, Jascha Sohl-Dickstein†, Jinho Choi†, Eduard Hovy†, Ondrej Dusek†, Sebastian Ruder†, Sajant Anand, Nagender Aneja, Rabin Banjade, Lisa Barthe, Hanna Behnke, Ian Berlot-Attwell, Connor Boyle, Caroline Brun, Marco Antonio Sobrevilla Cabezudo, Samuel Cahyawijaya, Emile Chapuis, Wanxiang Che, Mukund Choudhary, Christian Clauss, Pierre Colombo, Filip Cornell, Gautier Dagan, Mayukh Das, Tanay Dixit, Thomas Dopierre, Paul-Alexis Dray, Suchitra Dubey, Tatiana Ekeinhor, Marco Di Giovanni, Tanya Goyal, Rishabh Gupta, Louanes Hamla, Sang Han, Fabrice Harel-Canada, Antoine Honoré, Ishan Jindal, Przemyslaw K. Joniak, Denis Kleyko, Venelin Kovatchev, Kalpesh Krishna, Ashutosh Kumar, Stefan Langer, Seungjae Ryan Lee, Corey James Levinson, Hualou Liang, Kaizhao Liang, Zhexiong Liu, Andrey Lukyanenko, Vukosi Marivate, Gerard de Melo, Simon Meoni, Maxime Meyer, Afnan Mir, Nafise Sadat Moosavi, Niklas Muennighoff, Timothy Sum Hon Mun, Kenton Murray, Marcin Namysl, Maria Obedkova, Priti Oli, Nivranshu Pasricha, Jan Pfister, Richard Plant, Vinay Prabhu, Vasile Pais, Libo Qin, Shahab Raji, Pawan Kumar Rajpoot, Vikas Raunak, Roy Rinberg, Nicholas Roberts, Juan Diego Rodriguez, Claude Roux, Vasconcellos P. H. S., Ananya B. Sai, Robin M. Schmidt, Thomas Scialom, Tshephisho Sefara, Saqib N. Shamsi, Xudong Shen, Yiwen Shi, Haoyue Shi, Anna Shvets, Nick Siegel, Damien Sileo, Jamie Simon, Chandan Singh, Roman Sitelew, Priyank Soni, Taylor Sorensen, William Soto, Aman Srivastava, KV Aditya Srivatsa, Tony Sun, Mukund Varma T, A Tabassum, Fiona Anting Tan, Ryan Teehan, Mo Tiwari, Marie Tolkiehn, Athena Wang, Zijian Wang, Zijie J. Wang, Gloria Wang, Fuxuan Wei, Bryan Wilie, Genta Indra Winata, Xinyi Wu, Witold Wydmanski, Tianbao Xie, Usama Yaseen, Michael A. Yee, Jing Zhang, Yue Zhang

Abstract Data augmentation is an important method for evaluating the robustness of and enhancing the diversity of training data for natural language processing (NLP) models. In this paper, we present NL-Augmenter, a new participatory Python-based natural language (NL) augmentation framework which supports the creation of transformations (modifications to the data) and filters (data splits according to specific features). We describe the framework and an initial set of 117 transformations and 23 filters for a variety of NL tasks annotated with noisy descriptive tags. The transformations incorporate noise, intentional and accidental human mistakes, socio-linguistic variation, semantically-valid style, syntax changes, as well as artificial constructs that are unambiguous to humans. We demonstrate the efficacy of NL-Augmenter by using its transformations to analyze the robustness of popular language models. We find different models to be differently challenged on different tasks, with quasi-systematic score decreases. The infrastructure, datacards, and robustness evaluation results are publicly available on GitHub for the benefit of researchers working on paraphrase generation, robustness analysis, and low-resource NLP.

🇪🇸 El aumento de datos es un método importante para evaluar la solidez y mejorar la diversidad del entrenamiento datos para modelos de procesamiento de lenguaje natural (NLP). 🇮🇳 इस लेख में, हम एनएल-ऑगमेंटर का प्रस्ताव करते हैं - एक नया भागीदारी पूर्वक, पायथन में बनाया गया, लैंग्वेज (एनएल) ऑग्मेंटेशन फ्रेमवर्क जो ट्रांसफॉर्मेशन (डेटा में बदलाव करना) और फीलटर (फीचर्स के अनुसार डेटा का भाग करना) के नीरमान का समर्थन करता है। 🇨🇳 我们描述了NL-Augmenter框架及其初步包含的117种转换和23个过滤器，并大致标注分类了一系列可适配的自然语言任务 🇮🇷 این دگرگونی ها شامل نویز، اشتباهات عمدی و تصادفی انسانی، تنوع اجتماعی-زبانی، سبک معنایی معتبر، تغییرات نحوی و همچنین ساختارهای مصنوعی است که برای انسان ها مبهم است. 🇵🇪 NL-Augmenterpa allin kaynintam qawachiyku, tikrakuyninkunata servichikuspayku, chaywanmi qawariyku modelos de lenguaje popular nisqapa allin takyasqa kayninta. 🇮🇩 Kami menemukan model yang berbeda ditantang secara berbeda pada tugas yang berbeda, dengan penurunan skor kuasi-sistematis. Infrastruktur, kartu data, dan hasil evaluasi ketahanan dipublikasikan tersedia secara gratis di GitHub untuk kepentingan para peneliti yang mengerjakan pembuatan parafrase, analisis ketahanan, dan NLP sumber daya rendah.

*Corresponding author: kdhole@emory.edu
†Organizers & Steering Commitee

Northern European Journal of Language Technology

1 Introduction

Data augmentation, the act of creating new datapoints by slightly modifying copies or creating synthetic data based on existing data, is an important component in the robustness evaluation of models in natural language processing (NLP) and in enhancing the diversity of their training data. Most data augmentation techniques create examples through transformations of existing examples which are based on prior task-specific knowledge (Feng et al., 2021; Chen et al., 2021). Such transformations seek to disrupt model predictions or can be used as training candidates for improving regularization and denoising models, for example through consistency training (Xie et al., 2020). Figure 1 illustrates a number of possible transformations for a sample sentence.

However, the vast majority of transformations do not alter the structure of examples in drastic and meaningful ways, rendering them qualitatively less effective as potential training or test examples. Moreover, different NLP tasks may benefit from transforming different linguistic properties. Changing the word "happy" to "very happy" in an input is more relevant for sentiment analysis than for summarization (Mille et al., 2021). Despite this, many transformations are universally useful, for example changing places to ones from different geographic regions, or changing names to those from different cultures. Hence, a single repository that aggregates both task-specific and task-independent transformations will lower the barrier to entry for creating appropriate augmentation suites for any task.

Another advantage of supporting a broad range of transformations is the ability to capture the long-tailed nature and high diversity of surface forms of natural language (Bamman, 2017). The current paradigm of testing models on data drawn i.i.d. from long-tailed distribution results in the head of the distribution being emphasized even in the test dataset and rare phenomena implicitly ignored by aggregate performance numbers. Researchers have thus argued for more fine-grained breakdowns of results in ways that capture these under-represented groups (Mitchell et al., 2019). However, the identification of these groups depends on and benefits from different cultural backgrounds and expertise. To capture a wide range of backgrounds, we thus capitalize on the "wisdom-of-researchers" and develop NL-Augmenter in a participatory framework.

NL-Augmenter is a Python-based natural language (NL) augmentation framework that aims to enable more diverse and better characterized data during testing and training.[1] Drawing upon researchers from computational linguistics, NLP, and other related fields, we collect 117 different ways to augment data for NL tasks. To encourage task-specific implementations, we link each transformation to a widely-used data format (e.g. text pair, a question-answer pair, etc.) along with the task types (e.g. entailment, tagging, etc.) that they support. NL-Augmenter also provides more than 23 different filters, which can be used to create input subpopulations, according to features such as input complexity, input size, etc. Unlike a transformation, the output of a filter is a boolean value, indicating whether the input meets the filter criterion, e.g., whether the input text is classified as toxic. We evaluate the robustness of four common pre-trained language models on four different tasks by testing their performance on perturbed test sets. The results demonstrate how NL-Augmenter can easily corroborate prior findings that current pre-trained models are strongly affected by small perturbations in texts. Additionally, we expect NL-Augmenter to be an effective tool for training data augmentation to develop models that are robust to diverse language characteristics.

2 Related Work

Participatory Benchmarks & Wisdom-of-Researchers Addressing the problem of under-resourced African languages in machine translation, Masakhane adopted a participatory approach to construct benchmarks for over thirty languages (Nekoto et al., 2020). Such collaborative approaches are becoming increasingly common (Cahyawijaya et al., 2022) in NLP to keep up with the rapid pace of NLP progress via benefitting from collaboration. The Generation Evaluation and Metrics benchmark (Gehrmann et al., 2021, 2022), which started the development of NL-Augmenter, is a participatory project to document and improve evaluation processes in natural language generation. BIG-Bench[2] is a collaborative framework to collect few-shot tasks that gauge the abilities of large, pretrained language models. DynaBench (Kiela et al., 2021) iteratively evaluates models in a human-in-the-loop fashion by enabling humans to construct challenging examples. SyntaxGym (Gauthier et al., 2020) provides a platform for researchers to contribute and use evaluation sets with a focus on targeted syntactic evaluation of Language Models (LMs), particularly psycho-linguistically motivated ones. The collaboration process for NL-Augmenter is inspired by these projects allowing us to reach for a much broader scope and to collect transformations that operate on a larger variety of tasks and model types. Through our participatory approach, the lived experiences of a diverse group of individuals enable identifying and codifying an extensive list dimensions of variation

[1] https://github.com/GEM-benchmark/NL-Augmenter

[2] https://github.com/google/BIG-bench

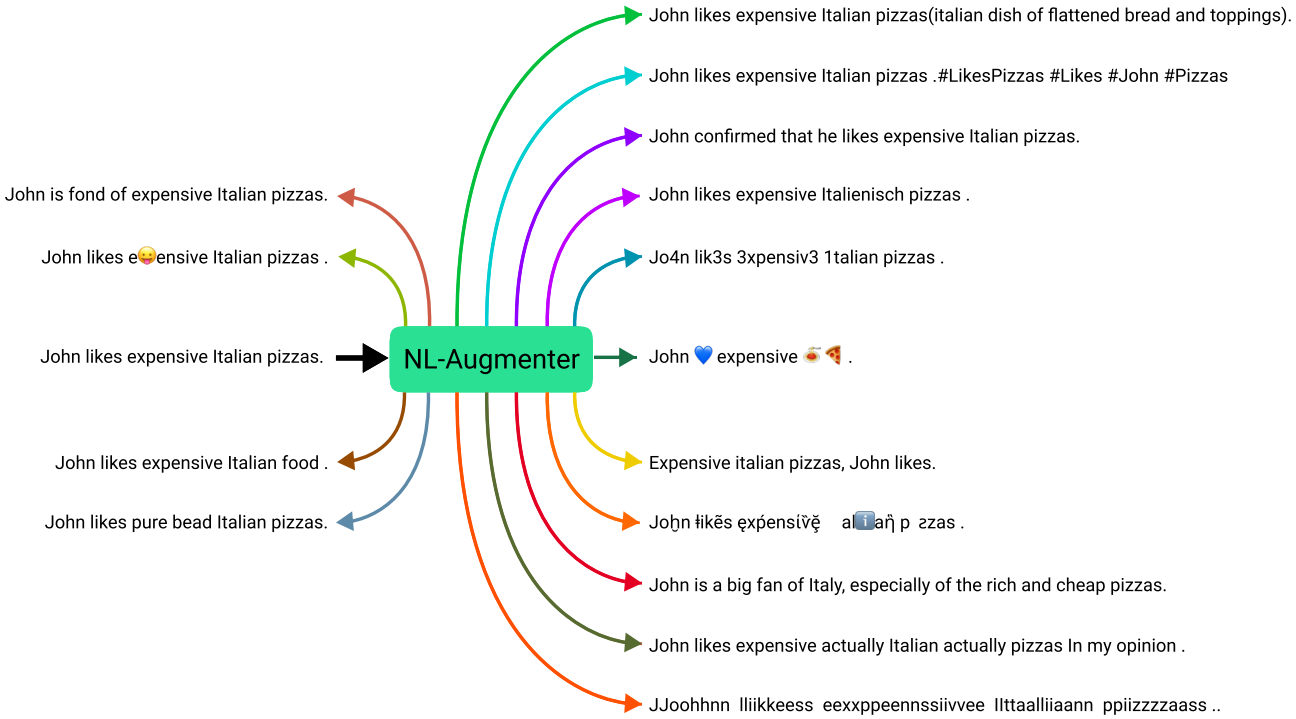

Figure 1: A few randomly chosen transformations of NL-Augmenter for the original sentence *John likes expensive pizzas.* While the meaning (almost) always remains the same and identifiable by humans, models can have a much harder time representing the transformed sentences.

which are encoded as executable transformations (Tan et al., 2021b). Leveraging the wisdom-of-the-crowd (Galton, 1907; Yi et al., 2010) is common in our field of NLP, often through the use of crowdsourcing platforms like Amazon Mechanical Turk that provide access to many raters, although not representative of the broader population (Fort et al., 2011). To harness the wisdom-of-researchers instead, we follow the example by BIG-bench which is hosted on GitHub and offers co-authorship in exchange for task contribution.

Robustness Evaluation Tools There are many projects with similar goals that inspired NL-Augmenter. For example Gardner et al. (2020) create "contrast" sets of perturbed test examples. In their approach, each example is manually perturbed, which may lead to higher-quality results but is costly to replicate for each new task due to scale and annotator cost. TextAttack (Morris et al., 2020) and TextFlint (Wang et al., 2021a) are libraries to conduct adversarial evaluations of English and Chinese models. They cover linguistic and task-specficic transformations, adversarial attacks, and subpopulation analyses. In contrast, while the majority of transformations are focused on English, NL-Augmenter supports many more languages and each contribution can specify a set of supported languages.

Robustness Gym (Goel et al., 2021) unifies four different types of robustness tests — subpopulations, transformations, adversarial attacks, and evaluation sets — in a single interface in their released library. While conceptually similar, the design of NL-Augmenter puts an emphasis on modularity to enable a low barrier of entry for contributors, which is reflected in its size and diversity. Checklist (Ribeiro et al., 2020) argues for the need to go beyond simple accuracy and evaluate the model on basic linguistic capabilities, for example their response to negations. Polyjuice (Wu et al., 2021) perturbs examples using GPT-2 — though this is automatic and scalable, it offers limited control over type of challenging examples generated, making fine-grained analysis beyond global challenge-set level difficult. In contrast, our method offers a richer taxonomy with 117 (and growing) transformations for extensive analysis and comparison.

Tan et al. (2021b) propose decomposing each real world environment into a set of dimensions before using randomly sampled and adversarially optimized transformations to measure the model's average- and worst-case performance along each dimension. NL-Augmenter can be used, out-of-the-box, to measure average-case performance and we plan to extend it to support worst-case evaluation.

Library	#Transform.	Task-specific?	Filters?	Diversity of Resources
TextAttack	∗19	✗	✗	WordNet (WD), Language Models (LM)
OpenAttack	15	✗	✗	WN, LM
NLPAug	16	✗	✗	WN, LM, PPDB
Checklist	12	✗	✗	WN, LM, Wikidata
Robustness Gym	< 20	✗	✓	WN
TextFlint	80	✓	✓	LM
NL-Augmenter	∗117	✓	✓	WN, LM, Wiki, Geographies, Abbreviations, NeoPronouns, PropBank, Implicatives, Emojis, etc.

Table 1: Comparison of NL-Augmenter with other data augmentation and robustness evaluation libraries. *These are configurable transformations with multiple child transformations.

3 NL-Augmenter

NL-Augmenter is a crowd-sourced suite to facilitate rapid augmentation of data for NLP tasks to assist in training and evaluating models. NL-augmenter was introduced in Mille et al. (2021) in the context of the creation of evaluation suites for the GEM benchmark (Gehrmann et al., 2021, 2022); three types of evaluation sets were proposed: (i) transformations, i.e. original test sets are perturbed in different ways (e.g. back-translation, introduction of typographical errors, etc.), (ii) subpopulations, i.e. test subsets filtered according to features such as input complexity, input size, etc.; and (iii) data shifts, i.e. new test sets that do not contain any of the original test set material.

In this paper, we present a participant-driven repository for creating and testing **transformations** and **filters**, and for applying them to all dataset splits (training, development, evaluation) and to all NLP tasks (NLG, labeling, question answering, etc.). As shown by Mille et al. (2021), applying filters and transformations to development/evaluation data splits allows for testing the robustness of models and for identifying possible biases; on the other hand, applying transformations and filters to training data (data augmentation) allows for possibly mitigating the detected robustness and bias issues (Wang et al., 2021b; Pruksachatkun et al., 2021; Si et al., 2021).

A majority of the augmentations that the framework supports are transformations of single sentences that aim to paraphrase these sentences in various ways. NL-Augmenter loosens the definition of "transformations" from the logic-centric view of strict equivalence to the more descriptive view of linguistics, closely resembling Bhagat and Hovy (2013)'s "quasi-paraphrases". We extend this to accommodate noise, intentional and accidental human mistakes, socio-linguistic variation, semantically-valid style, syntax changes, as well as artificial constructs that are unambiguous to humans (Tan et al., 2021b). Some transformations vary the socio-linguistic perspective permitting a crucial source of variation wherein language goals span beyond conveying ideas and content.

In this section, we provide organizational details, list the transformations and filters that the repository currently contains, and we present the list of tags we associated to transformations and filters and how we introduced them.

3.1 Participatory Workshop on GitHub

A workshop was organized towards constructing this full-fledged participant-driven repository. Unlike a traditional workshop wherein people submit papers, participants were asked to submit python implementations of transformations to the GitHub repository. Organizers of this workshop created a base repository extending Mille et al. (2021)'s NLG evaluation suite and incorporated a set of *interfaces*, each of which catered to popular NL example formats. This formed the backbone of the repository. A sample set of transformations and filters along with evaluation scripts were provided as starter code. Figure 2 shows an annotated code snippet of a submission. Following the format of BIG-Bench's review process, multiple review criteria were designed for accepting contributions. The review criteria (see Appendix C) guided participants to follow a style guide, incorporate test cases in JSON format, and encouraged novelty and specificity. Apart from the general software development advantages of test cases, they made reviewing simpler by providing an overview of the transformation's capability and scope of generations.

3.2 Review Process

Each participant was expected to follow the review criteria mentioned in Figure 3. Rule-based transformations depending on well-studied lexical resources like WordNet, Wikipedia, PropBank, Implications were almost always selected due to their high precision as well as their ability to offer diverse synonymy. Machine Learning based transformations (e.g. Transformers fine-tuned on paraphrase datasets) were encour-

aged if they included either previously reported or newly measured metrics. ML-based transformations based on previously published work were thus also accepted. Duplicate submissions were rejected.

Format of a Transformation

The name of the transformation, **ReplaceFinancialAmount** followed by the interface **SentenceOperation**.

The **tasks** that the transformation is applicable to.The **languages** for which transformations are generated. And the relevant **keywords** which categorise the transformation.

```
class ReplaceFinancialAmount(SentenceOperation):
    tasks = [
        TaskType.TEXT_CLASSIFICATION,
        TaskType.TEXT_TO_TEXT_GENERATION,]
    languages = ["en"]
    keywords = [
        "lexical",
        "rule-based",
        "external-knowledge-based",
        "possible-meaning-alteration",
        "high-precision"]

    def __init__(self, seed: int = 0, max_outputs: int = 1):
        super().__init__(seed=seed, max_outputs=max_outputs)

    def generate(self, sentence: str) -> List[str]:

        """
        The actual logic of the transformation. The
'generate' method takes in a sentence and returns
multiple transformed sentences.

        """
        return transformed_sentences
```

Figure 2: Participants were expected to write their python class adhering to the above format.

Those transformations which resulted in immeasurable meaning change or untracked label changes were rejected. During the peer review, reviewers examined example outputs to decide whether a transformation had immeasurable meaning change. Reviewers were asked to instigate constructive discussions and suggest improvements to the code and the transformations. As each transformation was paired with at least 2 reviewers[3] and the submissions were discussed publicly, most of these transformations had to improve & resubmit modified versions. The discussions between reviewers and participants leading up acceptances or rejections are available publicly to encourage transparency and reproducibility as well as foster ancillary projects.

Since reviewers were the main guarantors of quality, it was imperative to provide a fair and qualitative review to participants and hence submissions were scrutinised by both participants as well as the organizers. From our initial advertising on relevant mailing lists and personally emailing authors of the relevant papers (i.e. papers focused on paraphrasing, augmentation, adversarial learning and robustness analysis) helped us in obtaining a diverse pool of volunteers. The reviewers were affiliated to about 90 organisations during the course of review out of which approximately two-thirds were academic and the rest were industrial in nature. To ensure that the submissions adhere to the larger goals of the project we let organizers have the final say of acceptance, much like meta-reviewers in conferences.

[3]Some submissions also received up to 5-6 reviews.

3.3 Transformations and filters

We received a total of 170 submissions out of which 117 transformations and 23 filters were accepted and merged. They have been listed in Tables 2 and 3 respectively (and alphabetically ordered according to the submission name in the repository). For each transformation/filter, a link to the corresponding Appendix subsection is provided, where a detailed description, illustrations and an external link to the implementation in the NL-Augmenter repository can be found.

3.4 Tags for the classification of perturbations

We defined a list of tags which are useful for an efficient navigation in the pool of existing perturbations and for understanding the performance characteristics of the contributed transformations and filters (see e.g. the robustness analysis presented in Section 7). There are three main categories of tags: (i) General properties tags, (ii) Output properties tags, and (iii) Processing properties tags.

General properties tags are shown in Table 4, and cover the type of the augmentation, i.e. whether it is a transformation or a filter (*Augmented set type*), its general purpose, i.e. whether it is intended for augmentation, robustness, etc. (*General purpose*), for which NLP tasks the created data will be useful (*Task type*), to which languages it has been applied (*Language(s)*), and on which linguistic level of representation it operates, i.e. semantic, syntactic, lexical, etc. (*Linguistic level*).

Output properties tags, shown in Table 5, apply to transformations only; they provide indications about how the data was affected during the respective transformations. There are currently six properties in this category: one to capture the number of different outputs that a transformation can produce (*Output/Input ratio*), one to capture in which aspect the input and the output are alike (*Input/Ouptut similarity*), and four to capture intrinsic qualities of the produced text or structured data, namely how were the meaning, the grammaticality, the readability and the naturalness affected by the transformation (respectively *Meaning preservation*, *Grammaticality preservation*, *Readability preservation* and *Naturalness preservation*). Note that apart from Output/Input ratio, the output properties tags need to be specified manually for each transformation/filter (see Section 3.5), and are thus subject to the interpretation of the annotator.

Transformation	App.	Transformation	App.
Abbreviation Transformation	A.1	Mix transliteration	A.60
Add Hash-Tags	A.2	MR Value Replacement	A.61
Adjectives Antonyms Switch	A.3	Multilingual Back Translation	A.62
AmericanizeBritishizeEnglish	A.4	Multilingual Dictionary Based Code Switch	A.63
AntonymsSubstitute	A.5	Multilingual Lexicon Perturbation	A.64
Auxiliary Negation Removal	A.6	Causal Negation and Strengthening	A.65
AzertyQwertyCharsSwap	A.7	Question Rephrasing transformation	A.66
BackTranslation	A.8	English Noun Compound Paraphraser [N+N]	A.67
BackTranslation for Named Entity Recognition	A.9	Number to Word	A.68
Butter Fingers Perturbation	A.10	Numeric to Word	A.69
Butter Fingers Perturbation For Indian Languages	A.11	OCR Perturbation	A.70
Change Character Case	A.12	Add Noun Definition	A.71
Change Date Format	A.13	Pig Latin Cipher	A.72
Change Person Named Entities	A.14	Pinyin Chinese Character Transcription	A.73
Change Two Way Named Entities	A.15	SRL Argument Exchange	A.74
Chinese Antonym and Synonym Substitution	A.16	ProtAugment Diverse Paraphrasing	A.75
Chinese Pinyin Butter Fingers Perturbation	A.17	Punctuation	A.76
Chinese Person NE and Gender Perturbation	A.18	Question-Question Paraphraser for QA	A.77
Chinese (Simplified and Traditional) Perturbation	A.19	Question in CAPS	A.78
City Names Transformation	A.20	Random Word Deletion	A.79
Close Homophones Swap	A.21	Random Upper-Case Transformation	A.80
Color Transformation	A.22	Double Context QA	A.81
Concatenate Two Random Sentences (Bilingual)	A.23	Replace Abbreviations and Acronyms	A.82
Concatenate Two Random Sentences (Monolingual)	A.24	Replace Financial Amounts	A.83
Concept2Sentence	A.25	Replace Numerical Values	A.84
Contextual Meaning Perturbation	A.26	Replace Spelling	A.85
Contractions and Expansions Perturbation	A.27	Replace nouns with hyponyms or hypernyms	A.86
Correct Common Misspellings	A.28	Sampled Sentence Additions	A.87
Country/State Abbreviation	A.29	Sentence Reordering	A.88
Decontextualisation of the main Event	A.30	Emoji Addition for Sentiment Data	A.89
Diacritic Removal	A.31	Shuffle Within Segments	A.90
Disability/Differently Abled Transformation	A.32	Simple Ciphers	A.91
Discourse Marker Substitution	A.33	Slangificator	A.92
Diverse Paraphrase Generation	A.34	Spanish Gender Swap	A.93
Dislexia Words Swap	A.35	Speech Disfluency Perturbation	A.94
Emoji Icon Transformation	A.36	Paraphrasing through Style Transfer	A.95
Emojify	A.37	Subject Object Switch	A.96
English Inflectional Variation	A.38	Sentence Summarizaiton	A.97
English Mention Replacement for NER	A.39	Suspecting Paraphraser for QA	A.98
Filler Word Augmentation	A.40	Swap Characters Perturbation	A.99
Style Transfer from Informal to Formal	A.41	Synonym Insertion	A.100
French Conjugation Substitution	A.42	Synonym Substitution	A.101
Gender And Culture Diversity Name Changer	A.43	Syntactically Diverse Paraphrasing	A.102
Neopronoun Substitution	A.44	Subsequence Substitution for Seq. Tagging	A.103
Gender Neutral Rewrite	A.45	Tense	A.104
GenderSwapper	A.46	Token Replacement Based on Lookup Tables	A.105
GeoNames Transformation	A.47	Transformer Fill	A.106
German Gender Swap	A.48	Added Underscore Trick	A.107
Grapheme to Phoneme Substitution	A.49	Unit converter	A.108
Greetings and Farewells	A.50	Urban Thesaurus Swap	A.109
Hashtagify	A.51	Use Acronyms	A.110
Insert English and French Abbreviations	A.52	Visual Attack Letter	A.111
Leet Transformation	A.53	Weekday Month Abbreviation	A.112
Lexical Counterfactual Generator	A.54	Whitespace Perturbation	A.113
Longer Location for NER	A.55	Context Noise for QA	A.114
Longer Location Names for testing NER	A.56	Writing System Replacement	A.115
Longer Names for NER	A.57	Yes-No Question Perturbation	A.116
Lost in Translation	A.58	Yoda Transformation	A.117
Mixed Language Perturbation	A.59		

Table 2: List of transformations and link to their detailed descriptions in Appendix

Filter	App.	Filter	App.
Code-Mixing Filter	B.1	Polarity Filter	B.13
Diacritics Filter	B.2	Quantitative Question Filter	B.14
Encoding Filter	B.3	Question type filter	B.15
Englishness Filter	B.4	Repetitions Filter	B.16
Gender Bias Filter	B.5	Phonetic Match Filter	B.17
Group Inequity Filter	B.6	Special Casing Filter	B.18
Keyword Filter	B.7	Speech-Tag Filter	B.19
Language Filter	B.8	Token-Amount filter	B.20
Length Filter	B.9	Toxicity Filter	B.21
Named-entity-count Filter	B.10	Universal Bias Filter	B.22
Numeric Filter	B.11	Yes/no question filter	B.23
Oscillatory Hallucinations Filter	B.12		

Table 3: List of filters and link to their detailed descriptions in Appendix

Property	Definition	Tags
Augmented set type	Transformation or Filter (Subpopulation)?	Filter, Transformation, Multiple (specify), Unclear, N/A
General purpose	What will the data be used for? Augmenting training data? Testing robustness? Finding and fixing biases? Etc.	Augmentation, Bias, Robustness, Other (specify), Multiple (specify), Unclear, N/A
Task type	For which NLP task(s) will the perturbation be beneficial?	Quality estimation, Question answering, Question generation, RDF-to-text, Table-to-text generation, Sentiment analysis, Text classification, Text tagging, Text-to-text generation
Language(s)	To which language(s) is the perturbation applied?	*
Linguistic level	On which linguistic level does the perturbation operate?	Discourse, Semantic, Style, Lexical, Syntactic, Word-order, Morphological, Character, Other (specify), Multiple (specify), Unclear, N/A

Table 4: Criteria and possible tags for **General Properties** of perturbations

Processing properties tags, shown in Table 6, capture information related to the type of processing applied on the input (*Input data processing*), the type of algorithm used (*Algorithm type*), how it is implemented (*Implementation*), its estimated precision and recall (*Precision/recall*) and computational complexity (*Computational complexity / Time*), and whether an accelerator is required to apply the transformation/filter (*GPU required?*).

3.5 Tag retrieval and assignment

Transformation and filters are assigned tags for each of the properties listed in Tables 4-6. There are two sources for the tags: (i) assigning them manually, and (ii) using existing metadata embedded in the respective source code implementations of each given transformation and filter. The in-code metadata (see e.g. the *Keywords* field in Figure 2) provides descriptions for each one identifiable aspects such as the language(s) supported, the type of task that the transformation or filter is applicable for, and other characteristical keywords. The specification and type of this metadata was predefined as a requirement for all contributors to the NL-Augmenter project to enable identification of the type of transformation of filter being written by their respective author(s).

Having a language tag as shown in the sample transformation in Figure 2 separately was crucial to emphasize and encourage multi-lingual transformations and filters.

This metadata was initially collected through the creation of an automated script which programmatically iterated through each transformation and filter and gathered all stated metadata. The metadata was then mapped by the script into discrete property groups as defined in Tables 4-6. All contributing authors were invited to review the initially collected metadata and, where possible, add additional data.

Property	Definition	Tags
Output/input ratio	Does the transformation generate one single output for each input, or a few, or many?	=1, >1 (Low), >1 (High), Multiple (specify), Unclear, N/A
Input/output similarity	On which level are the input and output similar (if applicable)?	Aural, Meaning, Visual, Other (specify), Multiple (specify), Unclear, N/A
Meaning preservation	If you compare the output with the input, how is the meaning affected by the transformation?	Always-preserved, Possibly-changed, Always-changed, Possibly-added, Always-added, Possibly-removed, Always-removed, Multiple (specify), Unclear, N/A
Grammaticality preservation	If you compare the output with the input, how is the grammatical correctness affected by the transformation?	Always-preserved, Possibly-impaired, Always-impaired, Possibly-improved, Always-improved, Multiple (specify), Unclear, N/A
Readability preservation	If you compare the output with the input, how is the easiness of read affected by the transformation?	Always-preserved, Possibly-impaired, Always-impaired, Possibly-improved, Always-improved, Multiple (specify), Unclear, N/A
Naturalness preservation	If you compare the output with the input, how is the naturalness of the text affected by the transformation?	Always-preserved, Possibly-impaired, Always-impaired, Possibly-improved, Always-improved, Multiple (specify), Unclear, N/A

Table 5: Criteria and possible tags for **Output Properties** of perturbations (applicable to transformations only)

Property	Definition	Tags
Input data processing	What kind of NL processing is applied to the input?	Addition, Chunking, Paraphrasing, Parsing, PoS-Tagging, Removal, Segmentation, Simplification, Stemming, Substitution, Tokenisation, Translation, Other (specify), Multiple (specify), Unclear, N/A
Implementation	Is the perturbation implemented as rule-based or model-based?	Model-based, Rule-based, Both, Unclear, N/A
Algorithm type	What type of algorithm is used to implement the perturbation?	API-based, External-knowledge-based, LSTM-based, Transformer-Based, Other (specify), Multiple (specify), Unclear, N/A
Precision/recall	To what extent does the perturbation generate what it intends to generate (precision)? To what extent does the perturbation return an output for any input (recall)?	High-precision-High-recall, High-precision-Low-recall, Low-precision-High-recall, Low-precision-Low-recall, Unclear, N/A
GPU Required?	Is GPU needed to run the perturbation?	No, Yes, Unclear, N/A
Computational complexity / Time	How would you assess the computational complexity of running the perturbation? Does it need a lot of time to run?	High, Medium, Low

Table 6: Criteria and possible tags for **Processing Properties** of perturbations

4 Robustness Analysis

All authors of the accepted perturbations were asked to provide the task performance scores for each of their respective transformations or filters. In Section 4.1 we provide details on how the scores were obtained, and in Section 7 we provide a first analysis of these scores.

4.1 Experiment

The perturbations are currently split into three groups, according to the task(s) they will be evaluated on: text classification tasks, tagging tasks, and question-answering tasks. For experiments we focus on text classification and its relevant perturbations. We compare the models' performance on the original test data and on the perturbed data. The percentage of sentences be-

ing changed by a transformation (*transformation rate*) and the percentage of performance drop on the perturbed data compared to the performance on the original data (*score variation*) are reported.

Tasks. We choose four evaluation datasets among three English NLP tasks: (1) sentiment analysis on both short sentences (SST-2 (Socher et al., 2013)) and full paragraphs (IMDB Movie Review (Maas et al., 2011)), (2) Duplicate question detection (QQP) (Wang et al., 2019a), and (3) Natural Language Inference (MNLI) (Williams et al., 2017). These tasks cover both classifications on single sentences, as well as pairwise comparisons, and have been widely used in various counterfactual analysis and augmentation experiments (Wu et al., 2021; Kaushik et al., 2019; Gardner et al., 2020; Ribeiro et al., 2020).

Evaluation models. We represent each dataset/task with its corresponding most downloaded large model hosted on Huggingface (Wolf et al., 2020), resulting in four models for evaluation: `roberta-base-SST-2`, `roberta-base-imdb`, `roberta-large-mnli`, and `bert-base-uncased-QQP`.

Perturbation strategy. For each task, we perturb a random sample of 20% of the validation set. Since all the transformations are on single text snippets, for datasets with sentence pairs, i.e., QQP and MNLI, we perturb the first question and the premise sentence, respectively.

4.2 Results and Analysis

In this section, Tables 7 to 17 show the results of the robustness analysis performed on the four datasets described in Section 4.1 and presented according to the tags introduced in Section 3.4. As we will see further, many of the tags relay interesting qualitative assessments while in some cases there is no direct correlation.

General purpose (Table 7): Transformations designed with a "robustness testing" objective displayed mean performance drops between 9% and 13.7% across models. Interestingly, 34 sentence transformations designed for "augmentation" tasks showed similar mean robustness drops ranging between 4% and 13%, emphasizing the need to draw on the paraphrasing literature to improve robustness testing.

Task type (Table 8): The results table shows that there is not necessarily a correlation between which task a transformation is marked to be relevant for and which task it actually challenges the robustness of the models on.

Linguistic level (Table 9): Transformations making character level and morphological changes were able to show drastic decreases in the level of performance compared to those making lexical or syntactic changes. These drops in performance were consistent across all four models. `roberta-large` finetuned on the MNLI dataset was the most brittle - character-level transformations on an average dropped performance by over 31% and morphological changes dropped it by 28% while those which made lexical changes displayed a mean drop of 4.4%. The `visual_attack_letters` (A.111) transformation, which replaces characters with similarly looking ones (like *y* and *v*), shows a large accuracy drop from 94% to 56% on the 'roberta-base' model fine tuned on SST. 'bert-base-uncased' fine-tuned on the QQP dataset drops from 92 to 69. `roberta-large-mnli` drops from 91 to 47. In the case of `visual_attack_letters`, one can easily conceive a scenario in which a model is applied to OCR text which likely exhibit similar properties. In this case, one may expect similarly poor performance, arguably attributed to a narrow set of characters that the models have been exposed to. This drop could potentially be alleviated by adversarial training. As is shown in previous work (Si et al., 2021), training on augmented data improves the performance on the test set with same perturbations.

Meaning preservation (Table 11): 22 transformations which were marked as highly meaning preserving surprisingly showed a larger average performance drop as compared to 20 of those which were marked as possibly meaning changing. Not discounting the possibility of the noisiness of the transformation's logic, we believe further investigation could help understand whether models focus on the meaning of words or sentences or take shortcuts by focusing on commonly occuring surface forms associated with a particular prediction, as was already shown for some phenomena by McCoy et al. (2019), among others.

Grammaticality preservation (Table 12): Preserving grammaticality did not correlate with high robustness. Transformations marked as grammaticality `always-preserved` showed significant average drops of 10.6%, 8.1% and 4.6% across `roberta-base-SST-2`, `roberta-large-mnli` and `bert-base-uncased-QQP` respectively. For example, the `grapheme_to_phoneme` transformation showed drastic drops in performance: 13%, 20% and 13% respectively.

Readability and Naturalness (Tables 13-14): In general, as expected, the transformations tagged as modifying the readability or naturalness show large drops across all tasks and models, in particular the ones tagged as "always imparing" the input.

Unsurprisingly, many of the injected perturbations, despite being artificial would not distract human readers from the actual meaning and intent of the text (e.g. `simple_ciphers` transformation (A.91)). Character level perturbations might not distract human readers as much as compared to word level perturbations but the above language models on the other hand behaved contrarily. Such departure from learning mean-

ingful abstractions is further validated with the low correlation of grammaticality preservation and robustness. These results further re-question how we can expand these models from being just pure statistical learners to those which can incorporate meaning and surface-level abstraction, both across natural as well as artificial constructs. The large drops in performance of such perturbations necessitate looking at expanding training sets with even artificial data sources as well expand our definitions of text similarity from pure linguistic ones to those which abstract morphological, visual and other errors which can be unambiguous to humans.

Tables 10, 15, 16 and 17 show the robustness scores for **Input/Output similarity**, **Input processing**, **Implementation** and **Algorithm type** respectively. The score drops for these criteria may not be easily interpretable; e.g. that model-based implementations showed comparatively larger average drops as compared to rule-based implementations may not be due to the difference in implementation, but rather to which transformations were implemented that way .

5 Discussion and Broader Impact

Limitations In Section 7, we analyze the results of applying some of the transformations on existing datasets and running models on the perturbed data. Even though it was not possible to test all of the currently existing perturbations due to time constraints, the overall results show that the tested perturbations do pose a challenge to different models on different tasks, with quasi-systematic score drops. However, with so many transformations applied to four different datasets, the presented robustness analysis can only be shallow, and a separate analysis of each transformation would be needed in order to get more informative insights. Second, our superficial analysis above relies on tags which were in many cases annotated by hand, and some of the surprising results (e.g. meaning-preserving are more challenging than non-meaning-preserving transformations) may reflect a lack of consistency in the annotations. We believe that assessing the quality of the tag assignment so as to ensure a high inter-annotator agreement will be needed for reliable analyses in the future. Finally, the current robustness analysis only shows that the perturbations are effective for detecting a possible weakness in a model; further experiments are needed to demonstrate that the perturbations can also help mitigating the weaknesses they bring to light.

Dilution of Contributions While this is not our intent, there is a risk in large scale collections of work like this that individual contributions are being less appreciated than releasing them as a standalone project. This risk is a tradeoff with the advantage that it becomes much easier to switch between different transformations, which can lead to a better adoption of introduced methods. To proactively give appropriate credit, each transformation has a data card in the form of a standard README file mentioning the contributors and all participants are listed as co-authors of this paper. We further encourage all users of our repository to cite the work that a specific implementation builds on, if appropriate. The relevant citations are listed on the respective data cards and in the description in the appendix. In the same vein, there is a risk of NL-Augmenter as a whole to monopolize the augmentation space due to its large scope, leading to less usage of related work which may cover additional transformations or filters. While this is not our intention and we actively worked with contributors to related repositories to integrate their work, we encourage researchers to try other solutions as well.

Participatory Setup Conducting research in environments with a shared mission, a low barrier of entry, and directly involving affected communities was popularizied by Nekoto et al. (2020). This kind of participatory work has many advantages, most notably that it changes the typically prescriptive research workflow toward a more inclusive one. Another advantage is that through open science, anyone can help shape the overall mission and improve the end result. Following the related BIG-bench(Srivastava et al., 2022) project, we aimed to design NL-Augmenter in a similar spirit – by providing the infrastructure, the participation barrier is reduced to filling a templated interface and providing test example. By making the interface as flexible as possible, the contributions range from filters for subpopulations with specific protected attributes to transformations via neural style transfer. Through this wide range, we hope that researchers can apply a wider range of augmentation and evaluations strategies to their data and models.

6 Conclusion

In this paper, we introduced NL-Augmenter, a framework for text transformations and filters with the goal of assisting in robustness testing and data augmentation tasks. We demonstrated that through an open participation strategy, NL-Augmenter can cover a substantially wider set of languages, tasks, transformations, and filters than existing work, without a loss of focus. Our repository provides >117 transformations and >23 filters that have been documented and tested. We used these transformations to conduct robustness evaluations of popular transformer-based models and found that they are not robust, even to randomly (i.e.,

non-adversarially) sampled perturbations. Although our analyses have revealed some aspects in which NL-Augmenter can be improved, we showed how it can be beneficial to efforts in evaluating the robustness of NLP models. NL-Augmenter can serve as a crucial resource for data augmentation especially for low-resource domains and task-specific language processing. We welcome future contributions to improve its coverage of the augmentation space and to address its current shortcomings. Investigating the effect on model robustness with larger-scale experiments is a potential direction for future work.

7 Organization

NL-Augmenter is an effort organized by researchers and developers ranging across different niches of NLP. To acknowledge everyones contributions, we list the contribution statements below for all.

Steering Committee: Kaustubh Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahmood, Simon Mille, Jascha Sohl-Dickstein, Ashish Shrivastava, Samson Tan, Tongshuang Wu and Abinaya Mahendiran make up the steering committee. Jinho Choi, Eduard Hovy & Sebastian Ruder provided guidance and feedback. Kaustubh Dhole coordinates and leads the NL-Augmenter effort. All others provide feedback and discuss larger decisions regarding the direction of NL-Augmenter and act as organizers and reviewers.

Repository: Kaustubh, Aadesh, Zhenhao, Tongshuang, Ashish, Saad, Varun & Abinaya created the interfaces and the base repository NL-Augmenter for participants to contribute. This was also a continuation of the repository developed for creating challenge sets (Mille et al., 2021) for GEM (Gehrmann et al., 2021). All the other authors expanded this repository with their implementations.

Reviewers: Kaustubh, Simon, Zhenhao, Sebastian, Varun, Samson, Abinaya, Saad, Tongshuang, Aadesh, Ondrej were involved in reviewing the submissions of participants of the first phase. In the 2nd phase, all other authors performed a cross-review, in which participants were paired with 3 other partcipants. This was followed by a meta review by the organizers.

Robustness Evaluation: Ashish, Tongshuang, Kaustubh & Zhenhao created the evaluation engine. Simon, Kaustubh, Saad, Abinaya & Tongshuang performed the robustness analysis.

Website: Aadesh and Sebastian created the webpages for the project.

The abstract has been written in English, Spanish, Hindi, Chinese, Persian, Quechua, and Indonesian.

References

2006. Respectful Disability Language: Heres Whats Up! https://www.aucd.org/docs/add/sa_summits/Language%20Doc.pdf.

Bamman, David. 2017. Natural language processing for the long tail. In *DH*.

Berard, Alexandre, Ioan Calapodescu, and Claude Roux. 2019. Naver labs europe's systems for the wmt19 machine translation robustness task. *arXiv preprint arXiv:1907.06488*.

Bhagat, Rahul and Eduard Hovy. 2013. What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Bhatt, Abhinav and Kaustubh D. Dhole. 2020. Benchmarking biorelex for entity tagging and relation extraction.

Bird, Steven. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

Board, Smorga's. 2021. Frequently misspelled word list for dyslexia.

Bonial, Claire, Jena Hwang, Julia Bonn, Kathryn Conger, Olga Babko-Malaya, and Martha Palmer. 2012. English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*, 48.

Cahyawijaya, Samuel, Holy Lovenia, Alham Fikri Aji, Genta Indra Winata, Bryan Wilie, Rahmad Mahendra, Christian Wibisono, Ade Romadhony, Karissa Vincentio, Fajri Koto, Jennifer Santoso, David Moeljadi, Cahya Wirawan, Frederikus Hudi, Ivan Halim Parmonangan, Ika Alfina, Muhammad Satrio Wicaksono, Ilham Firdausi Putra, Samsul Rahmadani, Yulianti Oenang, Ali Akbar Septiandri, James Jaya, Kaustubh D. Dhole, Arie Ardiyanti Suryani, Rifki Afina Putri, Dan Su, Keith Stevens, Made Nindyatama Nityasya, Muhammad Farid Adilazuarda, Ryan Ignatius, Ryandito Diandaru, Tiezheng Yu, Vito Ghifari, Wenliang Dai, Yan Xu, Dyah Damapuspita, Cuk Tho, Ichwanul Muslim Karo Karo, Tirana Noor Fatyanosa, Ziwei Ji, Pascale Fung, Graham Neubig, Timothy Baldwin, Sebastian Ruder, Herry Sujaini, Sakriani Sakti, and Ayu Purwarianti.

2022. Nusacrowd: Open source initiative for indonesian nlp resources.

Chen, Jiaao, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2021. An empirical survey of data augmentation for limited data learning in nlp.

Dai, Xiang and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3861–3867, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Damodaran, Prithiviraj. Styleformer.

Deorowicz, Sebastian and Marcin G Ciura. 2005. Correcting spelling errors by modelling their causes. *International journal of applied mathematics and computer science*, 15:275–285.

Dhole, Kaustubh D. 2020. Resolving intent ambiguities by retrieving discriminative clarifying questions.

Dinan, Emily, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. 2020. Queens are powerful too: Mitigating gender bias in dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8173–8188, Online. Association for Computational Linguistics.

Dolan, William B and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Dopierre, Thomas, Christophe Gravier, and Wilfried Logerais. 2021. Protaugment: Unsupervised diverse short-texts paraphrasing for intent detection meta-learning. *CoRR*, abs/2105.12995.

Eger, Steffen and Yannik Benz. 2020. From hero to zéroe: A benchmark of low-level adversarial attacks.

Eger, Steffen, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019a. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Eger, Steffen, Gözde Gül Sahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019b. Text processing like humans do: Visually attacking and shielding NLP systems. *CoRR*, abs/1903.11508.

Eisner, Ben, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 48–54, Austin, TX, USA. Association for Computational Linguistics.

Fadaee, Marzieh, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Fan, Angela, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, et al. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Fan, Angela, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Çelebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. *ArXiv*, abs/2010.11125.

Feng, Steven Y, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Fort, Karën, Gilles Adda, and K. Bretonnel Cohen. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, 37(2):413–420.

Galton, Francis. 1907. Vox populi (the wisdom of crowds). *Nature*, 75(7):450–451.

Gangal, Varun, Steven Y Feng, Eduard Hovy, and Teruko Mitamura. 2021. Nareor: The narrative reordering problem. *arXiv preprint arXiv:2104.06669*.

Gardner, Matt, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating models local decision boundaries via contrast sets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1307–1323.

Gauthier, Jon, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. 2020. Syntaxgym: An online platform for targeted evaluation of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 70–76.

Gehrmann, Sebastian, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Anuoluwapo Aremu, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Chinenye Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Andre Niyongabo Rubungo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. The GEM benchmark: Natural language generation, its evaluation and metrics. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 96–120, Online. Association for Computational Linguistics.

Gehrmann, Sebastian, Abhik Bhattacharjee, Abinaya Mahendiran, Alex Wang, Alexandros Papangelis, Aman Madaan, Angelina McMillan-Major, Anna Shvets, Ashish Upadhyay, Bingsheng Yao, Bryan Wilie, Chandra Bhagavatula, Chaobin You, Craig Thomson, Cristina Garbacea, Dakuo Wang, Daniel Deutsch, Deyi Xiong, Di Jin, Dimitra Gkatzia, Dragomir Radev, Elizabeth Clark, Esin Durmus, Faisal Ladhak, Filip Ginter, Genta Indra Winata, Hendrik Strobelt, Hiroaki Hayashi, Jekaterina Novikova, Jenna Kanerva, Jenny Chim, Jiawei Zhou, Jordan Clive, Joshua Maynez, João Sedoc, Juraj Juraska, Kaustubh Dhole, Khyathi Raghavi Chandu, Laura Perez-Beltrachini, Leonardo F. R. Ribeiro, Lewis Tunstall, Li Zhang, Mahima Pushkarna, Mathias Creutz, Michael White, Mihir Sanjay Kale, Moussa Kamal Eddine, Nico Daheim, Nishant Subramani, Ondrej Dusek, Paul Pu Liang, Pawan Sasanka Ammanamanchi, Qi Zhu, Ratish Puduppully, Reno Kriz, Rifat Shahriyar, Ronald Cardenas, Saad Mahamood, Salomey Osei, Samuel Cahyawijaya, Sanja tajner, Sebastien Montella, Shailza, Shailza Jolly, Simon Mille, Tahmid Hasan, Tianhao Shen, Tosin Adewumi, Vikas Raunak, Vipul Raheja, Vitaly Nikolaev, Vivian Tsai, Yacine Jernite, Ying Xu, Yisi Sang, Yixin Liu, and Yufang Hou. 2022. Gemv2: Multilingual nlg benchmarking in a single line of code.

Gildea, Daniel and Martha Stone Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 239–246. ACL.

Goel, Karan, Nazneen Rajani, Jesse Vig, Samson Tan, Jason Wu, Stephan Zheng, Caiming Xiong annd Mohit Bansal, and Christopher Ré. 2021. Robustness Gym: Unifying the NLP evaluation landscape. *arXiv preprint arXiv:2101.04840*.

Goldberg, Yoav. 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309.

Goyal, Tanya and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252, Online. Association for Computational Linguistics.

Guntuku, Sharath Chandra, Mingyang Li, Louis Tay, and Lyle H Ungar. 2019. Studying cultural differences in emoji usage across the east and the west. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 13, pages 226–235.

Gupta, Aadesh, Kaustubh D. Dhole, Rahul Tarway, Swetha Prabhakar, and Ashish Shrivastava. 2021. Candle: Decomposing conditional and conjunctive queries for task-oriented dialogue systems.

Harel-Canada, Fabrice. 2021. Sibyl.

Hendrickx, Iris, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2013. Semeval-2013 task 4: Free paraphrases of noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 138–143.

hyperreality@GitHub. American british english translator. https://github.com/hyperreality/American-British-English-Translator.

Jalalzai, Hamid, Pierre Colombo, Chloé Clavel, Eric Gaussier, Giovanna Varni, Emmanuel Vignon, and Anne Sabourin. 2020. Heavy-tailed representations,

text polarity classification & data augmentation. In *Advances in Neural Information Processing Systems*, volume 33, pages 4295–4307. Curran Associates, Inc.

Jia, Robin and Percy Liang. 2017a. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031. Association for Computational Linguistics.

Jia, Robin and Percy Liang. 2017b. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Jindal, Ishan, Ranit Aharonov, Siddhartha Brahma, Huaiyu Zhu, and Yunyao Li. 2020. Improved semantic role labeling using parameterized neighborhood memory adaptation. *arXiv preprint arXiv:2011.14459*.

Kaushik, Divyansh, Eduard Hovy, and Zachary C Lipton. 2019. Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*.

Khachatrian, Hrant, Lilit Nersisyan, Karen Hambardzumyan, Tigran Galstyan, Anna Hakobyan, Arsen Arakelyan, A. Rzhetsky, and A. G. Galstyan. 2019. Biorelex 1.0: Biological relation extraction benchmark. In *BioNLP@ACL*.

Kiela, Douwe, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.

Kingsbury, Paul R. and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002, May 29-31, 2002, Las Palmas, Canary Islands, Spain*. European Language Resources Association.

Kočiskỳ, Tomáš, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Kovatchev, Venelin, Phillip Smith, Mark Lee, and Rory Devine. 2021. Can vectors read minds better than experts? comparing data augmentation strategies for the automated scoring of children's mindreading ability. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1196–1206, Online. Association for Computational Linguistics.

Krishna, Kalpesh, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762, Online. Association for Computational Linguistics.

Kumar, Ashutosh, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3609–3619, Minneapolis, Minnesota. Association for Computational Linguistics.

Lample, Guillaume, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Laserna, Charlyn M, Yi-Tai Seih, and James W Pennebaker. 2014. Um... who like says you know: Filler word use as a function of age, gender, and personality. *Journal of Language and Social Psychology*, 33(3):328–338.

Lauer, Mark. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds*. Ph.D. thesis.

Lee, Kenton, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for

natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Lhoest, Quentin, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario ako, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. 2021a. Datasets: A community library for natural language processing.

Lhoest, Quentin, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Mario ako, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Victor Sanh, Teven Le Scao, Kevin Canwen Xu, Nicolas Patry, Steven Liu, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Nathan Raw, Sylvain Lesage, Anton Lozhkov, Matthew Carrigan, Théo Matussière, Leandro von Werra, Lysandre Debut, Stas Bekman, and Clément Delangue. 2021b. huggingface/datasets: 1.14.0.

Li, Dianqi, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Li, Dianqi, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020b. Contextualized perturbation for textual adversarial attack. *CoRR*, abs/2009.07502.

Li, Zhenhao and Lucia Specia. 2019. Improving neural machine translation robustness via data augmentation: Beyond back-translation. *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*.

Lin, Bill Yuchen, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2019. Commongen: A constrained text generation challenge for generative commonsense reasoning. *arXiv preprint arXiv:1911.03705*.

Liu, Zihan, Genta Indra Winata, and Pascale Fung. 2021. Continual mixed-language pre-training for extremely low-resource neural machine translation. *arXiv preprint arXiv:2105.03953*.

Liu, Zihan, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. 2020. Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8433–8440.

Logeswaran, Lajanugen, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5108–5118.

Lu, Kaiji, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2019. Gender bias in neural natural language processing.

Ma, Edward. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Maas, Andrew, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marivate, Vukosi and Tshephisho Sefara. 2020. Improving short text classification through global augmentation methods. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 385–399. Springer.

McCoy, Tom, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Merriam-Webster. What is a diacritic, anyway?

Mille, Simon, Kaustubh D. Dhole, Saad Mahamood, Laura Perez-Beltrachini, Varun Gangal, Mihir Kale, Emiel van Miltenburg, and Sebastian Gehrmann. 2021. Automatic construction of evaluation suites for natural language generation datasets.

Miller, George A. 1998. *WordNet: An electronic lexical database*. MIT press.

Mishra, Shubhanshu, Sijun He, and Luca Belli. 2020. Assessing demographic bias in named entity recognition. *CoRR*, abs/2008.03415.

Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*, pages 220–229. ACM.

Morris, John, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Namysl, Marcin, Sven Behnke, and Joachim Köhler. 2020. NAT: Noise-aware training for robust neural sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1501–1517, Online. Association for Computational Linguistics.

Namysl, Marcin, Sven Behnke, and Joachim Köhler. 2021. Empirical error modeling improves robustness of noisy neural sequence labeling. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 314–329, Online. Association for Computational Linguistics.

Nekoto, Wilhelmina, Vukosi Marivate, Tshinondiwa Matsila, Timi Fasubaa, Taiwo Fagbohungbe, Solomon Oluwole Akinola, Shamsuddeen Muhammad, Salomon Kabongo Kabenamualu, Salomey Osei, Freshia Sackey, Rubungo Andre Niyongabo, Ricky Macharm, Perez Ogayo, Orevaoghene Ahia, Musie Meressa Berhe, Mofetoluwa Adeyemi, Masabata Mokgesi-Selinga, Lawrence Okegbemi, Laura Martinus, Kolawole Tajudeen, Kevin Degila, Kelechi Ogueji, Kathleen Siminyu, Julia Kreutzer, Jason Webster, Jamiil Toure Ali, Jade Abbott, Iroro Orife, Ignatius Ezeani, Idris Abdulkadir Dangana, Herman Kamper, Hady Elsahar, Goodness Duru, Ghollah Kioko, Murhabazi Espoir, Elan van Biljon, Daniel Whitenack, Christopher Onyefuluchi, Chris Chinenye Emezue, Bonaventure F. P. Dossou, Blessing Sibanda, Blessing Bassey, Ayodele Olabiyi, Arshath Ramkilowan, Alp Öktem, Adewale Akinfaderin, and Abdallah Bashir. 2020. Participatory research for low-resourced machine translation: A case study in African languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2144–2160, Online. Association for Computational Linguistics.

Nguyen, Toan Q, Kenton Murray, and David Chiang. 2021. Data augmentation by concatenation for low-resource translation: A mystery and a solution. In *Proceedings of the International Workshop on Spoken Language Translation*, Online. Association for Computational Linguistics.

Pais, Vasile Florian. 2019. *Contributions to semantic processing of texts; Identification of entities and relations between textual units; Case study on Romanian language*. Ph.D. thesis.

Palmer, Martha, Paul R. Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguistics*, 31(1):71–106.

Parikh, Soham, Ananya B. Sai, Preksha Nema, and Mitesh M. Khapra. 2019. Eliminet: A model for eliminating options for reading comprehension with multiple choice questions. *CoRR*, abs/1904.02651.

Park, Kyubyong and Seanie Lee. 2020. g2pm: A neural grapheme-to-phoneme conversion package for mandarin chinese based on a new open benchmark dataset. *CoRR*, abs/2004.03136.

Pierse, Charles. 2021. Transformers Interpret.

Piktus, Aleksandra, Necati Bora Edizel, Piotr Bojanowski, Edouard Grave, Rui Ferreira, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3226–3234, Minneapolis, Minnesota. Association for Computational Linguistics.

Pitler, Emily, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters*, pages 87–90, Manchester, UK. Coling 2008 Organizing Committee.

Ponkiya, Girishkumar, Rudra Murthy, Pushpak Bhattacharyya, and Girish Palshikar. 2020. Looking inside noun compounds: Unsupervised prepositional and free paraphrasing using language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4313–4323.

Ponkiya, Girishkumar, Kevin Patel, Pushpak Bhattacharyya, and Girish Palshikar. 2018. Treat us like the sequences we are: Prepositional paraphrasing of noun compounds using lstm. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1827–1836.

Prasad, Rashmi, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Pruksachatkun, Yada, Satyapriya Krishna, Jwala Dhamala, Rahul Gupta, and Kai-Wei Chang. 2021. Does robustness improve fairness? approaching fairness with word substitution robustness methods for text classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3320–3331, Online. Association for Computational Linguistics.

Qin, Libo, Minheng Ni, Yue Zhang, and Wanxiang Che. 2020. Cosda-ml: Multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3853–3860. International Joint Conferences on Artificial Intelligence Organization. Main track.

Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Raffo, Julio. 2021. WGND 2.0.

Raunak, Vikas, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183, Online. Association for Computational Linguistics.

Ravichander, Abhilasha, Siddharth Dalmia, Maria Ryskina, Florian Metze, Eduard Hovy, and Alan W Black. 2021. NoiseQA: Challenge Set Evaluation for User-Centric Question Answering. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Online.

Regina, Mehdi, Maxime Meyer, and Sébastien Goutal. 2020. Text data augmentation: Towards better detection of spear-phishing emails. *CoRR*, abs/2007.02033.

Ribeiro, Marco Tulio, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Shi, Haoyue, Karen Livescu, and Kevin Gimpel. 2021. Substructure substitution: Structured data augmentation for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3494–3508, Online. Association for Computational Linguistics.

Shi, Peng and Jimmy Lin. 2019a. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Shi, Peng and Jimmy Lin. 2019b. Simple BERT models for relation extraction and semantic role labeling. *CoRR*, abs/1904.05255.

Shrivastava, Ashish, Kaustubh Dhole, Abhinav Bhatt, and Sharvani Raghunath. 2021. Saying No is An Art: Contextualized Fallback Responses for Unanswerable Dialogue Queries. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 87–92, Online. Association for Computational Linguistics.

Shwartz, Vered and Ido Dagan. 2018. Paraphrase to explicate: Revealing implicit noun-compound relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1200–1211.

Si, Chenglei, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2021. Better robustness by more coverage: Adversarial and mixup data augmentation for robust finetuning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1569–1576, Online. Association for Computational Linguistics.

Smith, R. 2007. An overview of the tesseract OCR engine. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil*, pages 629–633. IEEE Computer Society.

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Srivastava, Aarohi, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch,

Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Sugiyama, Amane and Naoki Yoshinaga. 2019. Data augmentation using back-translation for context-aware neural machine translation. In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pages 35–44, Hong Kong, China. Association for Computational Linguistics.

Sun, Tony, Kellie Webster, Apurva Shah, William Yang Wang, and Melvin Johnson. 2021. They, them, theirs: Rewriting with gender-neutral english. *CoRR*, abs/2102.06788.

Tan, Fiona Anting, Devamanyu Hazarika, See-Kiong Ng, Soujanya Poria, and Roger Zimmermann. 2021a. Causal augmentation for causal sentence classification. In *Proceedings of the First Workshop on Causal Inference and NLP*, pages 1–20, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tan, Samson and Shafiq Joty. 2021. Code-mixing on sesame street: Dawn of the adversarial polyglots. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3596–3616, Online. Association for Computational Linguistics.

Tan, Samson, Shafiq Joty, Kathy Baxter, Araz Taeihagh, Gregory A. Bennett, and Min-Yen Kan. 2021b. Reliability testing for natural language processing systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4153–4169, Online. Association for Computational Linguistics.

Tan, Samson, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2920–2935, Online. Association for Computational Linguistics.

Tiedemann, Jörg and Santhosh Thottingal. 2020. OPUS-MT Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

Vijayakumar, Ashwin, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes.

Vijayakumar, Ashwin K., Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *CoRR*, abs/1610.02424.

Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Wang, Xiao, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, Qinzhuo Wu, Zhengyan Li, Chong Zhang, Ruotian Ma, Zichu Fei, Ruijian Cai, Jun Zhao, Xingwu Hu, Zhiheng Yan, Yiding Tan, Yuan Hu, Qiyuan Bian, Zhihua Liu, Shan Qin, Bolin Zhu, Xiaoyu Xing, Jinlan Fu, Yue Zhang, Minlong Peng, Xiaoqing Zheng, Yaqian Zhou, Zhongyu Wei, Xipeng Qiu, and Xuanjing Huang. 2021a. TextFlint: Unified multilingual robustness evaluation toolkit for natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 347–355, Online. Association for Computational Linguistics.

Wang, Yuxuan, Wanxiang Che, Jiang Guo, Yijia Liu, and Ting Liu. 2019b. Cross-lingual bert transformation for zero-shot dependency parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5721–5727.

Wang, Yuxuan, Wanxiang Che, Ivan Titov, Shay B. Cohen, Zhilin Lei, and Ting Liu. 2021b. A closer look into the robustness of neural dependency parsers using better adversarial examples. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2344–2354, Online. Association for Computational Linguistics.

Wei, Jason W. and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 6381–6387. Association for Computational Linguistics.

Wieting, John and Kevin Gimpel. 2017. Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *arXiv preprint arXiv:1711.05732*.

Wieting, John, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proceedings of Empirical Methods in Natural Language Processing*.

Williams, Adina, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Wilson, Steven, Walid Magdy, Barbara McGillivray, Kiran Garimella, and Gareth Tyson. 2020. Urban dictionary embeddings for slang NLP applications. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4764–4773, Marseille, France. European Language Resources Association.

Wiseman, Sam and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wu, Tongshuang, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Xie, Qizhe, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Xu, Liang, Qianqian Dong, Cong Yu, Yin Tian, Weitang Liu, Lu Li, and Xuanwei Zhang. 2020. Cluener2020: Fine-grained name entity recognition for chinese. *arXiv preprint arXiv:2001.04351*.

Yaseen, Usama and Stefan Langer. 2021. Data augmentation for low-resource named entity recognition using backtranslation. *CoRR*, abs/2108.11703.

Yi, Sheng Kung, Mark Steyvers, Michael Lee, and Matthew Dry. 2010. Wisdom of the crowds in minimum spanning tree problems. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32.

Yorke, Alex. butter-fingers. https://github.com/alexyorke/butter-fingers.

Yunfei. Chinese-Names-Corpus . https://github.com/wainshine/Chinese-Names-Corpus.

Zhang, Jing, Bonggun Shin, Jinho D Choi, and Joyce C Ho. 2021. Smat: An attention-based deep learning solution to the automation of schema matching. In *European Conference on Advances in Databases and Information Systems*, pages 260–274. Springer.

Zhang, Wei Emma, Quan Z. Sheng, and Ahoud Abdulrahmn F. Alhazmi. 2019a. Generating textual adversarial examples for deep learning models: A survey. *CoRR*, abs/1901.06796.

Zhang, Yuan, Jason Baldridge, and Luheng He. 2019b. PAWS: paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1298–1308. Association for Computational Linguistics.

Zhao, Zhe, Hui Chen, Jinbin Zhang, Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju, and Xiaoyong Du. 2019. Uer: An open-source toolkit for pre-training models. *EMNLP-IJCNLP 2019*, page 241.

A Transformations

The following is the list of all accepted transformations to NL-Augmenter project. Many of the transformations tokenize the sentences using SpaCy[4] or NLTK (Bird, 2006) tokenizers. We discuss the implementations of each alongwith their limitations. The title of each transformation subsection is clickable and redirects to the actual python implementation. Many of the transformations use external libraries and we urge readers to look at each implementation and its corresponding 'requirements.txt' files.

[4] https://spacy.io/

A.1 Abbreviation Transformation

This transformation replaces a word or phrase with its abbreviated counterpart "homework" -> "hwk" using a web-scraped slang dictionary.[5]

You → yu driving at 80 miles per hour → mph is why insurance is → tis so freaking → friggin expensive.

A.2 Add Hash-Tags

This transformation uses words in the text to generate hashtags. These hastags are then appeneded to the original text. Using the same words appearing in the sentence to generate the hashtags acts as redundant noise that models should learn to ignore. Hashtags are widespread in social media channels and are used to draw attention to the source text and also as a quick stylistic device.

I love domino's pizza. → #LovePizza #Love #I #Pizza

A.3 Adjectives Antonyms Switch

This transformation switches English adjectives in a sentence with their WordNet (Miller, 1998) antonyms to generate new sentences with possibly different meanings and can be useful for tasks like Paraphrase Detection, Paraphrase Generation, Semantic Similarity, and Recognizing Textual Entailment.

Amanda's mother was very beautiful → ugly .

A.4 AmericanizeBritishizeEnglish

This transformation takes a sentence and tries to convert it from British English to American English and vice-versa. A select set of words have been taken from hyperreality@GitHub.

I love the pastel colours → colors

A.5 AntonymsSubstitute

This transformation introduces semantic diversity by replacing an even number of adjective/adverb antonyms in a given text. We assume that an even number of antonyms transforms will revert back sentence semantics; however, an odd number of transforms will revert the semantics. Thus, our transform only applies to the sentence that has an even number of revertible adjectives or adverbs. We called this mechanism double negation.

Steve is able → unable to recommend movies that depicts the lives of beautiful → ugly minds.

[5] Scraped from https://www.noslang.com/dictionary

A.6 Auxiliary Negation Removal

This is a low-coverage transformation which targets sentences that contain negations. It removes negations in English auxillaries and attempts to generate new sentences with the oppposite meaning.

Ujjal Dev Dosanjh was not → Ujjal Dev Dosanjh was the 1st Premier of British Columbia from 1871 to 1872.

A.7 AzertyQwertyCharsSwap

Preferably use the above download link, as the release tarballs are generated deterministically → qre generqted deterministicqlly whereas GitHub's are not.

A.8 BackTranslation

This transformation translates a given English sentence into German and back to English.This transformation acts like a light paraphraser. Multiple variations can be easily created via changing parameters like the language as well as the translation models which are available in plenty. Backtranslation has been quite popular now and has been a quick way to augment examples (Li and Specia, 2019; Sugiyama and Yoshinaga, 2019).

Andrew finally returned → eventually gave Chris the French book the French book I bought last week.

A.9 BackTranslation for Named Entity Recognition

This transformation splits the token sequences into segments of entity mention(s) and "contexts" around the entity mention(s). Backtranslation is used to paraphrase the contexts around the entity mention(s), thus resulting in a different surface form from the original token sequence. The resultant tokens are also assigned new tags. Exploiting this transformation has shown to empirically benefit named entity tagging (Yaseen and Langer, 2021) and hence could arguably benefit other low-resource tagging tasks (Bhatt and Dhole, 2020; Khachatrian et al., 2019; Gupta et al., 2021).

A.10 Butter Fingers Perturbation

This perturbation adds noise to all types of text sources (sentence, paragraph, etc.) proportional to noise erupting from keyboard typos making common spelling errors. Few letters picked at random are replaced with letters which are at keyboard positions near the source letter. The implementation has been borrowed from here (Yorke) as used in (Mille et al., 2021). There has

also been some recent work in NoiseQA (Ravichander et al., 2021) to mimick keyboard typos.

Sentences → Senhences with gapping, such as Paul likes coffee → coffwe and Mary tea, lack an overt predicate to indicate → indicatx the relation → relauion between two or more arguments → argumentd .

A.11 Butter Fingers Perturbation For Indian Languages

This implements the butter fingers perturbation as used above for 7 Indian languages: Bangla, Gujarati, Hindi, Kannada, Malayalam, Oriya, Punjabi, Tamil, and Telugu. The implementation considers the InScript keyboard [6] which is decreed as a standard for Indian scripts.

A.12 Change Character Case

This transformation acts like a perturbation and randomly swaps the casing of some of the letters. The transformation's outputs will not work with uncased models or languages without casing.

Alice in Wonderland is a 2010 American live- action → actIon / animated → anImated dark fantasy → faNtasy adventure film.

A.13 Change Date Format

This transformation changes the format of dates.

The first known case of COVID-19 was identified in Wuhan, China in December → Dec 2019.

A.14 Change Person Named Entities

This perturbation changes the name of the person from one name to another by making use of the lexicon of person names in Ribeiro et al. (2020).

Andrew → Nathaniel finally returned the French book to Chris that I bought last week

A.15 Change Two Way Named Entities

This perturbation also changes the name of the person but also makes a parallel change in the label or reference text with the same name making it useful for text-to-text generation tasks.

He finally returned the French book to Chris → Austin that I bought last week

[6] https://en.wikipedia.org/wiki/InScript_keyboard

A.16 Chinese Antonym and Synonym Substitution

This transformation substitutes Chinese words with their synonyms or antonyms by using the Chinese dictionary[7] and NLP Chinese Data Augmentation dictionary[8].

A.17 Chinese Pinyin Butter Fingers Perturbation

This transformation implements the Butter Fingers Perturbation for Chinese characters. Few Chinese words and characters that are picked at random will be substituted with others that have similar pinyin (based on the default Pinyin keyboards in Windows and Mac OS). It uses a database of 16142 Chinese characters [9] and its associated pinyins to generate the perturbations for Chinese characters. A smaller database of 3500 [10] more frequently seen Chinese characters are also used in the perturbations with a higher probability of being used compared to less frequently seen Chinese characters. It also uses a database of 575173 words [11] that are combined from several sources [12] in order to generate perturbations for Chinese words.

A.18 Chinese Person Named Entities and Gender Perturbation

This perturbation adds noise to all types of text sources containing Chinese names (sentence, paragraph, etc.) by swapping a Chinese name with another Chinese name whilst also allowing the possibility of gender swap. CLUENER (Xu et al., 2020; Zhao et al., 2019) is used for tagging named entities in Chinese. The list of names is taken from the Chinese Names Corpus! (Yunfei). It can provide assistance in detecting biases present in language models and the ability to infer implicit gender information when presented with gender-specific names. This can also be useful in mitigating representation biases in the input text.

A.19 Chinese (Simplified & Traditional) Perturbation

This perturbation adds noise to all types of text sources containing Chinese words and characters (sentence,

[7] Chinese Dictionary: https://github.com/guotong1988/chinese_dictionary

[8] NLP Chinese Data Augmentation: https://github.com/425776024/nlpcda

[9] https://github.com/pwxcoo/chinese-xinhua

[10] https://github.com/elephantnose/characters

[11] http://thuocl.thunlp.org/

[12] https://github.com/fighting41love/Chinese_from_dongxiexidian

paragraph, etc.) by changing the words and characters between Simplified and Traditional Chinese as well as other variants of Chinese Characters such as Japanese Kanji, character-level and phrase-level conversion, character variant conversion and regional idioms among Mainland China, Taiwan and Hong Kong, all available as configurations originally in the OpenChineseConvert project [13].

A.20 City Names Transformation

This transformation replaces instances of populous and well-known cities in Spanish and English sentences with instances of less populous and less well-known cities to help reveal demographic biases (Mishra et al., 2020) prevelant in named entity recognition models. The choice of cities have been taken from the World Cities Dataset [14].

```
The team was established in Dallas → Viera West in 1898 and was a charter member of the NFL in 1920.
```

A.21 Close Homophones Swap

Humans are generally guided by their senses and are unconsciously robust against phonetic attacks. Such types of attacks are highly popular in languages like English which has an irregular mapping between pronunciation and spelling (Eger and Benz, 2020). This transformation mimics writing behaviors where users swap words with similar homophones either intentionally or by accident. This transformation acts like a perturbation to test robustness. Few words picked at random are replaced with words with similar homophones which sound similar or look similar. Some of the word choices might not be completely natural to normal human behavior, since humans "prefer" some words over others even they sound exactly the same. So it might not be fully reflecting the natural distribution of intentional or unintentional swapping of words.

```
Sentences with gapping, such as Paul likes coffee and Mary tea → Tee , lack an overt predicate to indicate the → Thee relation between two or more → Morr arguments.
```

A.22 Color Transformation

This transformation augments the input sentence by randomly replacing mentioned colors with different ones from the 147 extended color keywords specified by the World Wide Web Consortium (W3C) [15]. Some of the colors include “dark sea green”, “misty rose”, “burly wood”.

```
Tom bought 3 apples, 1 orange → misty rose , and 4 bananas and paid $10.
```

A.23 Concatenate Two Random Sentences (Bilingual)

Given a dataset, this transformation concatenates a sentence with a previously occuring sentence as explained in (Nguyen et al., 2021). A monolingual version is mentioned in the subsequent subsection below. This concatenation would benefit all text tasks that use a transformer (and likely other sequence-to-sequence architectures). Previously published work (Nguyen et al., 2021) has shown a large gain in performance of low-resource machine translation using this method. In particular, the learned model is stronger due to being able to see training data that has context diversity, length diversity, and (to a lesser extent) position shifting.

A.24 Concatenate Two Random Sentences (Monolingual)

This is the monolingual counterpart of the above.

```
I am just generating a very very very long sentence to make sure that the method is able to handle it. It does not even need to be a sentence. Right? This is not splitting on punctuation... I am just generating a very very very long sentence to make sure that the method is able to handle it. It does not even need to be a sentence. Right? This is not splitting on punctuation...
```

A.25 Concept2Sentence

This transformation intakes a sentence, its associated integer label, and (optionally) a dataset name that is supported by huggingface/datasets (Lhoest et al., 2021a,b). It works by extracting keyword concepts from the original sentence, passing them into a BART (Lewis et al., 2020) transformer trained on CommonGen (Lin et al., 2019) to generate a new, related sentence which reflects the extracted concepts. Providing a dataset allows the function to use transformers-interpret (Pierse, 2021) to identify the most critical concepts for use in the generative step. Underneath the hood, this transform makes use of the Sibyl tool (Harel-Canada, 2021), which is capable of also transforming the label as well. However, this particular implementation of C2S generates new text that is invariant (INV) with respect to the label. Since the model is trained on CommonGen, which is focussed on image captioning, the style of the

[13] https://github.com/BYVoid/OpenCC
[14] https://www.kaggle.com/juanmah/world-cities
[15] https://www.w3.org/TR/2021/REC-css-color-3-20210805/

output sentence would be geared towards scenic descritions and might not necessarily adhere to the syntax of the original sentence. Besides, it can be hard to argue that a handful subset of keywords could provide a complete description of the original sentence.

A.26 Contextual Meaning Perturbation

This transformation was designed to model the "Chinese Whispers" or "Telephone" children's game: The transformed sentence appears fluent and somewhat logical, but the meaning of the original sentence might not be preserved. To achieve logical coherence, a pretrained language model is used to replace words with alternatives that match the context of the sentence. Grammar mistakes are reduced by limiting the type of words considered for changes (based on POS tagging) and replacing adjectives with adjectives, nouns with nouns, etc. where possible.

This transformation benefits users who seek perturbations that preserve fluency but not the meaning of the sentence. For instance, it can be used in scenarios where the meaning is relevant to the task, but the model shows a tendency to over-rely on simpler features such as the grammatical correctness and general coherence of the sentence. A real-world example would be the training of quality estimation models for machine translation (does the translation maintain the meaning of the source?) or for text summarisation (does the summary capture the content of the source?).

Word substitution with pre-trained language models has been explored in different settings. For example, the augmentation library nlpaug (Ma, 2019) and the adversarial attack library TextAttack (Morris et al., 2020) include contextual perturbation methods. However, their implementations do not offer control over the type of words that should be perturbed and introduce a large number of grammar mistakes. If the aim is to change the sentence's meaning while preserving its fluency, this transformation can help to get the same effect with significantly fewer grammatical errors. Li et al. (2020a) propose an alternative approach to achieve a similar objective.

A.27 Contractions and Expansions Perturbation

This perturbation substitutes the text with popular expansions and contractions, e.g., "I'm" is changed to "I am"and vice versa. The list of commonly used contractions & expansions and the implementation of perturbation has been taken from Checklist (Ribeiro et al., 2020).

`He often does n't → not come to school.`

A.28 Correct Common Misspellings

This transformation acts like a lightweight spellchecker and corrects common misspellings appearing in text by looking for words in Wikipedia's Lists of Common Misspellings.

`Andrew andd → and Alice finally returnd → returned the French book that I bought lastr → last week`

A.29 Country/State Abbreviation

This transformation replaces country and state names with their common abbreviations[16]. Abbreviations can be common across different locations: "MH" can refer to Country Meath in Ireland as well as the state of Maharashtra in India and hence this transformation might result in a slight loss of information, especially if the surrounding context doesn't have enough signals.

`One health officer and one epidemiologist have boarded the ship in San Diego, CA → California on April 13, 2015 to conduct an environmental health assessment.`

A.30 Decontextualisation of the main Event

Semantic Role Labelling (SRL) is a powerful shallow semantic representation to determine who did what to whom, when, and where (and why and how etc). The core arguments generally talk about the participants involved in the event. Addtionally, contextual arguments on the other hand provide more specific information about the event. After tagging a sentence with an appropriate semantic role labels using an SRL labeller (Jindal et al., 2020; Shi and Lin, 2019a). This transformation crops out contextual arguments to create a new sentence with a minimal description of the event. Helping to generate textual pairs for entailment.

A.31 Diacritic Removal

"Diacritics are marks placed above or below (or sometimes next to) a letter in a word to indicate a particular pronunciation in regard to accent, tone, or stress as well as meaning, especially when a homograph exists without the marked letter or letters." Merriam-Webster. This transformation removes these diacritics or accented characters, and replaces them with their non-accented versions. It can be common for non-native or inexperienced speakers to miss out on any accents and specify non-accented versions.

[16] Countries States Cities Database: https://github.com/dr5hn/countries-states-cities-database

She lookèd → looked east an she lookèd → looked west.

A.32 Disability/Differently Abled Transformation

Disrespectful language can make people feel excluded and represent an obstacle towards their full participation in the society (Res, 2006). This low-coverage transformation substitutes outdated references to references of disabilities with more appropriate and respectful ones which avoid negative connotations. A small list of inclusive words and phrases have been taken from a public article on inclusive communication, Wikipedia's list of disability-related terms with negative connotations, terms to avoid while writing about disability.

They are deaf → person or people with a hearing disability.

A.33 Discourse Marker Substitution

This perturbation replaces a discourse marker in a sentence by a semantically equivalent marker. Previous work has identified discourse markers that have low ambiguity (Pitler et al., 2008). This transformation uses the corpus analysis on PDTB 2.0 (Prasad et al., 2008) to identify discourse markers that are associated with a discourse relation with a chance of at least 0.5. Then, a marker is replaced with a different marker that is associated to the same semantic class.

It has plunged 13% since → inasmuch as July to around 26 cents a pound. A year ago ethylene sold for 33 cents

A.34 Diverse Paraphrase Generation Using SubModular Optimization and Diverse Beam Search

This transformation generates multiple paraphrases of a sentence by employing 4 candidate selection methods on top of a base set of backtranslation models. 1) DiPS (Kumar et al., 2019) 2) Diverse Beam Search (Vijayakumar et al., 2018) 3) Beam Search (Wiseman and Rush, 2016) 4) Random. Unlike beam search which generally focusses on the top-k candidates, DiPS introduces a novel formulation of using submodular optimisation to focus on generating more diverse paraphrases and has been proven to be an effective data augmenter for tasks like intent recognition and paraphrase detection (Kumar et al., 2019). Diverse Beam Search attempts to generate diverse sequences by employing a diversity promoting alternative to the classical beam search (Wiseman and Rush, 2016).

A.35 Dislexia Words Swap

This transformation acts like a perturbation by altering some words of the sentences with abberations (Board, 2021) that are likely to happen in the context of dyslexia.

Biden hails your → you're relationship with Australia just days after new partnership drew ire from France.

A.36 Emoji Icon Transformation

This transformation converts emojis into their equivalent keyboard format (e.g., 🙂-> ":)") and vice versa (e.g., ":)" -> 🙂).

A.37 Emojify

This transformation augments the input sentence by swapping words with emojis of similar meanings. Emojis, introduced in 1997 as a set of pictograms used in digital messaging, have become deeply integrated into our daily communication. More than 10% of tweets[17] and more than 35% of Instagram posts[18] include one or more emojis in 2015. Given the ubiquitousness of emojis, there is a growing body of work researching the linguistic and cultural aspects of emojis (Guntuku et al., 2019) and how we can leverage the use of emojis to help solve NLP tasks (Eisner et al., 2016).

Apple is looking at buying U.K. startup for $132 billion. → 🍎 is 👀 at 🛍️ 🇬🇧 startup for $ 1️⃣ 3️⃣ 2️⃣.

A.38 English Inflectional Variation

This transformation adds inflectional variation to English words and can be used to test the robustness of models against inflectional variations. In English, each inflection generally maps to a Part-Of-Speech tag [19] in the Penn Treebank (Marcus et al., 1993). For each content word in the sentence, it is first lemmatised before randomly sampling a valid POS category and reinflecting the word according to the new category. The sampling process for each word is constrained using its POS tag to maintain the original sense for polysemous words. This has been adapted from the Morpheus (Tan et al., 2020) adversarial attack.

Ujjal Dev Dosanjh served → serve as 33rd Premier → Premiers of British Columbia from 2000 to 2001

[17] https://blog.twitter.com/en_us/a/2015/emoji-usage-in-tv-conversation
[18] https://instagram-engineering.com/
[19] Penn TreeBank POS

A.39 English Mention Replacement for NER

This transformation randomly swaps an entity mention with another entity mention of the same entity type. Exploiting this transformation as a data augmentation strategy has been empirically shown to improve the performance of underlying (NER) models (Dai and Adel, 2020).

A.40 Filler Word Augmentation

This augmentation adds noise in the form of colloquial filler phrases. 23 different phrases are chosen across 3 different categories: general filler words and phrases ("uhm", "err", "actually", "like", "you know"...), phrases emphasizing speaker opinion/mental state ("I think/believe/mean", "I would say"...) & phrases indicating uncertainty ("maybe", "perhaps", "probably", "possibly", "most likely").The latter two categories had shown promising results Kovatchev et al. (2021) when they were concatenated at the beginning of the sentence unlike this implementation which perform insertions at any random positions. Filler words are based on the work of Laserna et al. (2014) but have not been explored in the context of data augmentation.

A.41 Style Transfer from Informal to Formal

This transformation transfers the style of text from formal to informal and vice versa. It uses the implementation of Styleformer (Damodaran).

`What you upto → currently doing ?`

A.42 French Conjugation Substitution

This transformation change the conjugation of verbs for simple french sentences with a specified tense. It detects the pronouns used in the sentence in order to conjugate accordingly whenever a sentence contains differents verbs. This version only works for indicative tenses. It also only works for simple direct sentences (subject, verb, COD/COI), which contains a pronoun as subject (il, elle, je etc.). It does not detect when the subject is a couple of nouns ("les enfants" or "la jeune femme").

A.43 Gender And Culture Diversity Name Changer (1-way and 2-way)

Corpora exhibits many representational biases and this transformation focuses on one particular mediator, the personal names. It diversifies names in the corpora along two critical dimensions, gender and cultural background. Technically, the transformation samples a (country, gender) pair and then randomly draws a name from that (country, gender) pair to replace the original name. We collected 42812 distinct names from 141 countries.They are primarily from the World Gender Name Dictionary (Raffo, 2021).

Common name augmentations do not consider their gender and cultural implication. Thus, they do not necessarily mitigate biases or promote the minority's representation because the augmented name may be from the same gender and cultural background. This is the case, for example in the CheckList's (Ribeiro et al., 2020) implemented name augmentation. Taking the interaction of the names therein with ours, 34.0%, 33.5%, 31.9%, 30.8% of them are popular names in US, Canada, Australia, and UK, respectively. Only 0.4%, 0.4%, 0.5%, 2.1% of them are from India, Korea, China, and Kazakhstan.

`Rachel → Charity Green, a sheltered but friendly woman, flees her wedding day and wealthy yet unfulfilling life.`

A.44 Neopronoun Substitution

This transformation performs grammatically correct substitution from English to English of the gendered pronouns, he/she, in a given sentence with their neopronoun counterparts, based on a list compiled by UNC Greensboro and LGBTA WIKI[20]. NLP models, such as those for neural machine translation, often fail to recognize the neopronouns and treat them as proper nouns. This transformation seeks to render the training data used in NLP pipelines more neopronoun aware to reduce the risk of trans-erasure. The reason why a simple look-up-table approach might not work is due to the fact that the case may differ depending on the context.

`She → They had her → their friends tell her → them about the event.`

A.45 Gender Neutral Rewrite

This transformation involves rewriting an English sentence containing a single gendered entity with its gender-neutral variant. One application is machine translation, when translating from a language with gender-neutral pronouns (e.g. Turkish) to a language with gendered pronouns (e.g. English). This transformation is based on the algorithm proposed by Sun et al. (2021).

`His → Their dream is to be a fireman → firefighter when he → they grows → grow up.`

[20] https://intercultural.uncg.edu/wp-content/uploads/Neopronouns-Explained-UNCG-Intercultural-Engagement.pdf

A.46 GenderSwapper

This transformation introduces gender diversity to the given data. If used as data augmentation for training, the transformation might mitigate gender bias, as shown in Dinan et al. (2020). It also might be used to create a gender-balanced evaluation dataset to expose the gender bias of pre-trained models. This transformation performs lexical substitution of the opposite gender. The list of gender pairs (shepherd <–> shepherdess) is taken from Lu et al. (2019). Genderwise names used from Ribeiro et al. (2020) are also randomly swapped.

A.47 GeoNames Transformation

This transformation augments the input sentence with information based on location entities (specifically cities and countries) available in the GeoNames database[21]. E.g., if a country name is found, the name of the country is appended with information about the country like its capital city, its neighbouring countries, its continent, etc. Some initial ideas of this nature were explored in Pais (2019).

A.48 German Gender Swap

This transformation replaces the masculine nouns and pronouns with their female counterparts for German sentences from a total of 2226 common German names.[22]

Er → Sie ist ein Arzt → eine Ärztin und mein Vater → meine Mutter .

A.49 Grapheme to Phoneme Substitution

This transformation adds noise to a sentence by randomly converting words to their phonemes. Grapheme-to-phoneme substitution is useful in NLP systems operating on speech. An example of grapheme to phoneme substitution is "permit" → P ER0 M IH1 T'.

A.50 Greetings and Farewells

This transformation replaces greetings (e.g. "Hi", "Howdy") and farewells (e.g. "See you", "Good night") with their synonymous equivalents.

Hey → Hi everyone. It's nice → Pleased to meet you. How have → are you been ?

A.51 Hashtagify

This transformation modifies an input sentence by identifying named entities and other common words and turning them into hashtags, as often used in social media.

A.52 Insert English and French Abbreviations

This perturbation replaces in texts some well known English and French words or expressions with (one of) their abbreviations. Many of the abbreviations covered here are quite common on social medias platforms, even though some of them are quite generic. This implementation is partly inspired by recent work in Machine Translation (Berard et al., 2019).

A.53 Leet Transformation

Visual perturbations are often used to disguise offensive comments on social media (e.g., !d10t) or as a distinct writing style (1337 in leet speak) (Eger et al., 2019a), especially common in scenarios like video gaming. Humans are unconsciously robust to such visually similar texts. This perturbation replaces letters with their visually similar "leet" counterparts.[23]

Ujjal Dev Dosanjh served → U7jal 0ev D0san74 serv3d as 33rd Premier of British Columbia from → Pr33i3r 0f 8ritis4 00lu36ia fr0m 2000 to → t0 2001

A.54 Lexical Counterfactual Generator

This transformation generates counterfactuals by simply substituting negative words like "not", "neither" in one sentence of a semantically similar sentence pair. The substituted sentence is then backtranslated in an attempt to correct for grammaticality. This transformation would be useful for tasks like entailment and paraphrase detection.

A.55 Longer Location for NER

This transformation augments data for Named Entity Recognition (NER) tasks by augmenting examples which have a Location Tag. Names of locations are expanded by appending them with cardinal directions like "south", "N", "northwest", etc. The transformation ensures that the tags of the new sentence are accordingly modified.

A.56 Longer Location Names for testing NER

This transformation augments data for Named Entity Recognition (NER) tasks by augmenting examples that

[21] http://download.geonames.org/export/dump/

[22] https://de.wiktionary.org/wiki/Verzeichnis:Deutsch/Namen

[23] https://simple.wikipedia.org/wiki/Leet

have a Location (LOC) Tag. Names of location are expanded by inserting random prefix or postfix word(s). The transformation also ensures that the labels of the new tags are accordingly modified.

A.57 Longer Names for NER

This transformation augments data for Named Entity Recognition (NER) tasks by augmenting examples which have a Person Tag. Names of people are expanded by inserting random characters as initials. The transformation also ensures that the labels of the new tags are accordingly modified.

A.58 Lost in Translation

This transformation is a generalization of the Back-Translation transformation to any sequence of languages supported by the Helsinki-NLP OpusMT models (Tiedemann and Thottingal, 2020).

`Andrew finally returned → brought Chris back the French book the French book I bought last week I bought last week`

A.59 Mixed Language Perturbation

Mixed language training has been effective for cross-lingual tasks (Liu et al., 2020), to help generate data for low-resource scenarios (Liu et al., 2021) and for multilingual translation (Fan et al., 2021). Two transformations translate randomly picked words in the text from English to other languages (e.g., German). It can be used to test the robustness of a model in a multilingual setting.

`Andrew finally returned the → die Comic book to Chris that I bought last week → woche`

A.60 Mix transliteration

This transformation transliterates randomly picked words from the input sentence (of given source languae script) to a target language script. It can be used to train/test multilingual models to improve/evaluate their ability to understand complete or partially transliterated text.

A.61 MR Value Replacement

This perturbation adds noise to a key-value meaning representation (MR) (and its corresponding sentence) by randomly substituting values/words with their synonyms (or related words). This transformation uses a simple strategy to align values of a MR and tokens in the corresponding sentence inspired by how synonyms are substituted for tasks like machine translation (Fadaee et al., 2017). This way, there could be some problems in complex sentences. Besides, the transformation might introduce non-grammatical segments.

A.62 Multilingual Back Translation

This transformation translates a given sentence from a given language into a pivot language and then back to the original language. This transformation is a simple paraphraser that works on 100 different languages. Back Translation has been quite popular now and has been a quick way to augment (Li and Specia, 2019; Sugiyama and Yoshinaga, 2019; Fan et al., 2020).

`Being honest → Honesty should be one of our most important character traits → characteristics`

A.63 Multilingual Dictionary Based Code Switch

This transformation generates multi-lingual code-switching data to fine-tune encoders of large language models (Qin et al., 2020; Tan and Joty, 2021; Wang et al., 2019b) by making use of bilingual dictionaries of MUSE (Lample et al., 2018).

A.64 Multilingual Lexicon Perturbation

This perturbation helps to creates code-mixed sentences for both high-resource and low-resource languages by randomly translating words with a specified probability from any supported languages (e.g., English) to other supported languages (e.g., Chinese) by using a multilingual lexicon. Thus, it can be used to generate code-mixed training data to improve models for multilingual and cross-lingual settings. As of now 100 languages are supported and 3000 common English words listed on ef.com [24] are supported. The lexicon implementation is also 160x faster than its model based counterpart.

A.65 Causal Negation & Strengthening

This transformation is targeted at augmenting Causal Relations in text and adapts the code from the paper Causal Augmentation for Causal Sentence Classification (Tan et al., 2021a). There are two operations: 1. Causal Negation: Negative words like "not, no, did not" are introduced into sentences to unlink the causal relation. 2. Causal Strengthening: Causal meaning is strengthened by converting weaker modal words

[24] https://www.ef.com/wwen/english-resources/english-vocabulary/top-3000-words/

into stronger ones like "may" to "will" to assert causal strength.

The implementation provides users with the option to amend causal meaning automatically from the root word of the sentence, or by explicitly highlighting the index of the word they wish to amend. Additionally, we include WordNet (Miller, 1998) synonyms and tense matching to allow for more natural augmentations.

```
The rs7044343 polymorphism could be → was involved in regulating the production of IL-33.
```

A.66 Question Rephrasing transformation

This implementation rephrases questions for sentence tasks by using the T5 model used in A.75 for Question Answering tasks.

A.67 English Noun Compound Paraphraser [N+N]

This transformation replaces two-word noun compounds with a paraphrase, based on the compound paraphrase dataset from SemEval 2013 Task 4 (Hendrickx et al., 2013). It currently only works for English. Any two-word compound that appears in a dataset of noun compound paraphrases will be replaced by a paraphrase. If more than one two-word compound appears, then all combinations of compound paraphrases (including no paraphrase at all) will be returned. For example, the paraphrases of "club house" include "house for club activities", "house for club members", "house in which a club meets", etc. We start with replacing paraphrases with the highest score (the specified frequency in the annotated dataset), and paraphrases with the same score (ties) are sorted randomly. This transformation currently only checks for noun compounds from Hendrickx et al. (2013) and therefore has low coverage. To improve it, other datasets could be added, e.g., from Ponkiya et al. (2018) or Lauer (1995). To attain even wider-coverage (at the expense of lower precision), machine learning approaches such as Shwartz and Dagan (2018) or Ponkiya et al. (2020) could be considered. In addition, some of the the paraphrases in Hendrickx et al. (2013) sound a little odd (e.g., "blood cell" -> "cell of blood") and may not fit well in context.

A.68 Number to Word

This transformation acts like a perturbation to improve robustness on processing numerical values. The perturbated sentence contains the same information as the initial sentence but with a different representation of numbers.

A.69 Numeric to Word

This transformation translates numbers in numeric form to their textual representations. This includes general numbers, long numbers, basic math characters, currency, date, time, phone numbers, etc.

A.70 OCR Perturbation

This transformation directly induces Optical Character Recognition (OCR) errors into the input text. It renders the input sentence as an image and recognizes the rendered text using the OCR engine Tesseract 4 (Smith, 2007). It works with text in English, French, Spanish, and German. The implementation follows previous work by Namysl et al. (2021).

A.71 Add Noun Definition

This transformation appends noun definitions onto the original nouns in a sentence. Definitions of nouns are collected from Wikidata [25].

A.72 Pig Latin Cipher

This transformation translates the original text into pig latin. Pig Latin is a well-known deterministic transformation of English words, and can be viewed as a cipher which can be deciphered by a human with relative ease. The resulting sentences are completely unlike examples typically used in language model training. As such, this augmentation change the input into inputs which are difficult for a language model to interpret, while being relatively easy for a human to interpret.

A.73 Pinyin Chinese Character Transcription

This transformation transcribes Chinese characters into their Mandarin pronunciation using the Pinyin romanization scheme. The Character-to-Pinyin converter at the core of this transformation is a neural model by Park and Lee (2020).

A.74 SRL Argument Exchange

This perturbation adds noise to all types of English text sources (sentence, paragraph, etc.) proportional to the number of arguments identified by SRL BERT (Shi and Lin, 2019b). Different rules are applied to deterministically modify the sentence in a meaning-preserving manner. Rules look as follows: *if ARGM-LOC and ARGM-TMP both present, exchange them.*

[25] https://www.wikidata.org/wiki/Wikidata:Main_Page

Example: `[ARG0: Alex] [V: left] [ARG2: for Delhi] [ARGM-COM: with his wife] [ARGM-TMP: at 5 pm] .` → `Alex left for Delhi at 5 pm with his wife.`
The transformation relies on propbank annotations (Bonial et al., 2012; Kingsbury and Palmer, 2002; Palmer et al., 2005; Gildea and Palmer, 2002).

A.75 ProtAugment Diverse Paraphrasing

This transformation utilizes the ProtAugment method by Dopierre et al. (2021). The paraphrase generation model is a BART model (Lewis et al., 2020), finetuned on the paraphrase generation task using 3 datasets: Google-PAWS (Zhang et al., 2019b), MSR (Dolan and Brockett, 2005), Quora[26].

When parpahrasing a sentence, the transformation useu Diverse Beam Search (Vijayakumar et al., 2016) to generate diverse outputs. The diversity penalty term is by default set to 0.5 but can be set to custom values. Additionally, the transformation can use the following generation constraints: (1) A fraction of the words in the input sentence are forbidden in the paraphrase (default 0.7). (2) All bi-grams in the input sentence are forbidden in the paraphrase. This means the paraphrase cannot contain any bi-gram that are in the input sentence. This constraint enforces the paraphrase generation model to change the sentence structure.

A.76 Punctuation

This transformation removes/adds punctuation from an English sentence. This transformation was first introduced by Mille et al. (2021) and used as an example implemention for NL-Augmenter.

A.77 Question-Question Paraphraser for QA

This transformation creates new QA pairs by generating question paraphrases from a T5 model fine-tuned on Quora Question pairs [27]. Generated questions can have a very different surface form from the original question making it a strong paraphrase generator. A T5 model (Raffel et al., 2019; Wolf et al., 2020) fine tuned [28] on the Quora Question Pairs dataset was being used to generate question paraphrases. This transformation would benefit Question Answering, Question Generation as well as other tasks which could indirectly benefit eg. for dialog tasks (Shrivastava et al., 2021; Dhole, 2020).

[26] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs
[27] Quora Question Pairs
[28] https://huggingface.co/ramsrigouthamg/t5_paraphraser

A.78 Question in CAPS

This transformation upper-cases the context of a question answering example. It also adds upper-cased versions of the original answers to the set of acceptable model responses.

A.79 Random Word Deletion

This transformation randomly removes a word with a given probability p (by default 0.25). The transformation relies on whitespace tokenization and thus only works for English and other languages that are segmented via whitespace. Due to the destructive nature of the transformation, it is likely that the meaning of a sequence may be changed as a result of the change. A similar transformation was suggested by Wei and Zou (2019). Word dropout (Goldberg, 2017) has been common to help models understand unknown words encountered during evaluation by exposing them to this unknown-word condition during training itself.

A.80 Random Upper-Case Transformation

This perturbation adds noise to all types of text sources (sentence, paragraph, etc.) by randomly adding upper cased letters. With a default probably of 0.1, each character in a sequence is upper-cased. This transformation does not rely on a tokenizer and thus works with all languages that have upper and lower-case letters. One limiation of this transformation is that it will not affect a tokenizer that does lower case for all input. A similar transformation was suggested by Wei and Zou (2019). Further improvement of this transformation exists by potentially relying on extreme value theory (Jalalzai et al., 2020).

A.81 Double Context QA

This transformation repeats the context of a question answering example. This should not change the result in any way.

A.82 Replace Abbreviations and Acronyms

This transformation changes abbreviations and acronyms appearing in an English text to their expanded form and respectively, changes expanded abbreviations and acronyms appearing in a text to their shorter form. For example, “send this file asap to human resources” might be changed to “send this file

as soon as possible to HR". The list of abbreviations and acronyms used in this transformation where manually gathered focusing on common abbreviations present in business communications. When abbreviation are context-dependent or highly specific, the induced change may change the meaning of a text, or an abbreviation may not be available in the lookup. The transformation was first introduced by Regina et al. (2020).

A.83 Replace Financial Amounts

This transformation replaces financial amounts throughout a text with the same value in a different currency. The replacement changes the amount, the writing format as well as the currency of the financial amount. For example, the sentence "*I owe Fred 20 and I need 10 for the bus.*" might be changed to "*I owe Fred 2 906.37 Yen and I need 1 453.19 Yen for the bus.*" The transformation was first introduced by Regina et al. (2020).

A.84 Replace Numerical Values

This transformation looks for numerical values in an English text and replaces it with another random value of the same cardinality. For example, "*6.9*" may be replaced by "*4.2*", or "*333*" by "*789*". The transformation was first introduced by Mille et al. (2021).

A.85 Replace Spelling

This transformation adds noise to all types of English text sources (sentence, paragraph, etc.) using corpora of common spelling errors introduced by Deorowicz and Ciura (2005). Each word with a common misspelling is replaced by the version with mistake with a probability p which by default is set to 0.2.

A.86 Replace nouns with hyponyms or hypernyms

This transformation replaces common nouns with other related words that are either hyponyms or hypernyms. Hyponyms of a word are more specific in meaning (such as a sub-class of the word), eg: 'spoon' is a hyponym of 'cutlery'. Hypernyms are related words with a broader meaning (such as a generic category /super-class of the word), eg: 'colour' is a hypernym of 'red'. Not every word will have a hypernym or hyponym.

A.87 Sampled Sentence Additions

This transformation adds generated sentence to all types of English text sources (sentence, paragraph, etc.) by passing the input text to a GPT-2 model (Radford et al., 2019). By default, GPT-XL is used, together with the prompt "*paraphrase:*" appended to the original text, after which up to 75 tokens are sampled. Since the additional text is sampled from a model, the model may introduce harmful language or generate text that contradicts the earlier text or changes its meaning. The idea to sample one or more additional sentences was first introduced by Jia and Liang (2017a).

A.88 Sentence Reordering

This perturbation adds noise to all types of text sources (paragraph, document, etc.) by randomly shuffling the order of sentences in the input text (Lewis et al., 2020). Sentences are first partially decontextualized by resolving coreference (Lee et al., 2018).

This transformation is limited to input text that has more than one sentence. There are still cases where coreference can not be enough for decontextualization. For example, there could be occurences of ellipsis as demonstrated by Gangal et al. (2021) or events could be mentioned in a narrative style which makes it difficult to perform re-ordering or shuffling (Kočiskỳ et al., 2018) while keeping the context of the discourse intact.

A.89 Emoji Addition for Sentiment Data

This transformation adds positive emojis and smileys to positive sentiment data and negative emojis to negative sentiment data. For non-labelled data, it adds neutral smileys.

A.90 Shuffle Within Segments

In this transformation, a token sequence, for example BIO-tagged, is split into coherent segments. Thus, each segment corresponds to either a mention or a sequence of out-of-mention tokens. For example, a sentence "*She did not complain of headache or any other neurological symptoms .*" with tags O O O O O B-problem O B-problem I-problem I-problem I-problem O is split into five segments: [*She did not complain of*], [*headache*], [*or*], [*any other neurological symptoms*], [.]. Then for each segment, a binomial distribution (p=0.5) is used to decide whether it should be shuffled. If yes, the order of the tokens within the segment is shuffled while the label order is kept unchanged. This transformation is inspired by Dai and Adel (2020).

A.91 Simple Ciphers

This transformation modifies the text in ways that a human could rapidly decipher, but which make the input sequences almost completely unlike typical input sequences which are used during language model training. This transformation includes the following text

modifications: double the characters, double the words, add spaces between the characters, reverse all characters in the text, reverse the characters within each word, reverse the order of the words in the text, substitute homoglyphs, rot13 cipher.

A.92 Slangificator

This transformation replaces some of the words (in particular, nouns, adjectives, and adverbs) of an English text with their corresponding slang. The replacement is done with the subset of the "Dictionary of English Slang & Colloquialisms".[29] The amount of replacement is proportional to the corresponding probabilities of replacement (by default, 0.5 for nouns, adjectives, and adverbs each).

A.93 Spanish Gender Swap

This transformation changes the gender of all animate entities (mostly referring to people, and some animals) in a given Spanish sentence from masculine to feminine. This includes masculine nouns with feminine equivalents (e.g., *doctor* *doctora*), nouns with a common gender ("sustantivos comunes en cuanto al género", e.g., *el violinista* *la violinista*), personal pronouns, and (optionally) given names often used with a given gender (e.g., *Pedro* *Alicia*). Epicene nouns are excluded. In addition, the gender of adjectives, determiners, pronouns and participles are modified in order to maintain the grammatical agreement.

A.94 Speech Disfluency Perturbation

This perturbation randomly inserts speech disfluencies in the form of filler words into English texts. With a given probability (0.2 by default), a speech disfluency is inserted between words. The default disfluencies are "um", "uh", "erm", "ah", and "er". At least one filler word is always inserted by this transformation.

A.95 Paraphrasing through Style Transfer

This transformation provides a range of possible styles of writing English language. The following styles can be chosen:

- Shakespeare - Trained on written works by Shakespeare.
- Switchboard - Trained on a collection of conversational speech transcripts.
- Tweets - Trained on 5.2M English tweets.
- Bible - Trained on texts from the Bible.
- Romantic poetry - Trained on romantic poetry.
- Basic - A light, basic paraphraser with no specific style.

The transformation follows the models and formulations by Krishna et al. (2020).

A.96 Subject Object Switch

This transformation switches the subject and object of English sentences to generate new sentences with a very high surface similarity but very different meaning. This can be used, for example, for augmenting data for models that assess Semantic Similarity.

A.97 Sentence Summarizaiton

This transformation compresses English sentences by extracting subjects, verbs, and objects of the sentence. It also retains any negations. For example, "*Stillwater is not a 2010 American live-action/animated dark fantasy adventure film*" turns into "*Stillwater !is film*". Zhang et al. (2021) used a similar idea to this transformation.

A.98 Suspecting Paraphraser for QA

This paraphraser transforms a yes/no question into a declarative sentence with a question tag [30], which helps to add more question specific informality to the dataset. Example: "Did the American National Shipment company really break its own fleet?" -> "The American National Shipment company really broke its own fleet, didn't it".

A.99 Swap Characters Perturbation

This perturbation randomly swaps two adjacent characters in a sentence or a paragraph with a default probability (Zhang et al., 2019a).

A.100 Synonym Insertion

This perturbation adds noise to all types of text sources (sentence, paragraph, etc.) by randomly inserting synonyms of randomly selected words excluding punctuations and stopwords (Marivate and Sefara, 2020).

A.101 Synonym Substitution

This perturbation randomly substitutes some words in an English text with their WordNet (Miller, 1998) synonyms.

[29] http://www.peevish.co.uk/slang/index.htm

[30] https://www.englishclub.com/grammar/tag-questions.htm

A.102 Syntactically Diverse Paraphrasing using Sow Reap models

This transformation is capable of generating multiple syntactically diverse paraphrases for a given sentence based on the work of Goyal and Durrett (2020). The model paraphrases inputs using a two step framework: 1) SOW (Source Order reWriting): This step enumerates multiple feasible syntactic transformations of the input sentence. 2) REAP (REarrangement Aware Paraphrasing): This step conditions on the multiple reorderings/ rearragements produced by SOW and outputs diverse paraphrases corresponding to these reoderings. The transformation is designed to work only on single-sentence inputs. Multi-sentence inputs results in an empty string/no transformation. The model are trained on the ParaNMT-50M dataset (Wieting and Gimpel, 2017; Wieting et al., 2017), which can be argued to be a bit noisy.

A.103 Subsequence Substitution for Sequence Tagging

This transformation performs same-label subsequence substitution for the task of sequence tagging, which replaces a subsequence of the input tokens with another one that has the same sequence of tags (Shi et al., 2021). This is done as follows: (1) Draw a subsequence A from the input (tokens, tags) tuple. (2) Draw a subsequence B within the whole dataset, with the same tag subsequence. (3) Substitute A with B in the input example.

A.104 Change English Tense

This transformation converts English sentences from one tense to the other, for example simple present to simple past. This transformation was introduced by Logeswaran et al. (2018).

A.105 Token Replacement Based on Lookup Tables

This transformation replaces input tokens with their perturbed versions sampled from a given lookup table of replacement candidates. Lookup tables containing OCR errors and misspellings from prior work are given as examples. Thus, by default, the transformation induces plausible OCR errors and human typos to the input sentence.

The transformation is an adapted and improved version of the lookup table-based noise induction method from Namysl et al. (2020). The OCR lookup table is from Namysl et al. (2021) and the misspellings from Piktus et al. (2019).

A.106 Transformer Fill

This perturbation replaces words based on recommendations from a masked language model. The transformation can limit replacements to certain POS tags (all enabled by default). Many previous papers have used this technique for data augmentation (Ribeiro et al., 2020; Li et al., 2020b, inter alia).

A.107 Underscore Trick

This perturbation adds noise to the text sources like sentence, paragraph, etc. This transformation acts like a perturbation to test robustness. It replaces some random spaces with underscores (or even other selected symbols). This perturbation would benefit all tasks which have a sentence/paragraph/document as input like text classification and text generation, especially on tasks related to understanding/generating scripts.

A.108 Unit converter

This transformation converts length and weight measures to different units (e.g., kilometers to miles) picking at random the new unit but converting accurately the quantity. The transformation conserves the format of the original quantity: "100 pounds" is converted to "1600 ounces" but "one-hundred pounds" is converted to "one thousand, six hundred ounces". Generated transformations display high similarity to the source sentences.

A.109 Urban Thesaurus Swap

This perturbation randomly picks nouns from the input source to convert to related terms drawn from the Urban Dictionary [31] resource. It can be applied to an input text to produce semantically-similar output texts in order to generate more robust test sets. We first select nouns at random, then query the Urban Thesaurus website [32] to obtain a list of related terms to swap in (Wilson et al., 2020).

A.110 Use Acronyms

This transformation changes groups of words for their equivalent acronyms. It's a simple substitution of groups of words for their acronyms. It helps to increase the size of the dataset as well as improving the understanding of acronyms of models trained on data augmented with this transformation. This transformations works to increase the data for any task that has input texts. It is specially interesting for tasks on semantic similarity, where models should be aware of the

[31] https://www.urbandictionary.com/
[32] https://urbanthesaurus.org/

equivalence between a set of words and their acronym. The quality of the transformation depends on the list of acronyms. As of now, this list was scraped from wikipedia's List of Acronyms [33] and naively filtered, which leaves space for improvement .

A.111 Visual Attack Letter

This perturbation replaces letters with visually similar, but different, letters. Every letter was embedded into 576-dimensions. The nearest neighbors are obtained through cosine distance. To obtain the embeddings the letter was resized into a 24x24 image, then flattened and scaled. This follows the Image Based Character Embedding (ICES) (Eger et al., 2019a).

The top neighbors from each letter are chosen. Some were removed by judgment (e.g. the nearest neighbors for 'v' are many variations of the letter 'y') which did not qualify from the image embedding (Eger et al., 2019b).

A.112 Weekday Month Abbreviation

This transformation abbreviates or expands the names of months and weekdays, e.g. Mon. -> Monday. Generated transformations display high similarity to the source sentences and does not alter the meaning and the semantic of the original texts. It does not abbreviate plural names, e.g. Sundays, and does not influence texts without names of weekdays or months.

A.113 Whitespace Perturbation

This perturbation adds noise to text by randomly removing or adding whitespaces.

A.114 Context Noise for QA

This transformation chooses a set of words at random from the context and the question and forms a sentence out of them. The sentence is then prepended or appended to the context to create a new QA pair. The transformation is inspired by the the **AddAny** method described in Adversarial SQUAD (Jia and Liang, 2017b). However, instead of probing the model to generate adversaries, random words from the context and the question are simply selected and joined together into a sentence, ignoring grammaticality. The transformation attempts to create novel QA pairs assuming that the introduction of random words to the context is less likely to change the answer choice to an asked question.

[33] https://en.wikipedia.org/wiki/Lists_of_acronyms

A.115 Writing System Replacement

This transformation replaces the writing system of the input with another writing system. We use CJK Unified Ideographs[34] as the source of characters for the generated writing systems. The transformation would benefit text classification tasks, especially in the cases where the input writing system is undeciphered.

A.116 Yes-No Question Perturbation

This transformation turns English non-compound statements into yes-no questions. The generated questions can be answered by the statements that were used to generate them. The text is left largely unchanged other than the fronted/modified/added auxiliaries and be-verbs.

The transformation works by getting dependency parse and POS tags from a machine learning model and applying human-engineered, rule-based transformations to those parses/tags. This transformation would particularly benefit question-answering and question-generation tasks, as well as providing surplus legal text for language modeling and masked language modeling.

A.117 Yoda Transformation

This perturbation modifies sentences to flip the clauses such that it reads like "Yoda Speak". For example, "Much to learn, you still have". This form of construction is sometimes called "XSV", where "the X being a stand-in for whatever chunk of the sentence goes with the verb", and appears very rarely in English normally. The rarity of this construction in ordinary language makes it particularly well suited for NL augmentation and serves as a relatively easy but potentially powerful test of robustness.

B Filters

The following is the list of all submitted filters to NL-Augmenter. Filters are used to filter data and create subpopulations of given inputs, according to features such as input complexity, input size, etc. Therefore, the output of a filter is a boolean value, indicating that whether the input meet the filter criterion. We discuss the implementations of each filter alongwith their limitations. The title of each filter subsection is clickable and redirects to the actual python implementation.

B.1 Code-Mixing Filter

This filter identifies whether the input text is code-mixed. It checks that there is at least one sentence in

[34] https://en.wikipedia.org/wiki/CJK_Unified_Ideographs

the text where there are tokens representing at least 'k' unique languages (with at least a 'threshold' level of confidence that the token is of that language). It is useful for collecting code-mixed data to test the model's performance on multilingual tasks. The filter relies on `ftlid`[35] for language detection, therefore, this filter might be limited by the performance of the language detection tool.

(containing code-mixing) Yo estaba con Esteban yesterday, he was telling me about lo que su esposa vio en los Estados Unidos. →True

B.2 Diacritics Filter

This filter checks whether any character in the sentence has a diacritic. It can be used to create splits of the dataet where the sentences have diacritics. Accented characters are typically among the rarer characters and checking the model performance on such a split might help investigate model robustness.

(containing diacritics) She lookèd east an she lookèd west. →True

B.3 Encoding Filter

This filter filters examples which contain characters outside a given encoding. It can be used to find examples containing e.g. non-ASCII Unicode characters. Filtering out and testing examples that contain these characters can provide feedback on how to improve the models accordingly, since most models are trained with plain English text, which contains mostly ASCII characters. Sometimes non-ASCII character are even explicitly stripped away.

(containing non-ASCII characters) That souvenir sure was expensive at 60č.. or was it 60? →True

B.4 Englishness Filter

This filter identifies texts that contain uniquely British spellings, vocabulary, or slang. The filter uses a vocabulary of common British words/phrases and checks the number of occurence of British words in the given texts. The text is selected if the number exceeds a pre-defined threshold.

(containing British spellings) Colour is an attribute of light that is perceived by the human eye. →True

B.5 Gender Bias Filter

This filter filters a text corpus to measure gender fairness with respect to a female gender representation. It supports four languages (i.e. English, French, Polish and Russian) and can be used to define whether the female gender is sufficiently represented in a tested subset of sentences. The filter uses a list of lexicals, which includes filter categories such as personal pronouns, words defining the relation, titles and names, corresponding to the female and male genders accordingly.

(texts with unbalanced representation) "He went home", "He drives a car", "She has returned" →True

B.6 Group Inequity Filter

This is a bilingual filter (for English and French languages), which helps to discover potential group inequity issues in the text corpus. It is a topic agnostic filter which accepts user-defined parameters, consisting of keywords inherent to minor group (which potentially might suffer from the discrimination), major group, minor factor and major factor. The filter first flags the sentences as belonging to the minor, and the major groups, and then, the sentences from each of the groups are used to define the intersection with both factors. The filter then compares whether the percentage of major factors exceeds that of the minor factors to determine if the sentences have group inequity issues.

(containing group inequity issues) "He is a doctor", "She is a nurse", "She works at the hospital" →True

B.7 Keyword Filter

This is a simple filter, which filters examples based on a pre-defined set of keywords. It can be useful in creating splits for a specific domain.

(containing keyword "at") Andrew played cricket in India →True

B.8 Language Filter

This filter selects texts that match any of a given set of ISO 639-1 language codes (the default language being English). Language matching is performed using a pre-trained `langid.py` model instance. The model provides normalized confidence scores. A minimum threshold score needs to be set, and all sentences with confidence scores above this threshold are accepted by the filter.

(is English texts) Mein Luftkissenfahrzeug ist voller Aale →False

B.9 Length Filter

This filter filters data with the input text length matching a specified threshold. It can be useful in creating data with different length distributions.

(containing more than 3 words) Andrew played cricket in India →True

[35] https://pypi.org/project/ftlid/

B.10 Named-entity-count Filter

This filter filters data where the number of Named Entities in the input match a specified threshold (based on the supported conditions).

(containing more than 1 named entity) Novak Djokovic is the greatest tennis player of all time. →True

B.11 Numeric Filter

This filter filters example which contain a numeric value. In the tasks like textual entailment, question answering etc., a quantity (number) could directly affect the final label/response. This filter can be used to create splits to measure the performance separately on texts containing numeric values.

(containing numbers in texts) John bought a car worth dollar twenty five thousand . →True

B.12 Oscillatory Hallucinations Filter

This filter is designed to operate in text generation systems' outputs, with the purpose of extracting oscillatory hallucinations. Oscillatory hallucinations are one class of hallucinations characterized by repeating bigram structure in the output(Raunak et al., 2021). Typically, these behaviors are observed in models trained on noisy corpora. The filter counts the frequency of bigrams in both source and target texts, and compare the frequency difference with a pre-set threshold to determine whether the texts includes oscillatory hallucinations.

(containing hallucinations in target texts) Source: "Community, European Parliament common regional policy, Mediterranean region", Target: "Arbeitsbedingungen, berufliche Bildung, berufliche Bildung, berufliche Bildung" →True

B.13 Polarity Filter

This filter filters a transformed text if it does not retain the same polarity as an original text. This filter helps not to distort training data during augmentation for sentiment analysis-related tasks. While generating new data for a sentiment analysis task, it is important to make sure that generated data is labelled correctly.

(texts retaining polarity) "Hotel is terrible", "Hotel is great" →False

B.14 Quantitative Question Filter

This is a simple rule-based filter that can be used to identify quantitative questions. It can help to analyse models' performance on questions which require numerical understanding. It is also useful to study possible biases in question generation.

(being quantitative question) How long does the journey take? →True

B.15 Question type filter

This filter helps identify the question category of a question answering example based on the question word or the named entity type of the answer. Knowledge of the question type can help in the development of question answering systems (Parikh et al., 2019) as well as for assessing performance on individual splits.

(being where question) Where is Delhi located ? →True

B.16 Repetitions Filter

This filter finds texts with repetitions with simple heuristic rules. It might be helpful in finding repetitions that frequently occur in the spoken language data.

(containing repetitions in texts) I I want to sleep →True

B.17 Phonetic Match Filter

This filter selects texts that contain matching entries to a list of supplied keywords. It first transform the input sentence and the keywords into phenetic units and then compare whether the two phenetic unit sets have overlap.

(containing homophones of keyword "trombone") I left my trombno on the train →True

B.18 Special Casing Filter

This filter checks if the input sentence has a special casing, i.e. the string is either all lowercased, all uppercased or has title casing. It might be useful for creating splits that contain texts with unusual casing, e.g. misspellings.

(text being uppercased/lowercased/titlecased) let's go to chipotle →True

B.19 Speech-Tag Filter

This filter filters an example text based on a set of speech tags and identifies whether the count of selected POS tags meet the pre-defined conditions (e.g. above the threshold).

(containing 1 verb and 2 numbers in texts) It all happened between November 2007 and November 2008. →True

B.20 Token-Amount filter

This filter filters an example text based on whether certain keywords are present in a specified amount.

(containing 2 occurances of "in") Andrew played cricket in a soccer stadium in India at 9pm →True

B.21 Toxicity Filter

This filter filters an example text which has a toxicity value matching a particular threshold. It uses a pre-trained toxicity detector, which can provide 7 toxicity scores. All the 7 types of toxicity scores can be used as criteria for the filtering.

(text being toxic) I disagree. It is not supposed to work that way. →False

B.22 Universal Bias Filter

This filter works the same way as the Gender Bias Filter, but measures balance of representation for more categories (religion, race, ethnicity, gender, sexual orientation, age, appearance, disability, experience, education, economic status). The lexical seeds representing these categories are currently available in English only, however the pool of languages can be extended by a simple addition of the lexical seeds in a desired language to the lexicals.json file.

(texts being biased) "He is going to make a cake.", "She is going to program", "Nobody likes washing dishes", "She agreed to help him" →False

B.23 Yes/no question filter

This filter allows to select questions that can be correctly answered with either 'yes' or 'no'. Since it is rule-based, the limitation of this filter is that questions that are ambiguous might not be recognized.

(text being yes/no question) Wasn't she angry when you told her about the accident? →True

C Review criteria for submission evaluation

Figure 3 shows the detailed review criteria used for evaluating the transformation and filters submissions.

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Augmentation	34	20	0.63	-13.25	20	0.75	-6	18	0.74	-8.89	17	0.73	-4.41
Bias	3	1	0.5	-5	2	0.52	-11.5	2	0.53	-16	1	0.71	0
Robustness	15	8	0.82	-9.38	7	0.59	-8.14	7	0.65	-12.14	7	0.88	-13.71
Other*	1	1	0.5	-38	1	0.5	-23	1	0.5	-44	1	0.6	1
Multiple*	21	13	0.72	-4.15	13	0.64	-5.08	12	0.68	-4.08	11	0.92	-5.64
Total	74	43			43			40			37		

Table 7: Results of the robustness evaluation from the perspective of the **General purpose** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Qual. estim.	2	2	0.52	-2.5	2	0.51	-6	2	0.53	-6.5	1	0.56	0
Question ans.	3	2	0.7	-0.5	2	0.89	-1.5	2	0.77	-1	2	0.98	-4
Question gen.	2	1	0.41	0	1	0.77	-1	1	0.54	-2	1	0.97	-5
RDF to text	1	1	0.01	0	1	0.02	0	1	0.04	0	1	0.21	0
Sentiment ana.	4	1	0.99	-12	1	0.99	-14	1	0.93	-18	1	1	-15
Table to text	1	1	0.01	0	1	0.02	0	1	0.04	0	1	0.21	0
Text class.	95	52	0.71	-9.27	52	0.68	-6.21	49	0.69	-8.33	43	0.83	-5.74
Text tagging	25	17	0.79	-10.94	17	0.64	-6.82	16	0.66	-9.75	13	0.84	-9.23
Text to text gen.	92	49	0.69	-8.86	49	0.66	-5.86	46	0.68	-7.57	40	0.79	-5.62
Total	231	126			126			119			103		

Table 8: Results of the robustness evaluation from the perspective of the **Task type** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Semantic	3	1	1	-35	1	1	-20	1	1.0	-42	1	1	-3
Lexical	44	30	0.67	-5.83	30	0.61	-5	30	0.64	-4.4	25	0.73	-2.44
Syntactic	3	1	1	-8	1	0.74	-7	1	0.85	-15	1	1	0
Word-order	2	2	0.6	-1.5	2	0.61	-1	2	0.63	-2	1	1	0
Morphological	3	2	0.75	-25.5	2	0.75	-21.5	2	0.75	-28.5	2	0.8	-4.5
Character	6	2	1	-16.5	2	1.0	-12.5	1	0.95	-31	2	1	-26
Other*	1	1	0	0	1	0.7	-4	0			1	1	-1
Multiple*	25	9	0.74	-11.22	9	0.71	-7	9	0.74	-12.56	8	0.8	-14.5
Unclear	1	1	1	-46	1	0.79	-2	0			0		
Total	92	49			49			46			41		

Table 9: Results of the robustness evaluation from the perspective of the **Linguistic level** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Aural	5	3	1	-4.33	3	0.7	-6.67	2	0.7	-6.5	3	0.85	-3.67
Meaning	51	31	0.6	-8.58	32	0.64	-5.72	31	0.64	-7.52	28	0.74	-5.75
Visual	12	7	0.86	-15.29	6	0.8	-10.17	5	0.8	-12.8	5	0.92	-1
Other*	5	1	0.83	0	1	0.55	-4	1	0.69	-2	0		
Multiple*	2	1	1	-34	1	1	-20	1	1.0	-38	2	1	-23
N/A	2	2	0.92	-1	2	0.67	-6	2	0.77	-5	0		
Total	77	45			45			42			38		

Table 10: Results of the robustness evaluation from the perspective of the **Input/output similarity** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Alw. preserved	40	22	0.65	-9.77	22	0.63	-7.36	22	0.61	-11.23	19	0.72	-9.89
Poss. changed	33	20	0.78	-5.45	20	0.73	-5.15	17	0.75	-4.76	18	0.87	-1.5
Alw. changed	12	5	0.7	-4	5	0.54	-5.4	5	0.61	-6.8	3	0.78	-7.33
Alw. added	2	1	0	-94	1	0.7	-4	1	0.78	0	1	0.99	-1
Poss. removed	2	2	1	-18	2	1	-13	2	0.88	-23.5	1	1	-3
Total	89	50			50			47			42		

Table 11: Results of the robustness evaluation from the perspective of the **Meaning preservation** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Alw. preserved	31	19	0.59	-10.58	19	0.52	-4.63	18	0.53	-8.11	17	0.76	-4.94
Poss. impaired	36	20	0.69	-3.15	20	0.69	-4.55	19	0.72	-4.21	18	0.81	-2.11
Alw. impaired	2	1	0.93	-7	1	0.94	-20	1	0.92	-16	1	1	-1
Poss. improved	6	6	0.83	-16.33	6	0.8	-8.17	5	0.79	-14.8	2	0.52	-1.5
Unclear	1	1	1	-34	1	1	-20	1	1.0	-38	1	1	-45
N/A	2	2	1	-23.5	2	1	-22	2	1	-27	2	1	-36.5
Total	79	49			49			46			41		

Table 12: Results of the robustness evaluation from the perspective of the **Grammaticality preservation** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Alw. preserved	25	15	0.66	-3	15	0.54	-3.47	15	0.56	-5.53	12	0.83	-2.33
Poss. impaired	38	24	0.64	-10.67	24	0.69	-6.25	22	0.69	-6.59	22	0.79	-2.41
Alw. impaired	9	4	1	-25.25	4	1.0	-17.25	3	0.98	-36.67	4	1	-40
Poss. improved	4	4	0.75	-11.75	4	0.75	-8.75	4	0.75	-16.25	2	0.52	-1.5
Alw. improved	2	1		-1	1		-6	1	0.77	-5	0		
Unclear	1	1	1	0	1	0.06	0	1	0.15	0	1	0.32	0
Total	79	49			49			46			41		

Table 13: Results of the robustness evaluation from the perspective of the **Readability preservation** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Alw. preserved	18	9	0.59	-3.33	10	0.52	-3.5	9	0.51	-7.44	9	0.75	-2.56
Poss. impaired	45	29	0.66	-8.48	29	0.64	-5.38	27	0.67	-5.15	24	0.79	-1.75
Alw. impaired	8	4	1.0	-20.5	4	1.0	-16.25	4	0.97	-23.25	4	1	-32.25
Poss. improved	4	4	0.75	-11.75	4	0.75	-8.75	4	0.75	-16.25	2	0.52	-1.5
Unclear	1	1	1	-34	1	1	-20	1	1.0	-38	1	1	-45
Total	77	47			48			45			40		

Table 14: Results of the robustness evaluation from the perspective of the **Naturalness preservation** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Addition	1	1	0	-94	1	0.7	-4	1	0.78	0	1	0.99	-1
Paraphrasing	5	5	0.79	-1.8	5	0.74	-5.6	4	0.77	-6.25	3	0.77	-0.67
Parsing	1	1	0.02	0	1	0.16	-1	1	0.15	0	1	0.59	0
PoS-Tagging	5	3	0.44	-11.67	3	0.54	-6.67	3	0.54	-14.33	2	0.98	-1.5
Removal	2	2	1	-4.5	2	0.74	-6.5	2	0.81	-10	1	1	0
Segmentation	3	1	1	-4	1	0.93	-6	1	0.94	-5	1	1	-4
Substitution	17	13	0.63	-8.08	14	0.61	-8	14	0.64	-9.36	13	0.67	-5
Tokenisation	23	9	0.67	-4.89	9	0.5	-4.22	9	0.54	-4.56	10	0.76	-3.8
Translation	3	2	0.99	-11	2	0.99	-13.5	2	0.97	-18.5	1	1	-15
Other*	3	2	1	-17	2	1.0	-10	1	0.95	-38	2	1	-23
Multiple*	13	6	0.69	-1.33	5	0.6	-2.2	5	0.58	-4.8	3	0.72	-2
Unclear	1	1	1	-46	1	0.79	-2	0			0		
N/A	3	2	0.85	-18.5	2	0.9	-14	2	0.89	-20.5	2	1	-32
Total	81	48			48			45			40		

Table 15: Results of the robustness evaluation from the perspective of the **Input data processing** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
Model-based	19	11	0.95	-11.27	11	0.93	-7.64	9	0.93	-11.78	7	0.81	-2.43
Rule-based	66	38	0.65	-9.24	38	0.61	-6.26	37	0.64	-8.14	34	0.79	-6.5
Both	6	2	0.31	0	2	0.5	-0.5	2	0.42	-1.5	1	0.97	-5
Unclear	1	1	1	-7	1	0.84	-4	1	0.9	-2	1	1	-1
Total	103	52			52			49			43		

Table 16: Results of the robustness evaluation from the perspective of the **Implementation** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

		SST-2 Roberta-base			QQP BERT-base-unc.			MNLI Roberta-large			IMDB Roberta-base		
Tag	$\#_{All}$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$	$\#_{Evl}$	$\mathbf{R}_T$	$\mathbf{Var}_S$
API-based	22	14	0.78	-7.86	14	0.67	-7	13	0.73	-9.23	11	0.88	-11.45
Ext. K.-based	33	19	0.47	-11	19	0.55	-6.95	19	0.55	-7.89	20	0.68	-4.45
LSTM-based	1	1	1	0	1	1.0	0	0	0.9		1	1	-1
Transf.-based	15	7	0.89	-9.57	7	0.85	-5.29	6	0.87	-7.17	1	1	-4
Multiple*	3	1	0.41	0	1	0.77	-1	1	0.54	-2	1	0.97	-5
Unclear	1	0			0			0			1		-1
N/A	24	4	1.0	-13.25	4	0.77	-8.5	4	0.75	-18.75	3	0.89	-6
Total	103	46			46			43			38		

Table 17: Results of the robustness evaluation from the perspective of the **Algorithm type** criterion ($\#_{All}$ = Total number of tags, $\#_{Evl}$ Total number of evaluations collected, R_T = Transformation rate, Var_S = Score variation)

Correctness: Transformations must be valid Python code and must pass tests.

Interface: Participants should ensure that they use the correct interface. The complete list is mentioned here. E.g., for tasks like machine translation, a transformation which changes the value of a named entity (Andrew->Jason) might need parallel changes in the output too. And hence, it might be more appropriate to use `SentenceAndTargetOperation` or `SentenceAndTargetsOperation` rather than `SentenceOperation`. Similarly, if a transformation changes the label of a sentence, the interface's generate method should take as input the label too - eg. if your transformation reverses the sentiment, `SentenceAndTargetOperation` would be more appropriate then `SentenceOperation`. If you wish to add transformations for input formats other than those specified, you should add an interface here.

Applicable Tasks & Keywords: We understand that transformations can vary across tasks as well as a single transformation can work for multiple tasks. Hence all the tasks where the transformation is applicable should be specified in the list "tasks". The list of tasks has been specified here. The relevant keywords for the transformation should also be specified.

```
class ButterFingersPerturbation(SentenceOperation):
    tasks = [TaskType.TEXT_CLASSIFICATION, TaskType.TEXT_TO_TEXT_GENERATION, TaskType.TEXT_TAGGING]
    languages = ["en"]
    keywords = ["morphological", "noise", "rule-based", "high-coverage", "high-precision"]
```

Specificity: While this is not a necessary criterion, it is highly encouraged to have a specific transformation. E.g., a perturbation which changes gendered pronouns could give insights about gender bias in models.

Novelty: Your transformation must improve the coverage of NL-Augmenter in a meaningful way. The idea behind your transformation need not be novel, but its contribution to the library **must be different from the contributions of earlier submissions**. If you are unsure if your idea would constitute a new contribution, please email the organizers at nl-augmenter@googlegroups.com and we are happy to help.

Adding New Libraries: We welcome addition of libraries which are light and can be installed via `pip`. Every library should specify the version number associated and be added in a new requirements.txt in the transformation's own folder. However, we discourage the use of heavy libraries for a few lines of code which could be manually written instead. Please ensure that all libraries have MIT, Apache 2, BSD, or other permissive license. GPL-licensed libraries are not approved for NL-Augmenter. If you are unsure, please email the organizers at nl-augmenter@googlegroups.com.

Description: The `README.md` file should clearly explain what the transformation is attempting to generate as well as the importance of that transformation for the specified tasks. Here is a sample README.

Data and code source: The `README.md` file should have a subsection titled "Data and code provenance", which should describe where data or code came from, or that it was fully created by the author. This section should also disclose the license that any external data or code is released under.

Paraphrasers and Data Augmenters: Besides perturbations, we welcome transformation methods that act like paraphrasers and data augmenters. For non-deterministic approaches, we encourage you to specify metrics which can provide an estimate of the generation quality. We prefer high precision transformation generators over low accuracy ones. And hence it's okay if your transformation selectively generates.

Test Cases: We recommend you to add at least 5 examples in the file `test.json` as test cases for every added transformation. These examples serve as test cases and provide reviewers a sample of your transformation's output. The format of `test.json` can be borrowed from the sample transformations here. A good set of test cases would include good as well as bad generation. Addition of the the test cases is **not mandatory** but is encouraged.

Evaluating Robustness: To make a stronger PR, a transformation's potential to act as a robustness tool should be tested via executing `evaluate.py` and the corresponding performance should be mentioned in the README. Evaluation should only be skipped in case there is no support in the evaluation_engine.

Languages other than English: We strongly encourage multilingual perturbations. All applicable languages should be specified in the list of "languages".

Decent Programming Practise: We recommend adding docstrings to help others follow your code with ease. Check the PEP 257 Docstring Conventions to get an overview. If you are using spacy, we suggest you use the common global version like this.

All of the above criteria extend to filters too.

Figure 3: Participants and reviewers were provided with a set of review criteria.

D Contributor Affiliations

Kaustubh D. Dhole[3,18†], Varun Gangal[7†], Sebastian Gehrmann[23†], Aadesh Gupta[3†], Zhenhao Li[32†], Saad Mahamood[90†], Abinaya Mahendiran[45†], Simon Mille[53†], Ashish Shrivastava[2†], Samson Tan[91†], Tongshuang Wu[7†], Jascha Sohl-Dickstein[22†], Jinho D. Choi[18†], Eduard Hovy[7†], Ondrej Dusek[10†], Sebastian Ruder[13†], Sajant Anand[68], Nagender Aneja[74], Rabin Banjade[77], Lisa Barthe[19], Hanna Behnke[32], Ian Berlot-Attwell[80], Connor Boyle[81], Caroline Brun[49], Marco Antonio Sobrevilla Cabezudo[79], Samuel Cahyawijaya[26], Emile Chapuis[52], Wanxiang Che[24], Mukund Choudhary[37], Christian Clauss[33], Pierre Colombo[52], Filip Cornell[41], Gautier Dagan[84], Mayukh Das[63], Tanay Dixit[30], Thomas Dopierre[39], Paul-Alexis Dray[89], Suchitra Dubey[1], Tatiana Ekeinhor[86], Marco Di Giovanni[51], Tanya Goyal[4], Rishabh Gupta[29], Louanes Hamla[19], Sang Han[73], Fabrice Harel-Canada[70], Antoine Honoré[86], Ishan Jindal[27], Przemyslaw K. Joniak[66], Denis Kleyko[75], Venelin Kovatchev[65], Kalpesh Krishna[71], Ashutosh Kumar[34], Stefan Langer[59], Seungjae Ryan Lee[55], Corey James Levinson[33], Hualou Liang[15], Kaizhao Liang[76], Zhexiong Liu[78], Andrey Lukyanenko[43], Vukosi Marivate[14], Gerard de Melo[25], Simon Meoni[33], Maxime Meyer[86], Afnan Mir[4], Nafise Sadat Moosavi[62], Niklas Muennighoff[50], Timothy Sum Hon Mun[64], Kenton Murray[40], Marcin Namysl[20], Maria Obedkova[33], Priti Oli[77], Nivranshu Pasricha[46], Jan Pfister[83], Richard Plant[17], Vinay Prabhu[73], Vasile Pais[57], Libo Qin[24], Shahab Raji[58], Pawan Kumar Rajpoot[56], Vikas Raunak[44], Roy Rinberg[11], Nicholas Roberts[82], Juan Diego Rodriguez[72], Claude Roux[49], Vasconcellos P. H. S.[54], Ananya B. Sai[30], Robin M. Schmidt[16], Thomas Scialom[89], Tshephisho Sefara[12], Saqib N. Shamsi[88], Xudong Shen[48], Yiwen Shi[15], Haoyue Shi[67], Anna Shvets[19], Nick Siegel[4], Damien Sileo[42], Jamie Simon[68], Chandan Singh[68], Roman Sitelew[33], Priyank Soni[3], Taylor Sorensen[6], William Soto[61], Aman Srivastava[85], KV Aditya Srivatsa[37], Tony Sun[69], Mukund Varma T[30], A Tabassum[47], Fiona Anting Tan[36], Ryan Teehan[9], Mo Tiwari[60], Marie Tolkiehn[8], Athena Wang[4], Zijian Wang[33], Zijie J. Wang[21], Gloria Wang[31], Fuxuan Wei[24], Bryan Wilie[35], Genta Indra Winata[5], Xinyi Wu[81], Witold Wydmanski[38], Tianbao Xie[24], Usama Yaseen[59], Michael A. Yee[92], Jing Zhang[18], Yue Zhang[87]

[1]ACKO, [2]Agara, [3]Amelia R&D, New York, [4]Applied Research Laboratories, The University of Texas at Austin, [5]Bloomberg, [6]Brigham Young University, [7]Carnegie Mellon University, [8]Center for Data and Computing in Natural Sciences, Universität Hamburg, [9]Charles River Analytics, [10]Charles University, Prague, [11]Columbia University, [12]Council for Scientific and Industrial Research, [13]DeepMind, [14]Department of Computer Science, University of Pretoria, [15]Drexel University, [16]Eberhard Karls University of Tübingen, [17]Edinburgh Napier University, [18]Emory University, [19]Fablab by Inetum in Paris, [20]Fraunhofer IAIS, [21]Georgia Tech, [22]Google Brain, [23]Google Research, [24]Harbin Institute of Technology, [25]Hasso Plattner Institute / University of Potsdam, [26]Hong Kong University of Science and Technology, [27]IBM Research, [28]IIIT Delhi, [29]IIT Delhi, [30]IIT Madras, [31]Illinois Mathematics and Science Academy, [32]Imperial College, London, [33]Independent, [34]Indian Institute of Science, Bangalore, [35]Institut Teknologi Bandung, [36]Institute of Data Science, National University of Singapore, [37]International Institute of Information Technology, Hyderabad, [38]Jagiellonian University, Poland, [39]Jean Monnet University, [40]Johns Hopkins', [41]KTH Royal Institute of Technology, [42]KU Leuven, [43]MTS AI, France, [44]Microsoft, Redmond, WA, [45]Mphasis NEXT Labs, [46]National University of Ireland Galway, [47]National University of Science and Technology, Pakistan, [48]National University of Singapore, [49]Naver Labs Europe, [50]Peking University, [51]Politecnico di Milano and University of Bologna, [52]Polytechnic Institute of Paris, [53]ADAPT/Dublin City University, [54]Pontifical Catholic University of Minas Gerais, Brazil, [55]Princeton University, [56]Rakuten India, [57]Research Institute for Artificial Intelligence Mihai Drgnescu, Romanian Academy, [58]Rutgers University, [59]Siemens AG, [60]Stanford University, [61]SyNaLP, LORIA, [62]TU Darmstadt, [63]Technical University of Braunschweig, [64]The Alan Turing Institute, [65]The University of Texas at Austin; (University of Barcelona, University of Birmingham), [66]The University of Tokyo, [67]Toyota Technological Institute at Chicago, [68]UC Berkeley, [69]UC Santa Barbara / Google, [70]UCLA, [71]UMass Amherst, [72]UT Austin, [73]UnifyID, [74]Universiti Brunei Darussalam, [75]University of California, Berkeley and Research Institutes of Sweden, [76]University of Illinois, Urbana Champaign, [77]University of Memphis, [78]University of Pittsburgh, [79]University of São Paulo, [80]University of Toronto, [81]University of Washington, [82]University of WisconsinMadison, [83]University of Würzburg, [84]Universty of Edinburgh, [85]VMware, [86]Vade, [87]Westlake Institute for Advanced Study, [88]Whirlpool Corporation, [89]reciTAL, [90]trivago N.V., [91]AWS AI Research & Education, [92] University of Michigan, [91] Work done independent of AWS tenure.