ArgU: A Controllable Factual Argument Generator

Sougata Saha and Rohini Srihari
State University of New York at Buffalo
Department of Computer Science and Engineering
{sougatas, rohini}@buffalo.edu

Abstract

Effective argumentation is essential towards a purposeful conversation with a satisfactory outcome. For example, persuading someone to reconsider smoking might involve empathetic, well founded arguments based on facts and expert opinions about its ill-effects and the consequences on one's family. However, the automatic generation of high-quality factual arguments can be challenging. Addressing existing controllability issues can make the recent advances in computational models for argument generation a potential solution. In this paper, we introduce ArgU: a neural argument generator capable of producing factual arguments from input facts and real-world concepts that can be explicitly controlled for stance and argument structure using Walton's argument scheme-based control codes. Unfortunately, computational argument generation is a relatively new field and lacks datasets conducive to training. Hence, we have compiled and released an annotated corpora of 69,428 arguments spanning six topics and six argument schemes, making it the largest publicly available corpus for identifying argument schemes; the paper details our annotation and dataset creation framework. We further experiment with an argument generation strategy that establishes an inference strategy by generating an "argument template" before actual argument generation. Our results demonstrate that it is possible to automatically generate diverse arguments exhibiting different inference patterns for the same set of facts by using control codes based on argument schemes and stance.

1 Introduction

Although arguing is an innate human quality, formulating convincing arguments is an art. A successful narrative aiming to persuade someone should be rhetorically appealing, trustworthy, factually correct, and logically consistent, which makes formulating good arguments challenging. Incorporating neural language models, the relatively new field of computational argument generation has shown promise in assisting with argument synthesis. Argument generators like Project Debater (Slonim et al., 2021) have successfully formulated convincing arguments across different domains including legal, politics, education, etc., and can potentially find new argumentative connections. However, lacking explicit control mechanisms, neural argument generators often render illogical and inappropriate arguments, reducing their trustworthiness and applicability for practical use. Furthermore, training such models requires a considerable amount of quality data, which is hard to collect and annotate. Hence, we propose ArgU, a controllable neural argument generator trained on a curated and quality-controlled corpus of annotated argument texts from abortion, minimum wage, nuclear energy, gun control, the death penalty and school uniform.

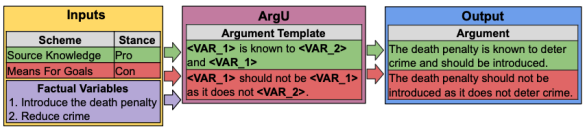

Figure 1: Generating stance and argument scheme controlled factual arguments using ArgU.

ArgU strives to enable effective, scalable and appealing argument generation. As depicted in Figure 1, it takes as input worldly knowledge and concepts as fact variables and coherently combines them to generate an argument that exhibits the desired pro/con stance and inference structure. Using control codes to regulate argument stance and reasoning, ArgU generates a variety of argument texts for the same set of facts [1], thus providing diverse response options. Internally ArgU implements a 2-step generation process, where it first generates an

[1]Unless explicitly mentioned, the term fact refers to real-world concepts, propositions, and knowledge and does not refer to only knowledge-based facts.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 8373–8388

"argument template", which depicts the structure of the final argument based on the control codes, and finally yields the argument text by modifying the template to include the augmented input fact variables. We ground our work on prominent theoretical foundations, where the inference structure-based control codes derive from six Walton's argument schemes: "Means for Goal", "Goal from Means", "From Consequence", "Source Knowledge", "Source Authority", and "Rule or Principle".

Since human annotation is expensive and time-consuming, we devise a multi-phased annotation framework for systematically leveraging human and automatic annotation mechanisms to yield a curated dataset of 69,428 examples for controllable argument synthesis. We release our curated corpus to facilitate further research; an example constitutes an argument text, a set of real-world concepts and knowledge from which the argument derives, and the stance and argument scheme of the text. We further detail and analyze our annotation framework and share variants of topic-independent computational models for automatically annotating factual spans from argument text and identifying the asserted argument schemes. We share our datasets and codebase here: *https://github.com/sougata-ub/argu-generator* and summarize our contributions below:

- We propose an argument generator that methodically generates factual arguments following a specified stance and argument scheme (Sec. 4).
- We share a quality-controlled annotated dataset conducive to training such generators. To our knowledge, this is the largest available corpora that identify argument schemes from argument text (Sec. 3.2.4).
- We share our annotation framework and release domain-independent computational models that automatically identify factual spans and argument schemes from argument text from any topic (Sec. 3).

2 Related Work

Argument schemes are typical inference patterns found in arguments. Walton provided an in-depth study of argument schemes (Walton et al., 2008) and defined 60 such schemes prevalent in daily argument text. Based on Walton's argumentation schemes, Kondo et al. (2021) proposed representing the reasoning structure of arguments using Bayesian networks and defined abstract network fragments termed idioms, which we use here.

Advances in neural methods for language modelling have enabled the field of computational argument generation. Hua and Wang (2018) introduced a factual argument generator that generates opposite stance arguments by yielding a set of talking point key phrases, followed by a separate decoder to produce the final argument text. Hua et al. (2019) proposed Candela, a framework for counter-argument generation similar to Hua and Wang (2018), which also controls for the style. Schiller et al. (2021) introduced Arg-CTRL: a language model for generating sentence-level arguments using topic, stance, and aspect-based control codes (Keskar et al., 2019). Khatib et al. (2021) constructed argumentation-related knowledge graphs and experimented with using them to control argument generation. Alshomary et al. (2021) explored a novel pipelined approach to generating counter-arguments that first identifies a weak premise and then attacks it with a neurally generated counter-argument. Hypothesizing that the impact of an argument is strongly affected by prior beliefs and morals, Alshomary et al. (2022) studied the feasibility of the automatic generation of morally framed argument text and proposed an argument generator that follows the moral foundation theory. Syed et al. (2021) introduced the task of generating informative conclusions from arguments. They compiled argument text and conclusion pairs and experimented with extractive and abstractive models for conclusion generation using control codes. Chakrabarty et al. (2021) experimented with argument text re-framing for positive effects. They created a suitable corpus and trained a controllable generator with a post-decoding entailment component for re-framing polarizing and fearful arguments such that it can reduce the fear quotient. Our work best aligns with Arg-CTRL and Candela, where we use control codes to regulate argument generation and implement a multi-step decoding pipeline to generate the final argument. However, unlike Arg-CTRL, we control for the argument scheme, and unlike Candela, our multi-step decoding utilizes an argument template as an intermediate step.

Most argumentation datasets identify argumen-

tative components (claims, premises, etc.), making them better suited for argument-mining tasks (Stab and Gurevych, 2014; Peldszus, 2015; Ghosh et al., 2016; Hidey et al., 2017; Chakrabarty et al., 2019). Further, existing argument scheme annotated corpora are either very restricted in domain and size (Reed et al., 2008; Feng and Hirst, 2011; Green, 2015; Musi et al., 2016; Visser et al., 2022; Jo et al., 2021) or only provide guidelines and tools for annotations (Visser et al., 2018; Lawrence et al., 2019). Hence, we use the BASN dataset (Kondo et al., 2021), which contains sizeable examples spanning six topics and identify argument schemes.

3 Argument Generation Corpus

Training a factual argument generator controlled for the stance and argument scheme requires examples that identify such features from the text: such a corpus is lacking. Hence, we introduce a two-phased annotation framework that yields a corpus of 69,428 examples which (i) identify argument schemes and factual spans from argument text and (ii) grounds the spans to a knowledge base (KB). In the first phase, we employ human annotators to identify factual spans from a subset of an existing dataset of 2,990 arguments which already identifies argument schemes. We further train computational models to annotate the remaining corpus for factual spans and perform extensive quality checks. In the second phase, we train models from the resultant Phase 1 dataset to automatically annotate a larger parallel corpus for both argument scheme and factual spans, yielding an annotated corpus of 69,428 arguments for training argument generators.

3.1 Phase 1 (P1): Initial Corpus Creation

Kondo et al. (2021) introduced the BASN dataset comprising 2,990 pairs of arguments and abstract network fragments derived from six Walton's argumentation schemes: "Means for Goal", "Goal from Means", "From Consequence", "Source Knowledge", "Source Authority", "Rule or Principle", and "Others". They utilized a knowledge base (KB) of 205 facts (termed as variables) spanning the topics of abortion, minimum wage, nuclear energy, gun control, the death penalty and school uniform to define the idioms. Figure 2 illustrates an example from the BASN dataset where variables from the KB formulate a pro-stance argument following the "Means for goals" argument scheme. We perform two annotation tasks in P1: (i) **Span Detection**: Annotate arguments by identifying (highlighting) non-overlapping factual spans from argument text. (ii) **Span Grounding**: Ground the identified factual spans to the available KB variables, or "Others" if the span is unrelated to any available variables.

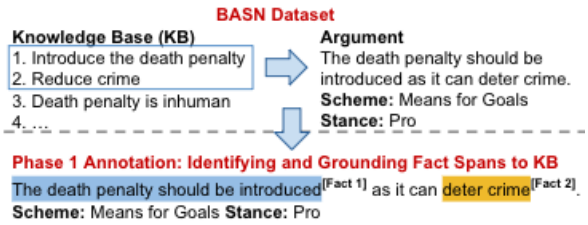

Figure 2: Phase 1 Annotation Pipeline.

We annotate 1,153 randomly sampled examples spanning all six topics and train a model for automatically annotating the remaining examples. We further perform human evaluations to determine the correctness of the automatic annotations.

3.1.1 Human Expert Annotation

Using Doccano (Nakayama et al., 2018), we annotated 1,153 examples from the BASN corpus for both the tasks of span detection and grounding, where each sample comprised an argument and a minimum of 2 to a maximum of 5 fact variables from the KB. Figure 8 (Appendix A) contains a screenshot from our Doccano annotation task. We employed two computational linguistics and computer science graduate students as paid expert annotators for the annotation task. Both annotators were appointed and compensated as per the legal norms. To be efficient with resources, each annotator independently annotated non-overlapping examples. Further, to ensure consistency across annotations, we computed inter-annotator agreement over 100 samples, which resulted in a Cohen's Kappa score of 0.78, indicating substantially high agreement.

3.1.2 Automatic Annotation: ArgSpan

We train ArgSpan: a Roberta-based tagger (Liu et al., 2019), on the annotated examples for automatically annotating the rest of the BASN dataset for both tasks. Figure 3 illustrates ArgSpan's architecture. ArgSpan inputs concatenated argument and fact variables and encodes them using a Roberta-based encoder. It reduces the hidden representation for each fact variable by passing the beginning of the string token (BOS) through a fully connected neural network layer. Finally, it uses a biaffine layer to capture the interaction between the argument text and each variable. The model is

trained end-to-end by minimizing the cross entropy loss between the predicted logit for each argument token and the actual BIO scheme encoded target label. Appendix A.1 contains further training details.

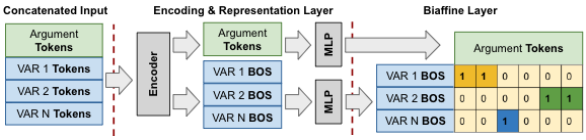

Figure 3: ArgSpan Architecture.

3.1.3 Evaluation

We automatically annotate the remaining BASN samples using ArgSpan. To gauge the quality of the automatic annotations, we ask one of the human evaluators to annotate 300 random examples from the remaining samples using Doccano and compare them with the model predictions. Detailed in Figure 4, we evaluate Span Detection by computing the F1 score between the overlapping predicted and human-identified tokens and achieve an average score of 91.1% across all 300 examples. We measure accuracy for evaluating Span Grounding and attain a score of 89.2%. With the additional 300 examples (total of 1,453), we re-train ArgSpan and perform inference on the remaining BASN samples, yielding a fully annotated corpus of 2,990 examples with KB-grounded factual spans and argument schemes from argument text. Also, we observe very few examples of the "Goal From Means" scheme in the resultant dataset and combine it with the more prevalent "Means for Goal" scheme, resulting in six argument schemes.

Human Annotation: **Criminologists** [1]
familiar with the effects of **the DP on crime** [2]
assert that the **DP does not deter crime** [3].

ArgSpan Annotation: **Criminologists** [2]
familiar with the effects of **the DP on crime** [2]
assert that the **DP does not deter crime** [3].

True Positives (TP) = (Criminologists, the, DP, on, crime, DP, does, not, deter, crime) = 10
False Positives (FP) = (the) = 1
False Negatives (FN) = (effects, of) = 2
Span Detection F1= 0.87
Label Accuracy = 2/3 = **0.66**

Figure 4: ArgSpan Evaluation.

3.2 Phase 2 (P2): Corpus Expansion

Kondo et al. (2021) used crowd-sourcing to create the BASN dataset, where crowd workers formulated argument text from a knowledge base comprising a limited number of premise-conclusion pairs (fact variables). Although such an approach resulted in a considerable number of arguments, using approximately 34 fact variables per topic, it lacks variety. Training an argument generator on such a corpus would limit its generalizability and use. Hence, we expand the P1 dataset with a parallel corpus (**PC**) of 66,180 examples from the Aspect-Controlled Reddit and CommonCrawl corpus by Schiller et al. (2021), and 733 combined examples from the Sentential Argument Mining, Arguments to Key Points and the debate portal-based Webis datasets (Stab et al., 2018; Friedman et al., 2021; Bar-Haim et al., 2020; Ajjour et al., 2019). Since the PC examples do not identify factual spans and argument schemes, we use the fully annotated P1 dataset to train **ArgSpanScheme**: a Roberta-based model that identifies factual spans and argumentation schemes from argument text. We automatically annotate the PC using ArgSpanScheme and combine them with the P1 dataset, to yield the P2 dataset.

3.2.1 ArgSpanScheme Architecture

Illustrated in Figure 5, we experiment with two variants of ArgSpanScheme to jointly extract factual spans and predict argument schemes from argument text. Both architectures use a Roberta-based encoder to encode an input argument text and differ in the final prediction layers, as detailed below.

Parallel Architecture Here we use two independent classification heads: (i) A span detection head which uses a linear layer to extract factual spans by classifying each encoded argument token as belonging to one of the three BIO tags. (ii) A scheme detection head which uses a linear layer to predict argument schemes by performing a multi-label (six labels including "Others") classification on the mean pooled encoded argument tokens.

Pipelined Architecture Argument schemes represent structures of inference and are invariant to the constituent facts. For example, although both arguments A: "Increase in the minimum wage is not favourable as it can increase unemployment", and B: "Increase in gun laws are favourable as it reduces gun violence", are from different topics, they follow a similar structure "X is/are (not) favourable as it Y", exhibiting "From Consequences" argument scheme. As depicted in Figure 5, we model this by performing selective multi-headed attention. We mask the factual spans predicted by the span de-

tection head and apply two layers of multi-headed self-attention on the remaining tokens. Finally, we pass the BOS token representation through a linear layer to predict the argument schemes. Appendix A.2 contains further training details.

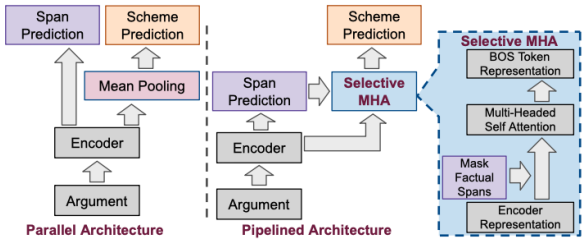

Figure 5: ArgSpanScheme Architectures.

3.2.2 Modelling Results and Evaluation

For both tasks of span and scheme detection, we compare the F1 score of the parallel and pipelined architectures across different data splits. We perform a 5-fold Cross Validation (CV) by randomly splitting the resultant dataset from P1 into 93% training and 7% validation split. We further assess the generalizability of ArgSpanScheme by training and validating on examples from non-overlapping topics. As illustrated in Figure 6, we set up five data splits (ids 1 to 5) comprising three combination ratios of training-validation topics (5:1, 4:2, and 2:4), which increases the difficulty by reducing the number of training topics.

ID	Topics						Name
0	5-fold Cross Validation: [Random Splits of 93% Train & 7% Validation] x 5						CV
1	GC	MW	NE	SU	DP	AB	5:1
2	GC	MW	NE	SU	DP	AB	
3	GC	MW	NE	SU	DP	AB	4:2
4	GC	MW	NE	SU	DP	AB	
5	GC	MW	NE	SU	DP	AB	2:4

Training
Validation

GC: Gun Control, **MW**: Minimum Wage, **NE**: Nuclear Energy, **SU**: School Uniform, **DP**: Death Penalty, **AB**: Abortion

Figure 6: ArgSpanScheme Data Splits.

Evaluating Span Prediction: For span detection we compute the F1 score at three levels of overlap: (i) **Partial Overlap**: A span level metric where a predicted span is true positive if at least 50% of its tokens overlap with the actual span. (ii) **Full Overlap**: A span level metric where a predicted span is true positive if all of its tokens overlap with the actual span. (iii) **Overall**: A token level metric which compares the predicted and actual token BIO labels. Table 1 shares the CV and combination ratio aggregated results for span detection. We observe similar performance for both ArgSpanScheme versions across all three levels of overlap.

Evaluating Scheme Prediction: We compare scheme-wise and overall F1 scores and share the results in Table 1. We observe that the parallel architecture slightly outperforms the pipelined version in CV, whereas the pipelined version almost always performs better for the non-overlapping splits. The results indicate that for scheme detection, incorporating a generalizable architecture by emphasizing the argument structure rather than the factual spans does lead to better results on unseen topics.

3.2.3 Automatic Annotation & Human Eval.

Based on the analysis of automatic evaluation results, we train a final pipelined version of ArgSpanScheme on the P1 dataset and perform inference on the PC to automatically annotate it for factual spans and argument schemes. We randomly sample 200 annotations and perform a human evaluation using one evaluator to ascertain the annotation quality.

Evaluating Span Prediction: We present the human evaluator with an argument text along with the model predicted spans and ask them to rate each example using two custom metrics: (i) **Span Precision**: On a continuous scale of 1 (low) to 5 (high), how sensible are the identified spans? Spans which are unnecessarily long or abruptly short are penalized. This metric evaluates whether the identified spans adequately convey meaningful information. (ii) **Span Recall**: On a continuous scale of 1 (low) to 5 (high), how well does the model perform in identifying all factual spans? Examples which fail to identify spans conveying real-world concepts and factual knowledge are penalized. We observe an average score of 4.1 (median 4.7) for Span Precision and 3.9 (median 4.4) for Span Recall, indicating the reliability of the automatic annotations.

Evaluating Scheme Prediction: Since identifying argument schemes is a much more difficult task, we first measure the evaluator's competency by presenting 30 random arguments from the BASN dataset and asking them to label each argument text with the most likely argument scheme. We compared the evaluator-assigned labels with the golden labels and found them to be matching in 53.3% of cases, with most matches belonging to the "from consequences", "rule or principle", and "means for goal" schemes. Although the labels majorly confirm, the fair amount of disagreement testifies to the task difficulty. Further, Table 5 (Appendix A)

	Span			Scheme						
Split	**Partial**	**Full**	**Overall**	**From Consequence**	**From Source Authority**	**From Source Knowledge**	**Goal From Means/ Means from Goals**	**Rule or Principle**	**Other**	**Overall**
CV	**0.86**/0.85	**0.92**/0.91	0.89/0.89	**0.94**/0.93	**0.92**/0.91	0.88/**0.90**	**0.96**/0.95	**0.97**/0.96	**0.88**/0.86	**0.95**/0.94
5:1	0.70/0.70	0.77/**0.78**	0.81/0.81	0.65/0.65	0.68/**0.85**	0.48/0.48	0.48/**0.56**	0.64/**0.66**	0.46/0.46	0.68/**0.69**
4:2	0.76/**0.77**	0.84/0.84	0.85/0.85	0.60/**0.71**	0.67/**0.70**	0.49/0.49	0.45/**0.47**	0.49/**0.55**	0.49/0.49	0.75/**0.82**
2:4	0.74/0.74	**0.82**/0.80	**0.82**/0.80	0.63/**0.73**	**0.69**/0.67	**0.50**/0.49	**0.47**/0.46	0.73/**0.77**	0.46/0.46	0.70/**0.77**

Table 1: ArgSpanScheme span and scheme prediction results for Parallel / Pipelined versions. The best performing model for each data split and task is highlighted in bold.

lists a few examples where we believe the evaluator labels are more accurate than the actual ones. Post-assessment, we asked the evaluator to evaluate the predicted argument schemes of the previously sampled 200 examples with a binary flag, where 1 signifies agreement and 0 signifies disagreement, and observe a fair agreement rate of 73%.

3.2.4 Dataset Post-processing

The PC initially contains 1,272,548 examples, which we automatically annotate for span and argument scheme using ArgSpanScheme. We persist samples where an argument scheme's predicted probability is at least 20% of the scheme's average probability and discard examples with the scheme predicted as "Others".

To make the PC consistent with the P1 data, we implement the following steps to normalize and ground the ArgSpanScheme-identified factual spans to the existing KB comprising fact variables from BASN or expand the KB with new knowledge wherever applicable. (i) **Direct Mapping**: Using sentence transformer embedding-based cosine similarity (Reimers and Gurevych, 2019) and a threshold of 0.85, we associate factual spans from the annotated PC with its most similar fact variable from the KB. (ii) **Indirect Mapping**: We use the sentence transformer-based community detection clustering algorithm to cluster similar factual spans from the annotated PC. For directly unmapped spans, we associate the KB fact variable of the nearest neighbour in its cluster. Figure 9 (Appendix A) further illustrates each step in detail.

We apply a series of filtering steps to ensure the quality of the final corpus. We only keep examples containing a maximum of 30% unnormalized factual spans and add those facts to the KB. Next, we discard instances containing more than 150 words in the argument text and persist examples containing 1-4 fact variables, with each variable present 2-4 times. Finally, to ensure argumentativeness, we parse the argument text using the Dialo-AP argument parser (Saha et al., 2022) and keep examples containing at least one claim. We combine the filtered PC with the P1 dataset to yield 69,428 examples, which we use for argument generation.

4 Controllable Argument Generation

Arguments based on similar facts but structured differently might lead to dissimilar consequences by exerting different perlocutionary effects. For example, consider argument A: "Reproductive rights advocates say enabling access to abortion is important towards reproductive rights", which exhibits the "From Source Authority" argument scheme, and B: "Access to abortion is important towards reproductive rights", which expresses "From Consequence". Although both arguments share the same view regarding the role of abortion in reproductive rights, backed by reproductive rights advocates who are experts, argument A might lead to a favourable outcome in a situation that demands authority. To assist the formulation of arguments exhibiting heterogeneous viewpoints and reasoning, we experiment with BART-based (Lewis et al., 2020) neural argument generators capable of generating factual argument text with distinct stances and argument schemes using control codes.

4.1 Model Architecture

Figure 7 illustrates our encoder-decoder based model architecture, which we discuss below.

4.1.1 Encoder

The model inputs a concatenated representation I_1 of the argument topic and the required KB fact variables. We prefix each variable with a token <VAR_X> where X $\in$ [0, 3] is an incremental id enforcing a random ordering over the variables. The representation I_1 is passed through a BART encoder E to yield a hidden representation H.

4.1.2 Decoder

A BART based decoder inputs H along with a set of control codes to generate the final argument A. We experiment with two types of decoding:

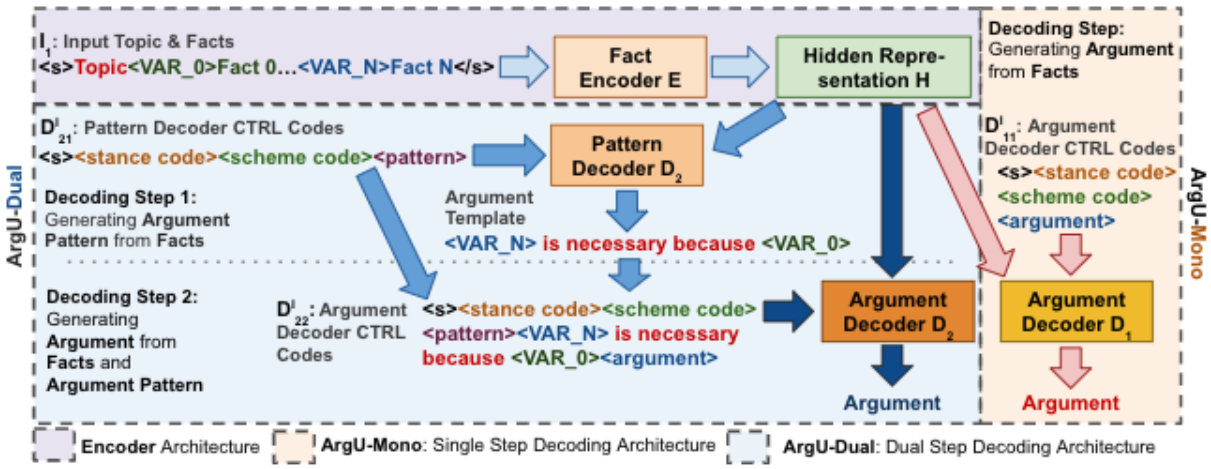

Figure 7: ArgU-Mono and Dual End-to-end Architectures.

Single Step Decoding: ArgU-Mono: As depicted in Figure 7, following the standard decoding strategy of an encoder-decoder architecture, the decoder D_1 inputs H along with three control codes (D^I_{11}) comprising the desired stance, argument scheme, and the argument text BOS token '<argument>', and learns the distribution $P(A|I_1, D^I_{11})$.

Dual Step Decoding: ArgU-Dual An argument generally exhibits structured reasoning by coherently combining variables using appropriate connectives and clauses. For example, the variables A: "introduce death penalty" and B: "reduce crime" can be combined as "A **has shown evidence in** B", resulting in a pro-death penalty argument "Introducing the death penalty **has shown evidence in** reducing crime". Following the same template of "A **has shown evidence in** B", the variables A: "enforce gun laws" and B: "reduce gun violence" can be combined to form an argument "Enforcing gun laws **has shown evidence in** reduction of gun violence". The ArgU-Dual architecture implements "argument templates" to model this property, where distinct argument texts exhibit similar structure and reasoning over variables.

To condition the argument generation on its template, we train decoder D_2 to create an argument template T before generating the actual argument A. As depicted in Figure 7, D_2 inputs H and a set of three control codes (D^I_{21}) comprising the desired stance, argument scheme, and the template BOS token '<pattern>', to learn the probability distribution $P(T|I_1, D^I_{21})$. Next, we suffix T with the argument BOS token '<argument>', and pass through D_2 to generate the final argument text and learn the distribution $P(A|T, D^I_{22})$.

4.2 Training, Experiments and Results

We use the resultant P2 dataset for our experiments and create random train-test set of 67,728 and 1,700 examples. To analyze the effect of each type of control code, we also perform ablation analysis and train two model variants: **ArgU-Stance** and **ArgU-Scheme**. Both implementations follow the same encoding and decoding steps as ArgU-Mono, with the only difference being the absence of scheme or stance-based control codes in respective architectures. Training details in Appendix A.3.

4.2.1 Automatic Evaluation Results

Apart from comparing standard metrics like corpus BLEU (Papineni et al., 2002) and Rouge-L (Lin, 2004), we define the following metrics to evaluate each model. (i) **Fact Faithfulness (Fact)**: This evaluates fact faithfulness by measuring the similarity between the input variables and the generated argument. We use the sentence transformer's semantic textual similarity to compute the average cosine similarity between the embeddings of the input variables and the model-generated argument, where a higher score correlates with better utilization of the fact variables. (ii) **Entailment (Entail) & Contradiction (Contra)**: This evaluates the relatedness between the original and generated argument. We use AllenNLP's (Gardner et al., 2018) Roberta-based textual entailment model pre-trained on the SNLI dataset (Bowman et al., 2015) to determine whether a generated argument entails (higher better) or contradicts (lower better) the original argument with at least 0.8 probability.

We share our results in Table 2 and observe that compared to others, ArgU-Dual majorly yields bet-

Model	BLEU	RougeL	Fact	Entail	Contra
Mono	0.150	0.379	0.641	0.399	0.140
Dual	**0.158**	**0.381**	0.641	**0.406**	0.144
Stance	0.151	0.375	0.641	0.400	**0.133**
Scheme	0.151	0.377	**0.642**	0.360	0.191

Table 2: Argument generation automatic evaluation results with best model highlighted for each metric.

Model	Fluency (K=0.61)	Stance (K=0.87)	Scheme (K=0.9)	Fact (K=0.68)	Logic (K=0.71)
Mono	**4.99**	0.78*	**0.83**	**3.89**	4.01
Dual	4.86	0.80*	**0.83**	3.88	4.06
Stance	4.95	**0.84**	0.79*	3.85	3.98*
Scheme	4.98	0.65*	0.79*	3.81	**4.17**

Table 3: Argument generation human evaluation results with best model highlighted for each metric. * denotes scores with at least 5% difference w.r.t the best score.

ter BLEU and RougeL scores and attains the best entailment results, indicating a better correlation with the original argument. On the contrary, using only argument schemes and stance-based control codes generally performs worse. We also observe that ArgU-Mono performs almost at par with ArgU-Stance across all metrics, whereas ArgU-Scheme contradicts the original argument the most. The results not only indicate the benefit of using both stance and scheme-based control codes but also indicate the superiority of the Dual architecture compared to Mono.

4.2.2 Human Evaluation Results

We perform a human evaluation study using the evaluators from Section 3.1.1. We created a worksheet with 50 random examples from the test set, where an example constitutes the argument topic, input KB variables, desired stance and argument scheme, the original argument from the dataset, and the generated argument text from each of the four models. The evaluators were asked to rate each generated argument text on the following five metrics. (i) **Fluency**: On a scale of 1 (low) to 5 (high), this scores the fluency and grammatical correctness of an argument. (ii) **Stance Appropriateness (Stance)**: On a binary scale, this determines if the stance exhibited by a generated argument aligns with the desired stance passed as control code. (iii) **Scheme Appropriateness (Scheme)**: On a binary scale, this determines if the argument scheme exhibited by a generated argument aligns with the desired scheme passed as control code. (iv) **Fact Faithfulness (Fact)**: On a scale of 1 (low) to 5 (high), this determines how well the generated argument incorporates the input variables. Ignoring variables or including additional facts (hallucination) are penalized. (v) **Logical Coherence (Logic)**: A subjective metric that rates the overall sensibleness of the logic portrayed by the generated argument text on a scale of 1 (low) to 5 (high).

We measure inter-annotator agreement by computing Cohens kappa (K) and observe substantial to high agreement across all metrics. Table 3 shares the averaged ratings from both evaluators. For each metric, we highlight in bold the best performing model(s) and mark with an asterisk the model(s) where the difference from the best is at least 5%. We further plot (in Appendix A) the scores by each metric in Figure 10, the variation of each metric across models in Figure 11, and the distribution of scores for each metric in Figure 12.

We observe a high variation across models for stance, followed by scheme and logic, and low variations for fact and fluency metrics, indicating stance, scheme, and logic-based metrics are better suited for measuring model performance. The low variation in fluency and fact metrics indicate that all models are fluent in generating arguments while incorporating the input variables, with ArgU-Mono performing the best. Trained with only stance-based control codes, ArgU-Stance yields the best results for stance appropriateness, while trained with only scheme-based control codes, ArgU-Scheme rates the lowest. Contrastly, ArgU-Scheme attains the highest rating for generating logically coherent arguments, whereas ArgU-Stance achieves the lowest rating. Thus, indicating the usefulness of using stance and scheme-based control codes for argument text generation. The ArgU-Dual and Mono variants rate similarly for both metrics, and rate high for scheme appropriateness, indicating that using control codes, the stance and scheme of an argument can be successfully controlled in tandem.

4.3 Discussion

Table 4 contains arguments generated by ArgU-Dual. Examples 1 and 2 show the model's capability of generating authoritative argument text with the correct stance by referring to human rights advocates and supporters, thus exhibiting the "From Source Authority" argument scheme. Similarly, examples 3 and 4 denote the model's capability of generating appropriate argument text following the "Rule or Principle" argument scheme for both

ID	Topic	Variables	Scheme	Stance	Argument Template	Argument Text	Comments
1	Death Penalty	<VAR_0> human rights around the world <VAR_1> mandatory death sentence	From Source Authority	Pro	<VAR_0>supporters of the bill say it is a step toward <VAR_1>	human rights supporters of the bills say it is a step towards a mandatory death sentence	Generated arg incorporates input control codes, variables and generated arg template
2				Con	<VAR_0>advocates have long argued that <VAR_1>	human rights advocates have long advocated that mandatory death sentences should be abolished	
3			Rule or Principle	Pro	<VAR_1>is not a violation of <VAR_0>	mandatory death sentence is not a violation of human rights	
4				Con	<VAR_1>is a violation of <VAR_0>	mandatory death sentence is a violation to international human rights law	
5	Abortion	<VAR_0> reproductive health and rights advocates <VAR_1> stop people from having abortions	From Consequence	Pro	<VAR_1>is an important step toward <VAR_0>	banning abortion is an important stepping toward reproductive rights	Pro & con args swapped
6				Con	<VAR_1>does nothing to <VAR_0>	banning abortion does nothing to advance women s reproductive rights	
7			From Source Knowledge	Pro	<VAR_1>has been proven to be effective in <VAR_0>	restricting access to abortion has been proved to be ineffective in protecting women s reproductive rights	Generated arg template modified during arg generation
8				Con	<VAR_1>is not the answer to <VAR_0>	banning abortion is not the solution to women s reproductive rights	

Table 4: ArgU Generated Samples.

stances. Examples 5 and 6 depict a scenario where the generator demonstrates shallow understanding and inanely combines the input variables, yielding contrasting stance arguments. Examples 7 and 8 highlight cases where the argument decoder modifies the generated argument template, which in example 7 changes the meaning of the argument.

5 Conclusion

Here we propose ArgU: A neural factual argument generator that systematically generates arguments following a specified stance and argument scheme. We devise a multi-step annotation framework to yield two golden and silver standard annotated datasets that we further use to train multiple ArgU variants. Implementing automatic and human evaluation, we thoroughly analyze ArgU's generation capabilities. Our findings indicate ArgU's applicability for aiding users to formulate situation-specific arguments by controlling the argument stance and scheme using control codes.

Acknowledgements

We thank the anonymous reviewers for providing valuable feedback on our manuscript. This work is partly supported by NSF grant number IIS2214070. The content in this paper is solely the responsibility of the authors and does not necessarily represent the official views of the funding entity.

Limitations

As depicted in Table 4, there are scenarios where ArgU demonstrates a lack of understanding and instead paraphrases the input variables to generate an incorrect response. It seems likely that the model associates negation with Con. However, in examples 5 and 6, the model does not factor the word "stop" in Variable 1, leading to arguments that contradict the intended stance. Further, in examples 7 and 8, the argument decoder seems to modify the generated template, which changes the overall meaning of example 7. Such scenarios might reduce the trust in the model, hurting its practical use.

All experiments involving ArgSpan, ArgSpanScheme, and ArgU only pertain to abortion, minimum wage, nuclear energy, gun control, the death penalty and school uniform. The model performance on any other topics is unknown. Although we test ArgSpanScheme on out-of-domain test sets, it still confines the six topics. Since ArgU is trained only on argument sentences with less than 150 tokens, it is more geared towards generating shorter arguments of less than 50 tokens. We further do not benchmark ArgU's inference time for practical use.

Ethics Statement

We acknowledge that all experiments were performed ethically and purely from an academic point of view. Although this research revolves around arguments from six sensitive topics and pre-trained models, the argument generators at our end are not explicitly trained to be discriminatory, exhibit bias, or hurt anyone's sentiments. Further, any generated text does not reflect the stance of the authors. The human evaluators were appointed and compensated as per the legal norms.

References

Yamen Ajjour, Milad Alshomary, Henning Wachsmuth, and Benno Stein. 2019. Modeling frames in argu-

mentation. 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 2922–2932, Hong Kong, China. Association for Computational Linguistics.

Milad Alshomary, Roxanne El Baff, Timon Gurcke, and Henning Wachsmuth. 2022. The moral debater: A study on the computational generation of morally framed arguments. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8782–8797, Dublin, Ireland. Association for Computational Linguistics.

Milad Alshomary, Shahbaz Syed, Arkajit Dhar, Martin Potthast, and Henning Wachsmuth. 2021. Counter-argument generation by attacking weak premises. In *FINDINGS*.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tuhin Chakrabarty, Christopher Hidey, and Smaranda Muresan. 2021. Entrust: Argument reframing with language models and entailment. *ArXiv*, abs/2103.06758.

Tuhin Chakrabarty, Christopher Hidey, Smaranda Muresan, Kathy McKeown, and Alyssa Hwang. 2019. AMPERSAND: Argument mining for PERSuAsive oNline discussions. 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 2933–2943, Hong Kong, China. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996, Portland, Oregon, USA. Association for Computational Linguistics.

Roni Friedman, Lena Dankin, Yufang Hou, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2021. Overview of the 2021 key point analysis shared task. In *Proceedings of the 8th Workshop on Argument Mining*, pages 154–164, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 549–554, Berlin, Germany. Association for Computational Linguistics.

Nancy Green. 2015. Identifying argumentation schemes in genetics research articles. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 12–21.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21, Copenhagen, Denmark. Association for Computational Linguistics.

Xinyu Hua, Zhe Hu, and Lu Wang. 2019. Argument generation with retrieval, planning, and realization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2661–2672, Florence, Italy. Association for Computational Linguistics.

Xinyu Hua and Lu Wang. 2018. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 219–230, Melbourne, Australia. Association for Computational Linguistics.

Yohan Jo, Seojin Bang, Chris Reed, and Eduard Hovy. 2021. Classifying argumentative relations using logical mechanisms and argumentation schemes. *Transactions of the Association for Computational Linguistics*, 9:721–739.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Khalid Al Khatib, Lukas Trautner, Henning Wachsmuth, Yufang Hou, and Benno Stein. 2021. Employing argumentation knowledge graphs for neural argument generation. In *ACL*.

Takahiro Kondo, Koki Washio, Katsuhiko Hayashi, and Yusuke Miyao. 2021. Bayesian argumentation-scheme networks: A probabilistic model of argument validity facilitated by argumentation schemes. In *Proceedings of the 8th Workshop on Argument Mining*,

pages 112–124, Punta Cana, Dominican Republic. Association for Computational Linguistics.

John Lawrence, Jacky Visser, and Chris Reed. 2019. An online annotation assistant for argument schemes. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 100–107. Association for Computational Linguistics.

Mike Lewis, 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, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Elena Musi, Debanjan Ghosh, and Smaranda Muresan. 2016. Towards feasible guidelines for the annotation of argument schemes. In *Proceedings of the third workshop on argument mining (ArgMining2016)*, pages 82–93.

Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. doccano: Text annotation tool for human. Software available from https://github.com/doccano/doccano.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

Andreas Peldszus. 2015. An annotated corpus of argumentative microtexts.

Chris Reed, Raquel Mochales Palau, Glenn Rowe, and Marie-Francine Moens. 2008. Language resources for studying argument. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Sougata Saha, Souvik Das, and Rohini K. Srihari. 2022. Dialo-AP: A dependency parsing based argument parser for dialogues. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 887–901, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2021. Aspect-controlled neural argument generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–396, Online. Association for Computational Linguistics.

Noam Slonim, Yonatan Bilu, Carlos Alzate, Roy Bar-Haim, Ben Bogin, Francesca Bonin, Leshem Choshen, Edo Cohen-Karlik, Lena Dankin, Lilach Edelstein, et al. 2021. An autonomous debating system. *Nature*, 591(7850):379–384.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. 2018. Cross-topic argument mining from heterogeneous sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3664–3674, Brussels, Belgium. Association for Computational Linguistics.

Shahbaz Syed, Khalid Al-Khatib, Milad Alshomary, Henning Wachsmuth, and Martin Potthast. 2021. Generating informative conclusions for argumentative texts. In *Findings*.

Jacky Visser, John Lawrence, Chris Reed, Jean Wagemans, and Douglas Walton. 2022. Annotating argument schemes. In *Argumentation Through Languages and Cultures*, pages 101–139. Springer.

Jacky Visser, John Lawrence, Jean Wagemans, and Chris Reed. 2018. Revisiting computational models of argument schemes: Classification, annotation, comparison. In *7th International Conference on*

Computational Models of Argument, COMMA 2018, pages 313–324. ios Press.

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

A Appendix

A.1 ArgSpan Training Details

We initialize ArgSpan weights with pre-trained Roberta base weights, and train using 2 Nvidia RTX A5000 GPUs with mixed precision (Micikevicius et al., 2018) and a batch size of 32. Prior to the biaffine layer, we reduce the hidden representation to 600 dimensions. We use a learning rate of 1E-5 and train till the validation loss stops improving for five steps. We also clip (Pascanu et al., 2013) the gradients to a unit norm and use AdamW (Loshchilov and Hutter, 2019) with the default PyTorch parameters for optimization.

A.2 ArgSpanScheme Training Details

We initialize ArgSpanScheme weights with pre-trained Roberta base weights, and train using 1 Nvidia RTX A5000 GPUs with mixed precision and a batch size of 64. We use 2 layers of multi-headed self attention using 4 attention heads. We use a learning rate of 1E-5 and train till the validation loss stops improving for five steps. We also clip the gradients to a unit norm and use AdamW with the default PyTorch parameters for optimization.

A.3 ArgU Training Details

We initialize model weights with pre-trained BART (Lewis et al., 2020) base weights and expand the embedding layer to accommodate 13 new tokens, detailed in Table 6 (Appendix A). We train all models over 2 Nvidia RTX A5000 GPUs with mixed precision and a batch size of 24. We use a learning rate of 1E-5 and train till the validation loss stops improving for five steps. We also clip the gradients to a unit norm and use AdamW with the default PyTorch parameters for optimization. We use beam search for decoding with a beam length of 5, a maximum length of 50 tokens, and a penalty for trigram repetitions in the generated argument.

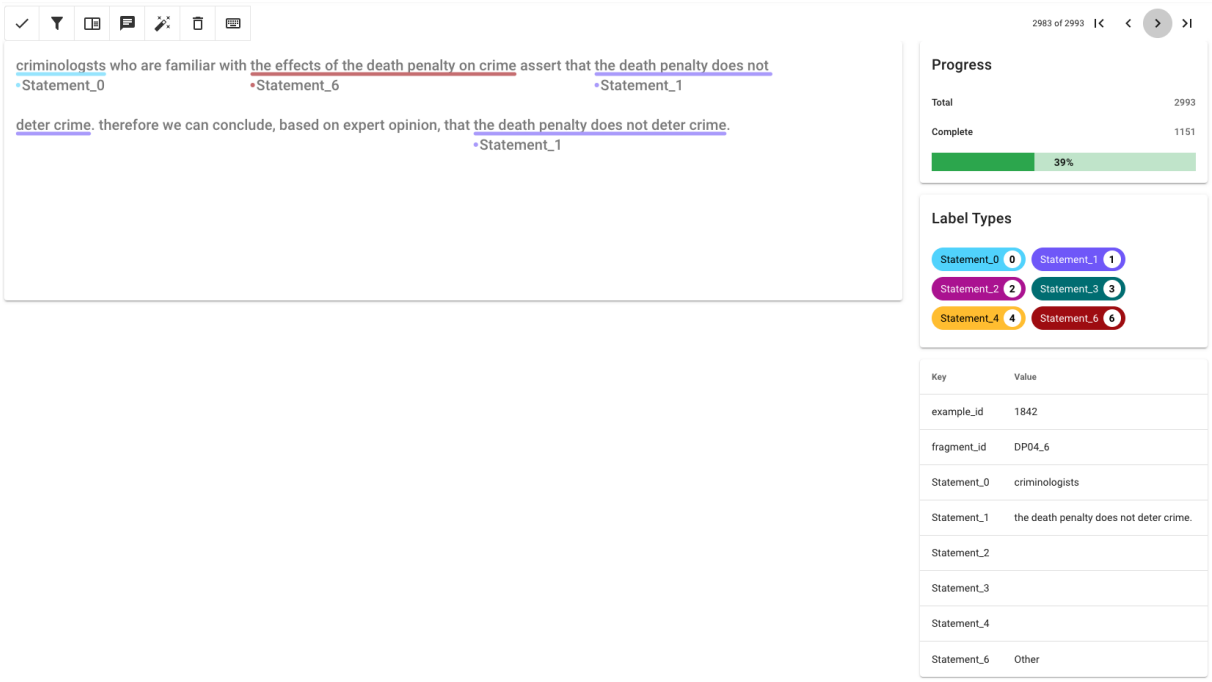

Figure 8: Doccano Annotation Screenshot.

ID	**Argument**	**Actual Label**	**Annotator Label**
1	abortion is necessary, because, unintended pregnancies are associated with birth defects, increased risk of child abuse, ad so on.	means for goal	from consequence
2	most students do not believe that school uniforms are useful, so uniforms should not be required.	from source knowledge	from source authority
3	the death penalty is unacceptable because of the racial bias in the criminal justice system. the death penalty does not follow a fair criminal justice system because of its racial bias.	rule or principle	from source authority
4	it is not necessary to require school uniforms, because t is important to respect students who believe that school uniforms are not necessary.	from source authority	from source knowledge
5	increasing the minimum wage reduces income inequality. reducing income inequality is desirable. we should increase the minimum wage.	from consequence	means for goal

Table 5: Annotator scheme conflicts

Description	**Tokens**
Argument scheme based control codes	<from_consequence>, <from_source_authority>, <from_source_knowledge>, <goal_from_means/means_for_goal>, <rule_or_principle>
Argument stance based control codes	<pro>, <con>
Variable identifiers	<VAR_0>, <VAR_1>, <VAR_2>, <VAR_3>
Decoder BOS tokens	<pattern>, <argument>

Table 6: Special Tokens and Control Codes

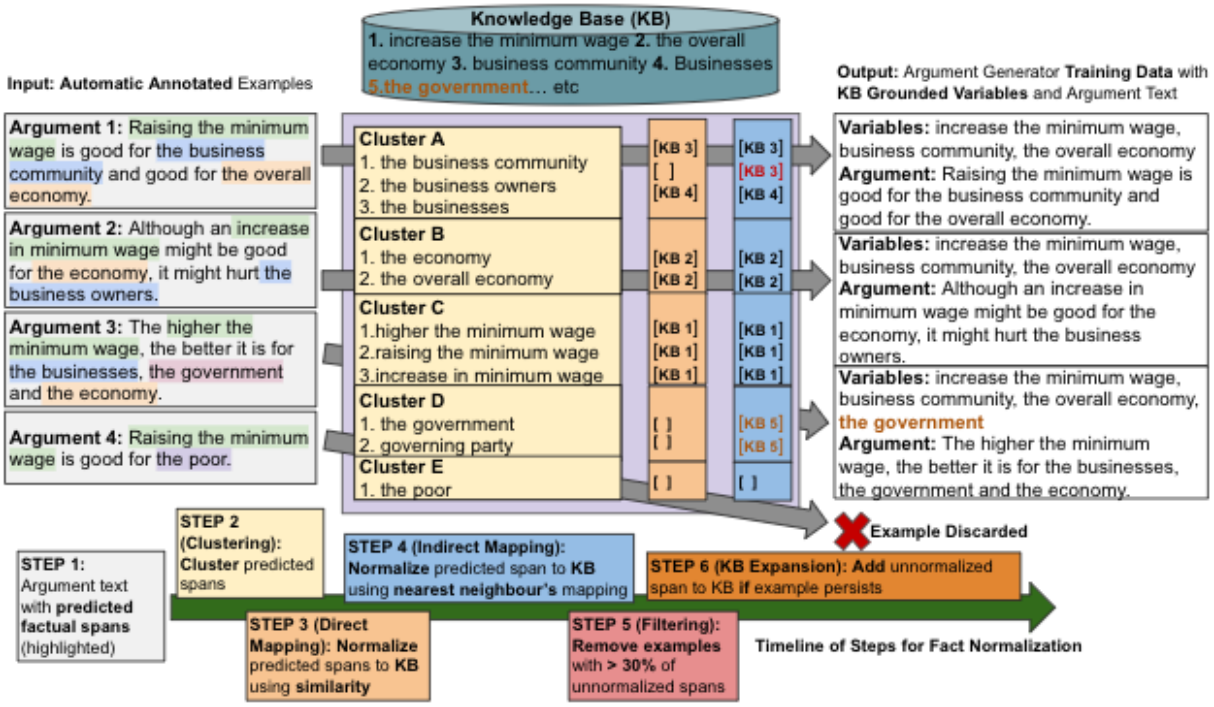

Figure 9: Phase 2 Dataset Fact Normalization Step.

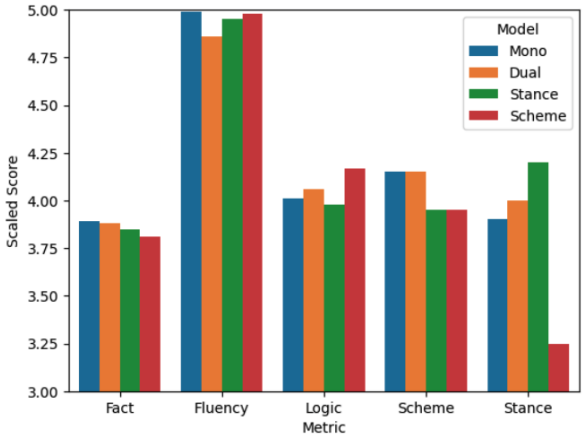

Figure 10: Metric-wise Model Comparison.

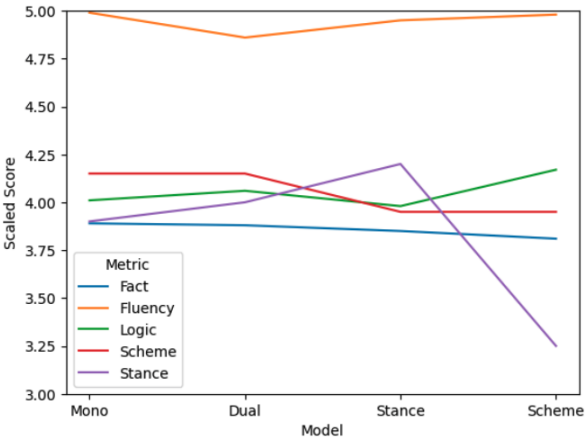

Figure 11: Model-wise metric Comparison.

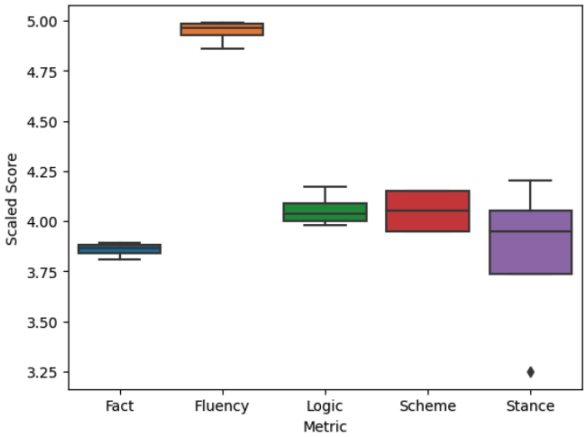

Figure 12: Metric-wise Score Distribution.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations Line 666

☑ A2. Did you discuss any potential risks of your work?
Ethics Statement Line 692

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Line 1-113

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

References. Line 704

☑ B1. Did you cite the creators of artifacts you used?
References. Line 704

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The research revolves around arguments from 6 sensitive topics, where some arguments might contain strong opinions. We clearly mention in the ethics statement that we do not share the opinions presented in the paper.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☑ Did you run computational experiments?

Section 3,4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3,4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3,4

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Section 3,4

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Section 3,4, Appendix

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Section 3,4

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Section 3,4

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.