Generating Explainable Product Comparisons for Online Shopping

Nikhita Vedula
Amazon, Seattle
veduln@amazon.com

Marcus Collins
Amazon, Seattle
collmr@amazon.com

Eugene Agichtein
Amazon, Seattle
eugeneag@amazon.com

Oleg Rokhlenko
Amazon, Seattle
olegro@amazon.com

ABSTRACT

An essential part of making shopping purchase decisions is to compare and contrast products based on key differentiating features, but doing this manually can be overwhelming. Prior methods offer limited product comparison capabilities, e.g., via pre-defined common attributes that may be difficult to understand, or irrelevant to a particular product or user. Automatically generating an informative, natural-sounding, and factually consistent comparative text for multiple product and attribute types is a challenging research problem. We describe HCPC (Human Centered Product Comparison), to tackle two kinds of comparisons for online shopping: (i) product-specific, to describe and compare products based on their key attributes; and (ii) attribute-specific comparisons, to compare similar products on a specific attribute. To ensure that comparison text is faithful to the input product data, we introduce a novel multi-decoder, multi-task generative language model. One decoder generates product comparison text, and a second one generates supportive, explanatory text in the form of product attribute names and values. The second task imitates a copy mechanism, improving the comparison generator, and its output is used to justify the factual accuracy of the generated comparison text, by training a factual consistency model to detect and correct errors in the generated comparative text. We release a new dataset[1] of 15K human generated sentences, comparing products on one or more attributes (the first such data we know of for product comparison). We demonstrate on this data that HCPC significantly outperforms strong baselines, by 10% using automatic metrics, and 5% using human evaluation.

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; *Supervised learning by classification*; **Multi-task learning**; *Transfer learning*; **Natural language processing**.

KEYWORDS

NLG, Product Comparison, Explainability, Fact Checking

[1]https://registry.opendata.aws/

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02.
https://doi.org/10.1145/3539597.3570489

ACM Reference Format:
Nikhita Vedula, Marcus Collins, Eugene Agichtein, and Oleg Rokhlenko. 2023. Generating Explainable Product Comparisons for Online Shopping. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570489

1 INTRODUCTION

Product comparison is a core part of shopping decision making [16, 21], improving both the quality and efficiency of purchase decisions among shoppers [16]. Consumers might seek out product comparisons out of interest, value consciousness, perfectionism, or quality [40]. The tremendous growth of online shopping and the advent of conversational shopping with virtual assistants [9] has overwhelmed shoppers with innumerable product choices. This has increased the cognitive load on customer memory [2], and the effort to go back and forth between products to understand and compare them, which may in turn increase buyer dissatisfaction [46]. Automatically generating informative, fluent, and explainable product comparisons for multiple product and attribute types for online shopping, is a challenging research problem. We must ensure that product comparison techniques can generalize across product categories with minimal manual effort, consider customer opinions, and scale to the diverse and massive data of modern marketplaces. First, we must select attributes or features to compare two or more products on [36]. E-commerce web sites contain a wealth of product attributes such as *brand* or *screen dimensions*. Detecting and prioritizing the right set of key attributes to compare products on can (i) help manufacturers understand which attributes to highlight in their web pages to best help customers; (ii) help customers navigate and refine their search on shopping websites; and (iii) educate customers about key considerations when they are shopping in unfamiliar product categories (*e.g.*, Electronics).

Second, a comparison of products on key attributes must be presented to users in a natural way, making it easy to understand why or how the products are comparable. One approach is to use comparison matrices or tables [2]. But requiring a user to understand and analyze large tables or to manually select and compare multiple product attributes can be taxing or infeasible due to human cognitive limitations [45]. Some e-commerce websites suggest '*similar products*' or '*other products bought by customers*', but do not explain to a user why those products might be similar or comparable.

Our proposed method HCPC (Human Centered Product Comparison) tackles several distinct research challenges to address the

above mentioned limitations: generating natural, explainable, and *human-centered* product comparisons; comparing attributes which aren't intrinsically ordered; and ensuring the factual correctness of the generated comparative text. HCPC performs two kinds of *explainable* product comparisons. First, it can compare two or more products by generating a natural language summary of these products' key features or 'highlights' (e.g. a printer in comparison to other printers). Second, HCPC can generate attribute-specific comparisons of two or more similar products. For instance, after an initial search for printers, customers may want to explore their options based on particular features, like *color printing* or *price*.

HCPC first detects salient or key *human-centered* product attributes based on their presence and sentiment in customer reviews, using an unsupervised (and hence scalable) method based on prior work [52]. We next extend the standard single-decoder natural language generation (NLG) model architecture to a multi-decoder transformer model, trained in a multi-task setting, for explainable comparative text generation. This novel multi-decoder approach has two goals: to enable the decoders to 'copy' specific text from the input data; and to generate additional output that can be used to explain the comparison text. Prior work has noted the benefits of natural language text over lists, tables or fixed rule-based templates, in terms of fluency, textual diversity and user satisfaction [5, 20, 55]. Unlike previous work [14, 26, 36], we use NLG to explain the presented product comparisons along specific product attributes, allowing customers to understand the comparison and decide if a product with those attributes meets their needs.

Comparing numerical product attributes (e.g. *price*) is intuitive, but it is less clear how to compare equally important non-numerical, *categorically*-valued attributes (e.g., *color*). We thus develop a notion of similarity based on semantic knowledge and customer search behavior to compare categorical attributes in an intuitive, weakly supervised (and therefore, scalable) manner. For the next research challenge, our multi-task, multi-decoder framework is naturally suited to confirm the factual correctness of the generated comparison text. We train HCPC's second decoder to output supporting text with attribute names and values, and also use this supporting text to train factual *error detection* and *error correction* models. This ensures that the generated comparison text is both factually consistent and explainable – which can be critically important to ensure users' trust in the system. We evaluate HCPC on a novel corpus of 15K human-generated diverse product comparison sentences. HCPC significantly outperforms strong baselines by 10% for comparative text generation using automatic evaluation metrics, and 5% using human evaluation. Thus, our main contributions are:

- We describe HCPC: a novel multi-task, multi-decoder approach to generate comparison text at both product and attribute levels. HCPC generalizes well across domains, avoiding model retraining for products or attributes unseen during training, and also scales to large product catalogues.
- HCPC's multi-decoder architecture generates supporting output text, which can be used to both *explain* the generated comparison text, and ensure that it is *factually consistent*.
- We describe a notion of similarity based on semantic knowledge and customer search behavior to compare categorical product attributes in an intuitive, weakly supervised manner.
- We make publicly available a novel corpus of 15K diverse, human-generated product comparison sentences.

2 RELATED WORK

Limited attention has been paid to the problem of product comparison for online shopping. Moraes et al. [36] reinforce the importance of comparing products on attributes such as price and quality, and studied their correlation with purchase decisions made by online shoppers. Hao et al. [14] propose a technique to recommend complementary products to online shoppers, but this does not directly compare similar products or generate natural language text for comparison. Le et al. [26] compare a pair of products on a given set of attributes. However unlike our work, their method only compares two products at a time, using a single fixed template to generate comparative text. Also unlike our work, none of the above methods are explainable. There are a very limited number of datasets available for product comparison research [19, 23]. They are derived from customer reviews, cover limited domains, are small in size, and rarely contain direct mentions of products, attributes or direct comparisons between products. We release a new dataset containing human generated text comparing diverse products.

Past work has proposed multi-task solutions for various NLG applications [6, 29, 60]. Architectures with multiple recurrent neural network decoders have been developed for unsupervised machine translation [25, 47] and generating distractor questions for reading comprehension [35]. We extend the single decoder architecture of the transformer language model, BART [27], to a multi-decoder setting in this work. Developing interpretable models that learn explanations for prediction tasks is another topic that has received attention in recent literature is [26, 28, 30]. Past work has used attention weights as a mechanism of providing explanations, but it has been observed that such explanations may not always be faithful to predictions [12, 17, 41]. Another line of work uses gradient based explanations in the form of generated token-level importance scores, to explain why a model makes a specific prediction for a given input [48, 51]. We adopt a generative approach for explanations, where HCPC's two decoders learn from different but related tasks: generating comparison text, and generating accompanying supportive, explanatory attribute-value information respectively. Our approach of generating comparative text bears some similarities to paraphrasing [32, 50, 57] and text prototype editing [11, 15, 20, 31]. But none of these have been framed in a multi-decoder setting for a combination of generating both natural language text, and supporting text to explain the generated text.

A key issue with NLG models is the detection and evaluation of factual inconsistencies such as missing information or hallucinations in the generated text [13, 18, 24, 37, 59]. Several techniques have been proposed to detect and correct different types of generation errors, such as grammatical or factual errors [11, 34, 39, 44, 53, 56]. Here, we use our generated supporting text to identify generated factually inconsistent sentences, and correct them conditioned on the product attribute-value information in a novel way.

3 THE HCPC FRAMEWORK

Given a set of attributes and their values associated with a set of products to be compared, our goal is to generate (i) natural sounding

text comparing a set of products faithfully based on their input attributes; and (ii) supporting text consisting of attribute names and values that can be used to justify or explain the generated comparative text. We generate two types of comparative text (see Figure 1): (i) product-level: describing or highlighting multiple key features of a product in comparison to other products; and (ii) attribute-level: comparing the set of products on the values of specific attributes common to them. We define an *Attribute* as a product feature, and each attribute takes a *Value*. For instance, the *attribute* 'brand' could have the *value* 'Nike'). We first detect the important or salient attributes associated with each product (Section 3.2); followed by learning two explainable NLG models in a multi-task setting to generate text for product-level and attribute-level comparisons (Sections 3.3 and 3.4). Finally, we learn models to detect and correct any factual inconsistencies in the generated text.

3.1 Data Collection

We collected a set of products from various categories and the catalog attributes associated with them, from the Amazon Product Reviews [38] dataset. To expand on the set of product catalog attributes, we added the names and ratings of product aspects separately rated by buyers, obtained by crawling `www.amazon.com`. We designed two separate annotation tasks, one each for descriptive product-level comparison and attribute-level product comparison. We recruited two native English-speaking human annotators, who are familiar with the domain of online shopping. For the first task, *Product Level Comparison*, we provided each annotator with a product and a list of the top 3 most *important* attributes or features. These features are derived from customer reviews and feedback, and are those most likely to help a shopper decide whether to purchase a product (based on [52] and described in Section 3.2). We then asked the annotators to write a natural language sentence that describes the product in terms of these key 'highlight' attributes. For the second task, *Attribute Level Comparison*, we constructed sets of 2-3 highly similar products [38] and provided them as input to the annotators, along with an attribute relevant to the product set. We asked the annotators to write a sentence comparing the values of that attribute for all the similar products in that set. For both tasks, we asked the annotators to write concise, fluent, easy to understand, grammatical, and stylistically diverse natural language sentences. We iteratively assessed random samples of text instances created by the human annotators, and revised our annotation guidelines accordingly, to ensure good quality data. To further ensure quality, a third annotator reviewed both sentences written for a given data instance, and chose the better one to keep in our corpus; based on grammar, faithfulness to the input, and fluency. Note that we do not obtain any comparative sentences from buyer-written product reviews, as done in the literature [19, 23], since they are often noisy, verbose, and can contain personal opinions or information irrelevant to the products being compared. Our curated dataset includes more than 15K product comparison sentences, covering about 8K products and more than 500 distinct numerical and categorical valued product attributes (Table 1). It has a high lexical diversity judging by the number of unique words in the vocabulary.

It is usually difficult for humans to assimilate and remember comparisons involving more than three items at a time [1, 2]. We thus restrict the number of products per comparison scenario to three or less in this work. However, HCPC can be extended to compare more than three products at a time.

Table 1: Our curated Product Comparison Corpus statistics.

Task Type	#Attributes	#Comparison Sentences	Avg.Text Length	#Unique Words
Product-Level	378	12K	25 words	66K
Attribute-Level	237	11K	19 words	39K

3.2 Unsupervised Key Attribute Selection

We use an unsupervised and domain agnostic technique, ReBARC [52], to identify attributes that are important to make shopping decisions about a product. ReBARC (Algorithm 1) uses two criteria from customer reviews: popularity, and customer opinions (sentiment), and also accounts for the seller identified attributes from the product title and catalog. Using this technique, HCPC ensures that attribute importance is evaluated based on direct and indirect mentions of product attributes by buyers in their reviews, as well as the (positive or negative) opinions of customers towards these attributes. The extracted key attributes are then used by HCPC in subsequent steps to generate product-level and attribute-level comparative text.

Algorithm 1 Select Unsupervised Key Attributes for Comparison

Require: For each product p, a set R of customer review sentences and set of attributes A (Section 3.1)

1: Extract a set of key phrases K from R with EmbedRank [3].
2: Choose the S review sentences containing a useful term or a product attribute. Let S_a be the set of sentences containing attribute $a \in A$. Append p's title to a sample of the S sentences.
3: Embed the S review sentences and A attributes with Sentence-BERT [43], to obtain e_s for $s \in S$ and e_a for $a \in A$.
4: Use Maximal Marginal Relevance (MMR) [4] to rank A on the cosine similarity of e_s with e_a, $\forall a, s$.
5: Choose the top 3 attributes per sentence. Total attributes = $3S$
6: Construct a ranked list L_1 as the top k most frequent attributes, out of the $3S$ ranked attributes.
7: Use a RoBERTa [33] model fine-tuned on the SST-2 sentiment detection benchmark [49], to compute a sentiment score for each review sentence S_a, $\forall a \in A$.
8: Average the absolute sentiment score values of all sentences S_a, to get the average sentiment score of a.
9: Re-rank L_1 using these aggregated sentiment scores to get the final ranked list L of the *top-k* attributes for each product.

3.3 Generating Product-Level Comparisons

3.3.1 Template-Guided Attribute Value Representation: Let $\mathcal{A} = \{a_i : v_i\}_{i=1}^{A}$ be a set of key attributes associated with a product P. a_i is the attribute name, and v_i denotes the value associated with a_i. We restrict the number of attribute-value pairs A to three or less, to reduce human cognitive load [1, 2]. Our goal is to translate $\mathcal{A}$ to a natural language text that is fluent, factually accurate, grammatical and stylistically diverse from other generated comparative texts.

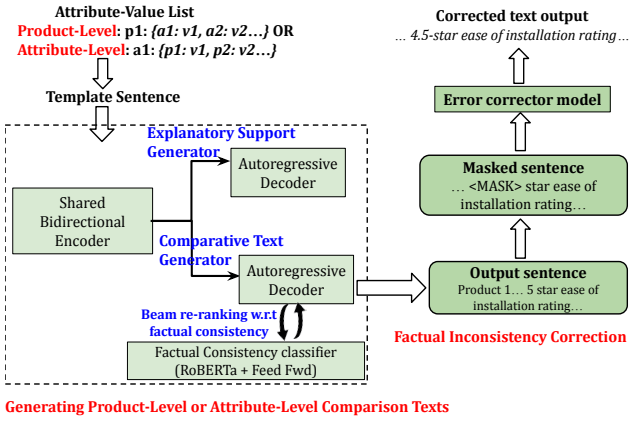

Figure 1: Model architectures for two types of comparisons (Sec. 3.3 and 3.4, examples in Table 7).

These generated textual descriptions of a set of products based on its key attributes can then be used to compare that set of attributes at a product-level, across multiple attributes (e.g., comparing printer A with printer B). One approach to do the above could be as a text infilling task, where we train a model to generate and insert tokens in between the list of attributes and values in $\mathcal{A}$ to connect them into a natural language sentence. However, such a formulation is more likely to produce grammatical or stylistic errors in the generated text, as we observe in Section 4. We thus frame our comparative text generation objective as a *template-guided natural language paraphrasing* task. We convert the attributes and their values for each product into simple, factually correct (possibly not fluent) sentences (see Figure 1), using a minimal set of manually defined templates (see examples in Table 7). These simple templated representations of $\mathcal{A} = \{a_i : v_i\}_{i=1}^{A}$ are then input to our proposed NLG model to be rewritten into accurate, coherent text.

3.3.2 Multi-decoder Transformer: Figure 1 shows an overview of the NLG model we developed, that utilizes the key product attributes to generate product-level comparison sentences. While template-guided, the key novelty of our model is its transformer-based multi-decoder model architecture. A multi-decoder model consists of a shared encoder, which transforms input data of different types or domains into a common latent space, and two or more decoders each dedicated to generate valid output text for a single domain or output task from the encoded representation.

One HCPC decoder generates comparative text, by paraphrasing the input template utterances into fluent and accurate output sentences. The second decoder learns an auxiliary task of generating *explainable support* text for the output of the first decoder, consisting of the key products and attributes regenerated from the input template sentence. These can be used to justify or explain the factual accuracy of the generated comparison text. For instance, if a set of three products are compared on their customer ratings and the comparison text decoder generates *'Product 2 is rated 1-star higher than Products 1 and 3'*, this statement can be explained by the support generator decoder output *'product 1_4 stars, product 2_5 stars, product 3_4 stars'*. Here '4 stars', '5 stars' and '4 stars' are the rating values of products 1, 2, and 3 respectively. Incorporating this task in our framework guides the shared encoder to focus on generating the correct product and attribute information from the template sentence – in a loose sense, teaching the model what to copy. We choose BART [27] for our generative model, because of its utilization of a denoising autoencoder, and the token span masking objective that it has been pre-trained on, that are in line with both the tasks proposed in our work to generate text for product comparison. This generated support can also help us identify and correct potentially incorrect entities (Section 3.3.5). We train our model with parallel sentences pairs: a template sentence and its fluent, grammatical version written as detailed in Section 3.1. We optimize a loss function that combines the negative log likelihood generation loss $\mathcal{L}_p$ of the support generator, and $\mathcal{L}_c$ of the comparison text generator, via a weighting hyper-parameter $\alpha \in \{0, 1\}$.

$$\mathcal{L}_p = \sum_{i=1}^{N_s} \log P(p_j | p_{1:j-1}, x, \theta) \; ; \; \mathcal{L}_c = \sum_{i=1}^{N_c} \log P(y_j | y_{1:j-1}, x, \theta)$$

$$\mathcal{L} = \alpha \mathcal{L}_p + (1 - \alpha)\mathcal{L}_c$$

where x is the input template utterance, y is the output rewritten text generated by the comparison text decoder, p is the support sequence of attribute names and values generated by the support decoder, and θ denotes the model parameters.

3.3.3 Detecting Factual Inconsistencies: The BART-based NLG models that we use may generate *factually inconsistent* text that is not faithful to the model input x. We thus train a classifier to predict whether the text generated by our multi-decoder model is factually consistent or not. The positive examples for our classifier are the factually consistent sentences s obtained from our annotated data (Section 3.1). For every sentence s, we define the entity set $\mathcal{E}$ associated with it as all the products, attributes, and values that occur in s; as well as all the named entities present in s. We analyzed model-generated factual errors, to derive a set of heuristics we then use to corrupt sentences s into a factually inconsistent sentences s':

- Duplicate a random entity $e \in s$ by re-inserting e in s.
- Remove a random word from a multi-word entity $e \in s$.
- Delete a random entity e from s.
- Assume s describes product p; randomly choose an entity e of type product name, attribute name, or attribute value. Find an entity $e' \notin s$, s.t. type(e) = type(e') belonging to a product p' similar to p (using data on similar products from Section 3.1). Then either (i) insert e' into s; or (ii) replace e with e' in s.
- Invert the sentiment of s by inserting or removing negation words such as *not* before certain types of entities.
- Identify comparative or superlative words $w \in s$ and replace them with antonyms. E.g., replace *higher* with *lower*.
- Replace a random numerical value $e \in s$ with another value numerically close to e.
- Replace a random numerical measurement unit e in s with another related measurement unit e'.

We label sentences containing irrelevant, unsupported or incorrect information with respect to input product data as 'factually inconsistent'. Thus, changing product attributes/values in s or randomly modifying using our heuristics will likely make it inconsistent. Using the positive examples from our original corpus, and the

negative examples generated using our heuristics, we train a factual inconsistency detection classifier consisting of a feed forward layer on top of a RoBERTa encoder, using a cross entropy loss objective.

3.3.4 Diverse and Factual Comparative Text Generation: We execute a separate beam search for both our NLG decoders. We use traditional beam search for the support generation decoder. However, for the comparison text decoder we use the diverse beam search (DBS) algorithm [54] to encourage greater variation in the generated text. DBS augments the original beam search objective with an additional dissimilarity term that measures and optimizes for the diversity between candidate sequences at each step of the search process. We additionally re-rank the DBS candidates present in the beam after generation, using our factual consistency detection model described in Section 3.3.3. That is, a potential output sequence predicted as factually consistent by our detection model receives a higher rank in the beam than a sequence predicted as factually inconsistent. Ties are resolved using the original language model log-likelihood scores. Thus we ensure our final generated comparison text is both factually accurate and stylistically diverse from the other generated instances.

3.3.5 Correcting Factual Inconsistencies: It is possible that the best ranked sequence after performing beam search (Section 3.3.4) still contains some factual inconsistencies. We correct these inconsistencies as a post-processing step by building a model based on the T5 [42] language model architecture. Our model edits potentially incorrect spans of input text, so that the generated rewrite is consistent with respect to the original input text. To generate training data for our error correction model, for each human annotated product comparison sentence s, we have available a factually accurate, its template t (Section 3.3.1), and factually inconsistent sentences s' (Section 3.3.3). We identify and mask (replace with a [MASK] placeholder token) the noisy or corrupted tokens that we artificially injected within s', hence creating a masked sentence s''. Thus, token masking approximates inconsistent tokens that need to be corrected. If there is no noisy token or we do not know which tokens are noisy in s', we mask as potentially incorrect entities a subset of entity tokens in s' that are not present in its template version t, or in its accompanying explanatory support. We now have a corpus whose instances each contain an original, factually consistent sentence s, its template version t, and the masked sentence s''. We then train a T5-based text correction model that concatenates and jointly encodes the masked claim s'' and template t inputs, to generate corrections by replacing the mask placeholders. Similar to masked language modeling, the training objective here is to generate a factually consistent sentence s conditioned on the masked claim s'' and template t.

3.4 Generating Attribute-Level Comparisons

Along with generating product-level comparative text, HCPC also aims to generate attribute-level product comparison texts along with their accompanying explanatory supports. Assume that a set of products $\{p_i{}_{i=1}^{P}\}$ are being compared on their values of an attribute a. A simple, manually designed template is first assigned to this group of products and their attribute values. We train a generative model according to the steps described in Section 3.3, and Figure 1. As before, the model uses our template-guided multi-decoder architecture based on BART, trained in a multi-task setting. The comparison text decoder learns text comparing attribute values of the products under consideration, while the support generation decoder generates supporting attribute-value information that can be used to explain the comparative text output (see Table 7 for examples of both types of comparisons).

3.4.1 Comparing Categorical Product Attributes. For numerically valued attributes, we can directly compute value based statistics such as the maximum, minimum, difference, or range, which can be used to generate attribute-level product comparison text. However, it is not easy to compare attributes that lack numerical values without involving external knowledge, personal biases or preferences. We find that nearly 50% of the key attributes identified in Section 3.2 are categorically-valued. We propose to compare categorical product attribute values using a novel, weakly supervised method. It incorporates attribute similarity based on semantic, commonsense knowledge [8] as well as customer search behavior. For such attributes, we learn a relative ordering instead of an absolute one, i.e., is value 'a' of a categorical attribute is more similar to values 'b' or 'c'? HCPC can thus compare categorical attributes which have some notion of "value-based similarity" (e.g. the color 'red' is more similar to 'pink' than to 'black'). We only consider those categorical attributes whose values can be compared using typical commonsense knowledge, in the absence of personal opinions or biases (e.g. *color, flavor*). We omit attributes that cannot be easily compared without personal opinions (e.g. *brand* of a product).

We first use weak supervision to obtain data regarding *similar* categorical attribute values for a set of products. For this, we collect customer search interactions from the logs of a large online shopping engine, and identify the product attribute filters applied by customers as they refined their search. We make the (weak) assumption that during a single search session for a given search query, the different values customers select for the same categorical attribute to refine their search are 'similar' to each other. We construct a set of training, validation and test instances by concatenating the original search query of the customer, and the name and value of the attribute used by them during their search session. As in Section 3.2, we use the Sentence-BERT model to encode each instance. Finally, we train a binary classifier on this weakly labeled data, to identify if a given pair of categorical attribute values are likely to be similar to each other or not. This approach can be used to generate comparative text template statements such as '*Color of Product 1 and Product 2 is similar*', which would then be paraphrased into a more fluent sentence by HCPC's multi-decoder model.

3.4.2 Summary. HCPC's goal is to generate natural-sounding, factually accurate text to compare products with a novel, multi-task, multi-decoder NLG model. Common ways to compare products across various e-commerce systems include comparison matrices, tables or lists of attribute names and their values. But as outlined in Sections 1 and 2, these methods do not scale or generalize well across product domains, suffer a higher human cognitive load (e.g., to understand large comparison tables), and are not explainable, natural or easy to understand in an online shopping setting. HCPC seeks to address these challenges. By design, HCPC generalizes well across product domains, avoiding the need for model re-training

Table 2: Generating product-level comparisons. † shows a significant ($p < 0.01$) improvement of our proposed method over all baselines. Best results are in bold. 'Multi-dec' indicates the use of our multi-decoder model architecture.

Method	BLEU	ROUGE	BERT-score	Grammatical	DC
$COMPARE_{ER}$ [26]	0.33	0.4	0.48	65%	0.81
CTRL [22]	0.48	0.6	0.65	70%	2.1
PEGASUS [57]	0.5	0.63	0.68	68%	1.76
Single decoder (BART [27])	0.57	0.7	0.75	68%	2.08
Single decoder (T5 [42])	0.54	0.62	0.68	63%	2.15
Text infilling (BART [27])	0.46	0.53	0.59	52%	2.1
HCPC-templateOnly	0.49	0.56	0.6	70%	1.03
HCPC-GPT2 (multi-dec)	0.54	0.62	0.7	55%	2.05
HCPC-T5 (multi-dec)	0.59	0.71	0.77	70%	**2.33**
HCPC-noDBS (multi-dec)	0.54	0.64	0.71	68%	2.08
HCPC-noFIC (multi-dec)	0.52	0.62	0.7	64%	1.95
HCPC (ours, multi-dec)	**0.66**†	**0.77**†	**0.81**†	**74.1%**†	2.23

Table 3: Generating attribute-level product comparisons. † shows a significant ($p < 0.01$) improvement of our proposed method over all baselines. Best results are in bold. 'Multi-dec' indicates a multi-decoder model architecture.

Method	BLEU	ROUGE	BERT-score	Grammatical	DC
$COMPARE_{ER}$ [26]	0.35	0.42	0.46	63%	0.66
CTRL [22]	0.42	0.48	0.53	66%	1.63
PEGASUS [57]	0.45	0.51	0.57	63 %	1.6
Single decoder (BART [27])	0.49	0.55	0.6	55%	1.7
Single decoder (T5 [42])	0.43	0.5	0.55	60%	1.7
Text infilling (BART [27])	0.42	0.52	0.57	55%	1.6
HCPC-templateOnly	0.4	0.47	0.55	70%	1.04
HCPC-GPT2 (multi-dec)	0.46	0.52	0.58	60%	1.5
HCPC-T5 (multi-dec)	0.51	0.56	0.6	71%	1.63
HCPC-noDBS (multi-dec)	0.48	0.54	0.59	65%	1.55
HCPC-noFIC (multi-dec)	0.46	0.54	0.58	66%	1.45
HCPC (ours, multi-dec)	**0.577**†	**0.65**†	**0.69**†	**77.8%**†	**1.75**

to handle products or attributes unseen during training. HCPC's key-attribute selection component is unsupervised, its categorical attribute-level comparison component is weakly supervised, and its supervised NLG components are domain-agnostic, enabling HCPC to scale to large product catalogues. HCPC's second support-generation decoder can help its comparative text decoder focus on generating the correct attribute values, can be used to identify incorrectly generated entity tokens, can explain the factual correctness of the generated comparative text, and can be used to train an error correction model to edit incorrectly generated product comparison text. We also create and plan to release a novel corpus of 15K diverse, human-generated product comparison sentences.

4 EVALUATION

4.1 Experimental Setup and Results

We evaluate HCPC on the human-generated data described in Section 3.1, which we will release publicly. We train all models on 80% of the data, leaving 10% each for validation and testing. We run our experiments three times with different random seeds to smooth out variance, and fine-tune all hyperparameters based on validation set performance, with human annotations as ground truth (Section 3.1). We fine-tune a BART-base encoder-decoder model for NLG (12 layers, 140M parameters) and a t5-base model for error correction (12 layers, 220M parameters), for 10 epochs with a learning rate of 5×10^{-5} and early stopping, and a RoBERTa-large model (355M parameters) for factual inconsistency detection.

4.1.1 Evaluating Comparative Text and Support Generation: We compare the performance of the multi-decoder generative model components of HCPC against the following baselines, using the automatic NLG metrics BLEU, ROUGE and BERT-score [58]. We also automatically evaluate grammatical accuracy using a recent unsupervised grammatical error correction approach [56]. We evaluate how fluent or natural sounding the generated text is using the Dale-Chall readability index (DC) [7]:

(i) $COMPARE_{ER}$ [26] uses a fixed template to compare a pair of products: *[product 1] is better at [attribute 1] than [product 2], but worse at [attribute 2]*. We modify this template for product-level comparisons in Table 2, and for multiple-product, attribute-level comparison in Table 3.

(ii) CTRL [22] is a language model for controlled text generation conditioned by specific text. We concatenate the task name (*product* or *attribute*) along with the respective attribute-value and product information ($\{a_i : v_i\}_{i=1}^{A}$ for product-level comparison or $\{p_i : v_i\}_{i=1}^{P}$ for attribute-level comparison), and train a CTRL model to generate a relevant sentence expressing this information.

(iii) PEGASUS [57] is a language model for abstractive text generation, trained using self-supervised pre-training tasks.

(iv) Single-decoder (BART, T5): Instead of a multi-decoder setup, we remove the second explainable support decoder, and keep only the comparison decoder, based on either BART or T5.

(v) Text infilling (BART): Instead of learning a model to paraphrase an input template sentence, we train a BART model to generate and insert tokens in between the list of products, attributes and values to connect them into a natural language sentence.

(vi) HCPC-templateOnly: the template comparative sentence. (Section 3.3.1) is used as the comparison without further processing.

(vii) HCPC-GPT2 and HCPC-T5: Variants of HCPC that substitute the BART decoders with GPT2 and T5.

(viii) HCPC-noDBS: a variant of HCPC using traditional beam search instead of diverse beam search.

(ix) HCPC-noFIC: a variant of HCPC without our proposed error detection and correction mechanism.

Tables 2 and 3 show that HCPC outperforms all baselines significantly, by at least 7% in terms of BLEU, ROUGE and BERT-score. Overall, more than 74% of the product comparison sentences generated by HCPC for both product-level and attribute-level comparisons have been judged as grammatically accurate (fourth column of Tables 2 and 3). The last column measures the readability of the generated sentences (a higher DC indicates a higher textual complexity). The fourth and fifth rows of Tables 2 and 3 show that BART and T5 models when trained with a single decoder are outperformed by HCPC's multi-decoder setup by at least 7% for product-level and 9% for attribute-level comparisons, across all metrics. This reinforces the effectiveness of our multi-decoder architecture. We also found HCPC's explainability support decoder to obtain a BLEU score of 0.73 and 0.64, and a ROUGE score of 0.85 and 0.72 respectively, for the two tasks of product-level and attribute-level comparison (not shown in Tables 2 and 3 due to space constraints). This is at least 10% higher than the corresponding multi-decoder baselines.

We next asked three human annotators to rate the quality of 200 randomly chosen generated product-level and attribute-level comparative text and their justifying support (if applicable), generated

Table 4: Human evaluation of the generated comparative sentences and support quality. † indicates a significant difference between the results of HCPC (ours) and PEGASUS.

Method	Fluency	Factual Consistency	Grammar	Support quality	Support usefulness
PEGASUS [57]	2.12	2.04	2.03	N/A	N/A
HCPC-T5	2.41	2.33	2.21	2.44	2.49
HCPC (ours)	2.56†	2.49†	2.46†	2.58	2.63

by two baseline approaches and our proposed method. Annotators rated the text between 1 (bad) and 3 (good) for fluency, grammatical correctness in English and factual consistency with respect to the input product and its attributes. We also asked annotators to rate the quality of the generated support, and whether it could be used to usefully explain its accompanying product comparison sentence. Table 4 shows that HCPC substantially outperforms the two baselines across all metrics (inter-annotator agreement Fleiss' $\kappa = 0.71$). Human evaluators prefer HCPC's fluency, factuality, and grammar over baselines in 67% of the instances. The last two columns show that annotators found the support generated by HCPC to have good quality and be beneficial to justify the accompanying product comparison text. We further observe significant improvements to specific types of comparisons. For attribute-level comparison sentences involving numerical differences (e.g., of the form *higher than*, *most expensive* etc), annotators prefer sentences generated by HCPC to the baselines across all metrics.

4.1.2 Evaluating Factual Inconsistency: An essential component of HCPC is to determine whether the generated comparative sentence output is factually consistent or not with the original input product data, without loss of any relevant information or hallucinating any additional data. Table 4 shows that, according to annotators, HCPC substantially outperforms strong baselines in terms of factual consistency. We compare our factual inconsistency detection to an existing, high performing approach FactCC [24] in Table 5, which shows the percentage of "missed" inconsistent sentences (the false negative rate), and the average human factual consistency ratings from 1 (bad) to 3 (good) to a sample of 200 generated instances.

Table 5 also lists two metrics proposed in the literature to measure entity-level consistency of the generated text: precision-target (prec_t), and recall-target (rec_t) [37]. $prec_t = \frac{\mathcal{N}(h \cap t)}{\mathcal{N}(h)}$, and $rec_t = \frac{\mathcal{N}(h \cap t)}{\mathcal{N}(t)}$, where $\mathcal{N}(t)$ is the number of named-entities in the ground truth reference text and $\mathcal{N}(h \cap t)$ is the number of named-entities in the generated text that are also present in the ground truth reference text. The F1-score $F1_t$ is then computed as the harmonic mean of $prec_t$ and rec_t. We observe in Table 5 that our method misses detecting 14% of factually inconsistent product comparisons, much better than the FactCC baseline (23%) used in conjunction with our multi-decoder setup. Our proposed approach for factual inconsistency detection achieves an improvement of 6.5% for the entity-level metric $F1_t$. The last column shows human annotators' ratings for a sample of 200 generated instances, where HCPC significantly improves over FactCC in factual inconsistency detection.

4.1.3 Evaluating Similar Categorical Attribute Values: As noted earlier, nearly 50% of the key attributes identified by HCPC are categorically-valued. We evaluate our approach to compare categorical product attribute values with a human evaluation task on 233 instances. We presented a group of three potential values (a, b, c) for a given attributes to the human annotators, and asked them to find whether a was more similar to b or c for that attribute. We compared the human annotations to the output of our proposed approach for comparing values of categorical attributes. Table 6 shows that our model correctly predicts a candidate attribute value a being similar to another value b or c for the same attribute about 85% of the time. There is a 60%-40% split between the similar values predicted by our method, when both values are annotated as equally similar to the candidate attribute value a. We observe similar performance trends when individually focusing on specific attributes, e.g. *color*. For example, more than 72% of the time, HCPC correctly predicts that the value, b or c that humans say is most similar to a.

Table 5: Evaluating the factual consistency of the text generated by the comparative text decoder. † shows a statistically significant improvement of our approach over FactCC.

Method	Missed	$\text{Precision}_\text{t}$	Recall_t	F1_t	Human (sample of 200)
HCPC-FactCC	23%	0.74	0.702	0.72	2.28
HCPC (ours)	14%†	0.85†	0.73†	0.785†	2.49†

Table 6: Human (column headers) and model (row labels) similarity judgements confusion matrices. First two rows: for all attributes. Last two rows: for the attribute 'color'.

	a **sim. to** b	a **sim. to** c	a **sim. to** b **&** c
a **sim. to** b	0.86	0.16	0.6
a **sim. to** c	0.13	0.84	0.4
a **sim. to** b **(color)**	0.76	0.28	0.61
a **sim. to** c **(color)**	0.24	0.72	0.39

4.2 Discussion

4.2.1 Explainability: The second decoder of our novel multi-decoder transformer architecture generates supportive text justifying the first decoder's comparison text. As shown by the fourth, fifth and last rows of Tables 2 and 3, this supportive task can work as an auxiliary function to improve the performance of our comparison text decoder, by guiding it to attend to the correct attribute values. The generated supportive text may be more readable, *e.g.*, expanded numerical units or shortened attribute names. We can further use this support to identify and mask potentially erroneous entities and try to correct them (Section 3.3.5). For example, given the attribute list of a product as *{'size': '1tb solid state drive', 'battery life rating': '4.29', 'touch screen rating': '3.25'}*; the generated support is *'size_1 TB, battery life rating_4.29 stars, touch screen rating_3.25 stars'*. Another key function of the generated support is to *explain* or *justify* whether our generated comparative text is factually consistent. For example, the generated support (e.g. *'product 1_1 TB, product 2_1 TB, product 3_10 GB'*) can explain the factual accuracy of this attribute-level comparison sentence (e.g. '*Products 1 and 2 have a higher disk storage size than Product 3*').

We next evaluate the explainability of the HCPC-generated supports. We manually tag product names, attribute names and attribute values that can be valid supporting text, for a random sample

Table 7: Examples of product-level and attribute-level comparison text utterances generated by different models, along with their attribute-value information, their template-based representation and the human annotator output.

Model	Generated Text Sequence
Attr-Value Info	*easy to install rating: 4.5 stars, really like picture quality, value for money rating 4 stars (Product-Level Comparison)*
Template	The great things about Product 1 compared to Product 2 and Product 3 are it's easy to install rating 4.5 stars, really like picture quality and value for money rating 4 stars.
Reference (human)	According to reviews, product 1 appeals to customers over products 2 and 3 because it has great value for money, high picture quality and is easy to install.
PEGASUS [57]	Product 1 has an easy to install rating of 4.5 stars, really like picture quality and value for money, compared to Product 2 and Product 3.
HCPC-T5	Product 1 is favorable over Product 2 and Product 3 because of it's 4 star value for money rating, picture quality and 4.5 star easy to install rating.
HCPC (ours)	Product 1 has a high value for money, along with great picture quality and a 4.5-star ease of installation rating which makes it favorable over Products 2 & 3.
Attr-Value Info	*max copy speed black white_product 1: 30 ppm, _product 2: 20 ppm, _product 3: 30 ppm (Attribute-Level Comparison)*
Template	Customers like the max copy speed black white of product 1 and product 3 which is 10 ppm higher than Product 2.
Reference (human)	Customers are pleased with the maximum black and white copying speed of Products 1 and 3 because it is 10 ppm faster than Product 2.
PEGASUS [57]	The max black and white copy speed of product 1 and product 3 is 10 ppm higher than Product 2.
HCPC-T5	Customers like the max black and white copy speed of Products 1 and 3, which is 10.0 ppm higher than Product 2.
HCPC (ours)	Customers really like the maximum black and white copying speed of Products 1 and 3, which is 10.0 ppm faster than Product 2.
Attr-Value Info	*assembly not required_product 1: False, _product 2: True, _product 3: True (Attribute-Level Comparison)*
Template	Product 1 is less desirable than Product 2 and Product 3 because of assembly not required False.
Reference (human)	Product 1 is less desirable that products 2 and 3 because it has to be assembled.
PEGASUS [57]	Product 1 is less desirable than Product 2 and Product 3 because they don't have to be assembled.
HCPC-T5	Product 1 is less desirable than Products 2 and 3 because assembly is not required.
HCPC (ours)	Product 1 is less desirable than Products 2 and 3 because it does not require assembly.

of 200 comparative text instances generated by HCPC. We measure the generated explanatory support quality with respect to this human annotated ground truth, using the metrics [10]: (i) plausibility (F1 score of the tokens present in the model generated support with respect to the true support); (ii) sufficiency (does the model generated support contain all information needed to explain and assess the factual correctness of the generated text); and (iii) compactness (do all tokens of the model generated support explain or assess the factuality of the generated comparative text). We obtain 72%, 79% and 76% respectively for the above three metrics on our human annotated sample of support instances generated by HCPC.

4.2.2 Scalability and Adaptation to New Domains: Scalability is an important challenge, given the fast growing number of product and attribute domains in the field of e-commerce. We believe HCPC can scale well because its key-attribute selection component is unsupervised, its attribute-level comparison component for categorical values is weakly supervised, and its supervised NLG components are domain-agnostic. With about 1000 unique product attributes across multiple product categories, our models described in Sections 3.3 and 3.4 can be trained in a couple of hours. Our multi-decoder model inference time is only 5 milliseconds higher than the inference time with a single decoder language model. To demonstrate that our supervised NLG components are domain agnostic, we test them in a zero-shot setting, by constructing training and testing data which have disjoint product categories and attributes. We then evaluate HCPC and the baselines as before, but now on unseen product categories and attributes. We observe that the template-based baselines, i.e. $COMPARE_{ER}$ and *HCPC-templateOnly* suffer the largest drop in performance relative to the original setting (all categories and attributes seen in training). All language model baselines slightly decrease in performance across metrics. Even in this out-of-domain setting, HCPC significantly outperforms baselines, as in the in-domain setting (Tables 2 and 3). Thus, HCPC successfully generates product comparisons for previously unseen product categories and attribute types. Our template-guided input scheme offers effective generalization capabilities, where minimal additional annotations are required for new product or attribute domains.

4.2.3 Qualitative Analysis: We assume in this work that all required attribute-value information is available for each product being compared. Table 7 shows examples generated by our model and baseline approaches. Overall, HCPC generates comparative sentences of good quality, fluency and grammatical accuracy, in comparison to other baselines. Based on human annotation of errors made by HCPC's multi-decoder model, about 21% are related to grammar, fluency or style, and the rest are related to content. HCPC's error detection and correction (Section 3.3.5) correctly handles many errors, such as removing hallucinated words not found in any input data; or replacing incorrect numbers or entities with the correct ones. However, there remain some errors related to logic or arithmetic operations that are not very well handled by our current approach. For instance, the last set of examples in Table 7 compare the '*assembly not required*' attribute. None of the models correctly handle the logical 'double negative' case when this attribute is *False*. In addition, sometimes the explanatory support generated by our HCPC contains extra, irrelevant information. For example, for the comparative text *'Product 1 is more expensive than Product 2'*, HCPC generates the support *'product 1_30$, product 2_10$, product 3_50$'*. The span *'product 3_50$'* is not required to explain or assess the factuality of the comparative text.

5 CONCLUSIONS

We described a novel technique called HCPC for product comparison, that consists of a multi-decoder transformer language model and a factual consistency detection and correction model. It identifies key attributes for products using *unsupervised* methods for scale; and generates explainable product level and attribute level comparison text. HCPC is domain-agnostic, and can scale to large product catalogues with minimal modelling effort. Finally, we compile and make available a novel, first of its kind, human-annotated dataset for product comparison that consists of 20K sentences comparing a set of 2-3 products. In future, we would like to contextualize or personalize product comparison by incorporating past customer behavior or conversational context. We would also like to improve the logical consistency of our NLG models.

REFERENCES

[1] G. Alvarez and P. Cavanagh. 2004. The capacity of visual short-term memory is set both by visual information load and by number of objects. *Psychological science* (2004).

[2] J. Álvarez Márquez and J. Ziegler. 2020. In-Store Augmented Reality-Enabled Product Comparison and Recommendation. In *ACM RecSys*.

[3] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint arXiv:1801.04470* (2018).

[4] J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *ACM SIGIR*.

[5] A. Challa, K. Upasani, A. Balakrishnan, and R. Subba. 2019. Generate, Filter, and Rank: Grammaticality Classification for Production-Ready NLG Systems. *arXiv preprint arXiv:1904.03279* (2019).

[6] S. Chen, Y. Zhang, and Q. Yang. 2021. Multi-task learning in natural language processing: An overview. *arXiv preprint arXiv:2109.09138* (2021).

[7] E. Dale and J.S. Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin* (1948).

[8] E. Davis. 2014. *Representations of commonsense knowledge*. Morgan Kaufmann.

[9] Statista Research Department. 2014-2023. Global retail e-commerce sales. https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

[10] J. DeYoung, S. Jain, N. Fatema Rajani, E. Lehman, C. Xiong, R. Socher, and B. Wallace. 2019. ERASER: A benchmark to evaluate rationalized NLP models. *arXiv preprint arXiv:1911.03429* (2019).

[11] A. Elgohary, C. Meek, M. Richardson, A. Fourney, G. Ramos, and A.H. Awadallah. 2021. NL-EDIT: Correcting semantic parse errors through natural language interaction. *arXiv:2103.14540* (2021).

[12] B. Gino, L. Yang, P. Damian, R. Oliver, C. Massimiliano, W. Roger, et al. 2020. On identifiability in transformers. In *International Conference on Learning Representations*.

[13] Z. Guo, M. Schlichtkrull, and A. Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics* (2022).

[14] J. Hao, T. Zhao, J. Li, X. Dong, C. Faloutsos, Y. Sun, and W. Wang. 2020. P-Companion: A principled framework for diversified complementary product recommendation. In *ACM CIKM*.

[15] N. Hossain, M. Ghazvininejad, and L. Zettlemoyer. 2020. Simple and effective retrieve-edit-rerank text generation. In *ACL*.

[16] G. Häubl and V. Trifts. 2000. Consumer Decision Making in Online Shopping Environments: The Effects of Interactive Decision Aids. *Marketing Science* (2000).

[17] S. Jain, S. Wiegreffe, Y. Pinter, and B. Wallace. 2020. Learning to Faithfully Rationalize by Construction. In *Association for Computational Linguistics*.

[18] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung. 2022. Survey of hallucination in natural language generation. *arXiv preprint arXiv:2202.03629* (2022).

[19] N. Jindal and B. Liu. 2006. Identifying comparative sentences in text documents. In *ACM SIGIR*.

[20] M. Kale and A. Rastogi. 2020. Template guided text generation for task-oriented dialogue. *arXiv preprint arXiv:2004.15006* (2020).

[21] A. Kamis. 2006. Search strategies in shopping engines: An experimental investigation. *IJEC* (2006).

[22] N. Keskar, B. McCann, L. Varshney, C. Xiong, and R. Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858* (2019).

[23] W. Kessler and J. Kuhn. 2014. A Corpus of Comparisons in Product Reviews. In *LREC*.

[24] W. Kryściński, B. McCann, C. Xiong, and R. Socher. 2020. Evaluating the factual consistency of abstractive text summarization. *EMNLP* (2020).

[25] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. *ICLR* (2018).

[26] T. Le and H. Lauw. 2021. Explainable Recommendation with Comparative Constraints on Product Aspects. In *ACM WSDM*.

[27] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, et al. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[28] Y. Li and K. Yao. 2021. Interpretable nlg for task-oriented dialogue systems with heterogeneous rendering machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[29] Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, et al. 2020. Glge: A new general language generation evaluation benchmark. *arXiv preprint arXiv:2011.11928* (2020).

[30] H. Liu, Q. Yin, and W.Y. Wang. 2019. Towards explainable NLP: A generative explanation framework for text classification. *ACL* (2019).

[31] X. Liu, D. Liu, B. Yang, H. Zhang, J. Ding, W. Yao, W. Luo, H. Zhang, and J. Su. 2022. KGR4: Retrieval, Retrospect, Refine and Rethink for Commonsense Generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[32] X. Liu, L. Mou, F. Meng, H. Zhou, J. Zhou, and S. Song. 2019. Unsupervised paraphrasing by simulated annealing. *arXiv preprint arXiv:1909.03588* (2019).

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, et al. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[34] J. Mallinson, J. Adamek, E. Malmi, and A. Severyn. 2022. EdiT5: Semi-Autoregressive Text-Editing with T5 Warm-Start. *arXiv preprint arXiv:2205.12209* (2022).

[35] K. Maurya and M. Desarkar. 2020. Learning to distract: A hierarchical multi-decoder network for automated generation of long distractors for multiple-choice questions for reading comprehension. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

[36] F. Moraes, J. Yang, R. Zhang, and V. Murdock. 2020. The role of attributes in product quality comparisons. In *CHIIR*.

[37] F. Nan, R. Nallapati, Z. Wang, C. Santos, H. Zhu, D. Zhang, K. McKeown, and B. Xiang. 2021. Entity-level Factual Consistency of Abstractive Text Summarization. *arXiv preprint arXiv:2102.09130* (2021).

[38] J. Ni, J. Li, and J. McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*.

[39] S. Panthaplackel, M. Allamanis, and M. Brockschmidt. 2021. Copy that! editing sequences by copying spans. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[40] Y. Park and U. Gretzel. 2010. Influence of consumers' online decision-making style on comparison shopping proneness and perceived usefulness of comparison shopping tools. *JCER* (2010).

[41] D. Pruthi, M. Gupta, B. Dhingra, G. Neubig, and Z. Lipton. 2020. Learning to Deceive with Attention-Based Explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[42] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[43] N. Reimers and I. Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.

[44] S. Rothe, J. Mallinson, E. Malmi, S. Krause, and A. Severyn. 2021. A simple recipe for multilingual grammatical error correction. *arXiv preprint arXiv:2106.03830* (2021).

[45] K. Rydzewska, J. Pawłowska, R. Nielek, A. Wierzbicki, and G. Sedek. 2021. Cognitive Limitations of Older E-Commerce Customers in Product Comparison Tasks. In *IFIP Conference on HCI*.

[46] B. Scheibehenne, R. Greifeneder, and P. Todd. 2010. Can there ever be too many options? A meta-analytic review of choice overload. *JCR* (2010).

[47] S. Sen, K. Gupta, A. Ekbal, and P. Bhattacharyya. 2019. Multilingual unsupervised NMT using shared encoder and language-specific decoders. In *ACL*.

[48] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[49] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

[50] Y. Su, D. Vandyke, S. Baker, Y. Wang, and N. Collier. 2021. Keep the primary, rewrite the secondary: A two-stage approach for paraphrase generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

[51] M. Sundararajan, A. Taly, and Q. Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*.

[52] N. Vedula, M. Collins, E. Agichtein, and O. Rokhlenko. 2022. What Matters for Shoppers: Investigating Key Attributes for Online Product Comparison. In *European Conference on Information Retrieval*. 231–239.

[53] N. Vedula and S. Parthasarathy. 2021. Face-keg: Fact checking explained using knowledge graphs. In *Proceedings of WSDM*.

[54] A.K. Vijayakumar, M. Cogswell, R.R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424* (2016).

[55] S. Volokhin, M. Collins, O. Rokhlenko, and E. Agichtein. 2022. Generating and Validating Contextually Relevant Justifications for Conversational Recommendation. In *ACM SIGIR Conference on Human Information Interaction and Retrieval*.

[56] M. Yasunaga, J. Leskovec, and P. Liang. 2021. LM-Critic: Language Models for Unsupervised Grammatical Error Correction. *EMNLP* (2021).

[57] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.

[58] T. Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *ICLR* (2020).

[59] C. Zhou, G. Neubig, J. Gu, M. Diab, P. Guzman, L. Zettlemoyer, and M. Ghazvininejad. 2020. Detecting hallucinated content in conditional neural sequence generation. *arXiv preprint arXiv:2011.02593* (2020).

[60] C. Zhu, M. Zeng, and X. Huang. 2019. Multi-task learning for natural language generation in task-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*.