ExSiM: Towards Explainable Automated Evaluation of NLG Systems

Anonymous ACL submission

Abstract

Automated evaluation of Natural Language Generation (NLG) tasks is hard due to the possibility of multiple correct outputs. The common practice of evaluating NLG systems involves computing the similarity between a collection of automatically generated documents and their corresponding (human-written) golden reference documents. Unfortunately, existing document similarity metrics are black boxes and, thus, hard to interpret and explain, making robust evaluation of NLG tasks even more challenging. To address this issue, this paper introduces a new evaluation metric called ExSiM that provides a vector of scores instead of a single similarity score, where each component of the vector describes a particular property of the similarity metric, thus providing a natural way of explanation. Our experimental results with Wikipedia article triplets and a custom-created narrative dataset demonstrate that the proposed ExSiM vector can perform comparably to traditional metrics like BERTScore and ROUGE for undirected similarity assessment while providing useful explanations. In addition, ExSiM yields a higher human-machine agreement for directed similarity assessment.

1 Introduction

With the rise of Large Language Models (LLMs), the application of Natural Language Generation (NLG) systems has gained more popularity than ever (Novikova et al., 2017; Wang et al., 2018). As NLG systems are adopted more widely, automated evaluation of these systems at scale becomes an important issue. The most common practice for conducting such evaluations involves computing the similarity between a collection of automatically generated documents and their corresponding (human-written) golden reference documents. The traditional way of computing the similarity between a pair of documents has been to compare the overall lexical (n-gram-based)/semantic (embeddings-based) overlap between those documents. For example, ROUGE (Lin, 2004a) considers direct lexical overlap, while metrics like S+WMS (Clark et al., 2019), MoverScore (Zhao et al., 2019), and BERTScore (Zhang et al., 2020a), are based on semantic similarity between two documents. Unfortunately, these metrics have been criticized for many limitations, as discussed below.

1. **Difficulty in Storyline Matching:** In classical document similarity metrics like TF-IDF-Cosine-Similarity (Ramos, 2003), ROUGE (Lin, 2004a), etc., intra-document order between sentences is lost as these metrics compute similarity solely based on word frequencies or n-gram overlaps. This makes distinguishing two similar storylines with different orders for the same set of events difficult. Likewise, for recent embedding-based semantic metrics, the practice of averaging lower-level (e.g., sentence) embeddings to construct higher-level (e.g., document) ones (Le and Mikolov, 2014; Sultan et al., 2015; Saggau et al., 2023) loses order sensitivity.
2. **Difficulty handling Information Split and Fusion:** Consider a hypothetical task of using an NLG model (e.g., ChatGPT) to rearrange a collection of jumbled sentences into a coherent story. In this case, storyline matching becomes challenging as the NLG model would not simply copy and reorder the input sentences but rather generate novel sentences that may sometimes fuse two or more original sentences into one or split one original sentence into two or more. Existing document similarity metrics cannot properly capture these information split/fusion scenarios while matching the ordering of information simultaneously.
3. **Lack of Explainability:** Recently, transformer-based (especially LLM-based) architecture has gained much popularity for computing document similarity (Saggau et al., 2023; Fu et al.,

2023). Unfortunately, these black boxes provide a single similarity score and lack explainability, as all the reasoning behind the output score is hidden inside a pretrained neural network.

4. **Default Commutative Property:** Current similarity metrics treat the task as inherently symmetric/commutative, but this is not always true. Although many use cases are commutative, e.g. document clustering and recommender systems, where one does not care about directional similarity, i.e., no documents have superiority over one another. But there are also use cases where the task is non-commutative, like, most notably, NLG evaluation, where there is a ground truth reference document that a generated document is being compared against. In this case, a directional similarity between the reference and the generated is desired, and a commutative property should not be enforced.

To address the abovementioned limitations, we introduce ExSiM (Explainable Similarity Metric), which can quantify how well a reference narrative/storyline is preserved within a generated narrative while accounting for information splits/fusions. ExSiM also provides a vector of scores instead of a single similarity score, where each vector component describes a particular aspect of similarity, thus providing a natural explanation. Finally, ExSiM allows users to either preserve or discard the commutative property as they see fit based on their target application.

For evaluation, we performed two sets of experiments: 1) Undirected (commutative) similarity assessment experiments with sets of article triplets from Wikipedia, and 2) Directed (non-commutative) similarity assessment experiments with a handcrafted narrative pair dataset. These tests show that the proposed ExSiM metric can achieve comparable performance to well-known measures such as BERTScore and ROUGE in the case of undirected similarity assessment, i.e., the case when we look at how similar things are without worrying about which direction the similarity goes. Additionally, the ExSiM vector offers helpful explanations on how two documents are similar instead of providing a single numeric score. Finally and most importantly, the ExSiM metric yields a higher correlation with human judgments when assessing the similarity between two documents in a specific direction (non-commutative), e.g., matching machine-generated text against reference text.

2 Related Work

NLG evaluation metrics have been extensively studied in the literature over the past two decades. *ROUGE* (Lin, 2004b) is perhaps the most popular metric used today for the evaluation of automated NLG systems, mainly because it is a simple and automatic process. As of today, around 192 variants of *ROUGE* have been proposed (Graham, 2015) including *ROUGE* with word embedding (Ng and Abrecht, 2015) and synonym (Ganesan, 2018), graph-based lexical measurement (ShafieiBavani et al., 2018), Vanilla *ROUGE* (Yang et al., 2018) and highlight-based *ROUGE* (Hardy et al., 2019). However, ROUGE scores are not self-explanatory and cannot distinguish between similar narratives with different storylines.

Researchers also attempted to develop methods for evaluating reference-free model-generated text (Louis and Nenkova, 2013; Xenouleas et al., 2019). Distance measures between the machine-generated document and reference document based on word embeddings have also been proposed (Zhao et al., 2019; Sun and Nenkova, 2019). Model-based evaluation for text generation has also been a recent trend (Sellam et al., 2020; Zhang et al., 2020a; Yuan et al., 2021). There are also works done on taking lower-level similarity metrics and constructing higher-level ones (Wei et al., 2022; Gahman and Elangovan, 2023; Gómez and Vázquez, 2022). Yet, these metrics possess the same weaknesses as ROUGE in terms of lacking explainability and their inability to distinguish storylines.

Recently, researchers have spent a lot of effort evaluating different aspects of text generation techniques that rely on measuring textual similarity in some capacity. For example, Fabbri et al. (2021); Zhong et al. (2022) discussed how to perform meta-evaluation of summarization metrics along four explainable dimensions: *coherence, consistency, fluency, and relevance*. However, these explainable dimensions were evaluated manually by humans without any automation, which ExSiM offers.

As an interesting development, recent research has witnessed the emergence of Large Language Models (LLMs) like ChatGPT (Xie et al., 2023), as a versatile tool for evaluating various NLP tasks as well. For instance, Gao et al. (2023); Fu et al. (2023); Laban et al. (2023); Wang et al. (2023) investigated the strengths and limitations of ChatGPT as an evaluator of textual similarity and summa-

rization quality. Moreover, the GPT-4 model, as an evaluator of text generation and similarity tasks, shows better alignment with human judgement (Liu et al., 2023). Yet, LLMs are deep neural networks that are difficult to interpret.

3 Proposed Metric

We propose ExSiM as a document-level similarity metric, which leverages rigorously optimized sentence-level similarity metrics as fundamental blocks for building the document-level metric. As mentioned in Section 1, ExSiM returns a vector of interpretable numbers that capture different aspects of document similarity instead of providing a single score, as mentioned below.

- Global Storyline Similarity
- Localized Storyline Similarities
- Frequency of Splits and Fusions
- Coverage of Information
- Information Preservation
- Hallucination

We follow four sequential steps to compute these vector components as described below and visually demonstrated in Figure 1.

1. Information Segmentation
2. Segment Matching
3. Storyline Recreation
4. Scoring

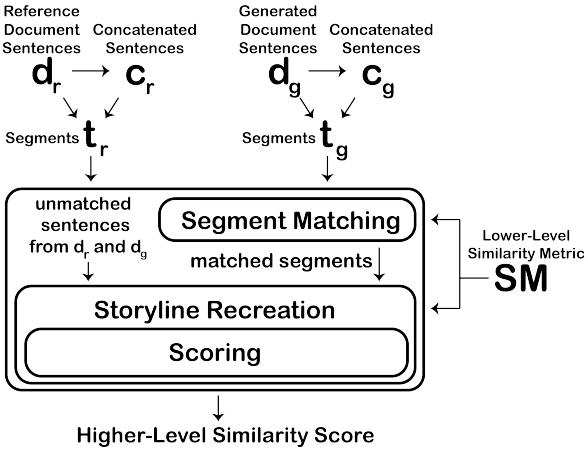

Figure 1: Framework for computing the ExSiM metric

3.1 Step 1: Information Segmentation

For model information split and fusion, we assume that clauses are units of information and all sentences are composed of at most two clauses. Under this assumption, we propose to measure a directional similarity score where we attempt to match each sentence from the generated document to one/more sentences in the reference, which needs to consider three cases as follows:

- **Matching Single Sentences :** The trivial case is a direct matching between two single sentences.
- **Matching Fusion:** To capture fusion, we concatenate two adjacent sentences (let us call them segments) in the reference document and match them to a single sentence in the generated one. Formally, we construct from the reference document a set of concatenated adjacent sentences, $c_{r,i} = s_{r,i} + s_{r,i+1}$ such that $0 \leq i < n_r - 1$. For an example, refer to Figure 2 and find how segment $t_{r,9}$ matched with segment $t_{g,6}$.
- **Matching Split:** To capture split, we match a singular sentence in the reference document to two in the generated document, which recreates a split, we construct in-kind $c_{g,i}$ such that $0 \leq i < n_g - 1$.

Since, from now on, sentences and concatenated adjacent sentences are treated in a similar manner, we will refer to them collectively as *segments*. We have a collection of $2n_r - 1$ segments from $d_r = (s_{r,0}, s_{r,1}, \ldots, s_{r,n_r-1})$ and $2n_g - 1$ from d_g likewise. We will use the following notation to represent the reference segments according to the following given $0 \leq j < 2n_r - 1$ (and likewise for generated segments):

$$t_{r,j} = \left\{ \begin{array}{l} c_{r,j/2}, \text{if } j \text{ is even} \\ s_{r,(j-1)/2}, \text{if } j \text{ is odd} \end{array} \right\} \quad (1)$$

3.2 Step 2: Segment Matching

Because of our simplified assumption, we may now construct a quite simple methodology to match segments. Consider a reference document d_r of length n_r and a generated document d_g of length n_g. A hypothetical segment-matching scenario can be seen in Figure 2.

We now want to derive which matches are best. This is where our lower(sentence)-level similarity metric (SM) comes into play. We compute the similarity on each possible segment pair between the documents, generating $(2n_r - 1)(2n_g - 1)$ similarity scores. To institute our transformation, we now choose the best matches using a greedy algorithm similar to the one used by Islam and Inkpen (2008). More specifically, we consider three cases:

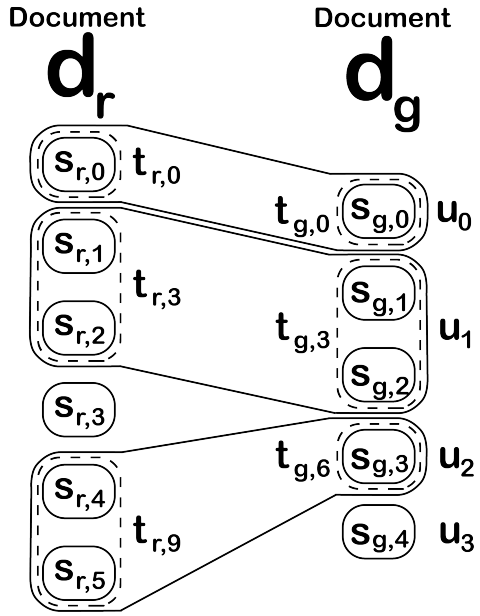

Figure 2: A hypothetical example of reference and generated document with their segments matched

- **Case 1: Sentence Vs. Sentence:** When the current best-matched segments (according to our greedy criteria) are both single sentences, we must disallow any concatenations containing each matched sentence from further consideration to avoid duplication. To be exact, supposing we matched $s_{g,x}$ with $s_{r,y}$. We would get rid of all the matches of the form $(s_{g,x}, t_{r,j})$ for all j and also the concatenations of the form $(c_{g,x-1}, t_{r,j})$ and $(c_{g,x}, t_{r,j})$ for all j. This also goes for eliminating the matches containing $s_{r,y}$ and its concatenations.
- **Case 2: Sentence Vs. Concatenation:** We, again, disallow matches that involve either of the sentences or the concatenation to avoid duplication. Similarly, we disallow the concatenations that contain the sentence. *But what about for the matched concatenation?* In this case, it is not just sufficient to disallow the two sentences that make up the matched concatenation. But since those sentences are disallowed, so too must we disallow the other concatenations that contain them. To be exact, supposing we matched $c_{g,x}$ with $s_{r,y}$. The full set of matches we would get rid of for $c_{g,x}$ would be $(c_{g,x-1}, t_{r,j})$, $(s_{g,x}, t_{r,j})$, $(c_{g,x}, t_{r,j})$, $(s_{g,x+1}, t_{r,j})$, and $(c_{g,x+1}, t_{r,j})$ for all j. As for the matched sentence, the duplicative matches would be disallowed like normal.
- **Case 3: Concatenation Vs. Concatenation:** In this case, we disallow the duplicative matches for both concatenations as described above.

The pseudo-code of our greedy approach is provided as Algorithm 1. It's noteworthy that we can record four values for our final similarity vector as part of this algorithm: the frequency of sentence splits, the frequency of fusions, and the coverage of the reference and generated documents. This last metric is computed by finding the percentages of how much each document matched the other.

Algorithm 1 Segment Matching

Require: Documents d_r, d_g, and similarity metric SM
  $s_r[]$, $s_g[] \leftarrow$ sentences(d_r), sentences(d_g)
  $c_r[]$, $c_g[] \leftarrow$ adjacentConcats(d_r), adjacentConcats(d_g)
  $t_r[] \leftarrow [s_r[0], c_r[0], s_r[1], c_r[1], \dots]$ ▷ Segments
  $t_g[] \leftarrow [s_g[0], c_g[0], s_g[1], c_g[1], \dots]$

  $simScores[][]$ ▷ Find Similarity Scores
  for $t_r[i]$ **in** $t_r[]$ **do**
    for $t_g[j]$ **in** $t_g[]$ **do**
      $simScores[i][j] = SM(t_r[i], t_g[j])$
    end for
  end for

  $matches[]$ ▷ Find Segment Matches
  while $max(simScores[][]) > 0$ **do**
    $i, j \leftarrow maxIndex(simScores[][])$
    $matches[]$ += $(t_r[i], t_g[j])$
    $simScores[i-1, i, i+1][] \leftarrow 0$ ▷ Zero out columns
    if i is odd **then** ▷ If a concatenation
      $simScores[i-2, i+2][] \leftarrow 0$
    end if
    $simScores[][j-1, j, j+1] \leftarrow 0$ ▷ Zero out rows
    if j is odd **then** ▷ If a concatenation
      $simScores[][j-2, j+2] \leftarrow 0$
    end if
  end while

3.3 Step 3: Storyline Recreation

At this stage, we have found m segment matches, i.e., $T = ((t_{r,0}, t_{g,0}), (t_{r,1}, t_{g,1}), \dots, (t_{r,m}, p_{g,m}))$. Let us further assume that the original order of d_g orders T, and therefore, $t_{g,0}$ is the earliest matched segment in d_g and $t_{g,m}$ is the last.

Let us now begin with how our metric scores the recreation of a storyline as a human reader would. When we, as a reader, are asked to compare documents, we, of course, read the reference document first. This establishes the context against which we should compare the generated document. To simulate this behavior, ExSiM scans through the new document d_g while periodically referring back to the ground truth d_r. This makes our metric directional/non-commutative.

For storyline recreation, we can neither use only matched segments nor merely use the sentences as given. To this end, let us refer to d_g's *used segments* as all the segments in d_g that were matched alongside the unmatched sentences in d_g. And when

ordered, these used segments comprise the entire d_g document. If we refer to these k used segments as $u_0, u_1, \ldots, u_{k-1}$ (see Figure 2), we can say:

$$d_g = u_0 + u_1 + \cdots + u_{k-1} \quad (2)$$

Given the used segments in d_g, one can see how well the generated narrative transforms to the reference one by iterating on a segment-by-segment basis. But narrative is not truly a disordered set of independent thoughts, but the sequenced set of connections formed between them. Thus, let us use *connections* to mean the semantic meaning that relates one segment in a given document to the next segment in that same document. A diagram of these connections can be found in Figure 3.

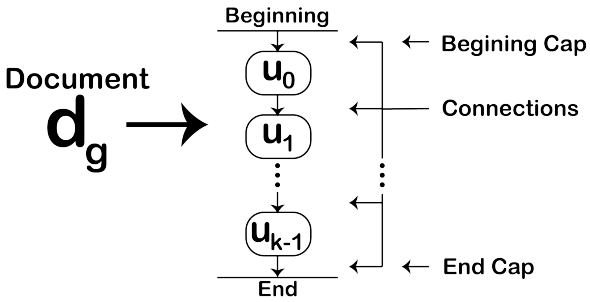

Figure 3: Labeling of connections for document d_g

Given all this, we estimate the ease of narrative transformation between d_r and d_g in a computationally efficient manner by determining for each narrative connection between used segments in d_g, how well they are supported by the reference d_r.

As discussed, some connections occur between every two adjacent used segments in d_g. But to better mimic how human readers compare texts, we also add *cap* connections between the beginning of the generated document and the first segment, and likewise between the last segment and the end of the document. Thus, if we have k used segments in d_g, then we have $k + 1$ connections (see Figure 3). Now, these connections must be scored in their correspondence to d_r, the ground truth. We will refer to the score values for each connection as $\{v_0, v_1, \ldots, v_k\}$.

Ease of Transformation Score: For a given connection in d_g, we score it by how easily it can be transformed back into d_r. The prototypical case (without any exceptions) is as follows. Suppose we are finding the score v_2 of the connection between u_1 and u_2 as seen in Figure 2. $u_1 = t_{g,3}$ and $u_2 = t_{g,6}$, and furthermore, $t_{g,3}$ is matched with $t_{r,3}$ while $t_{g,6}$ is matched with $t_{r,9}$. This means that our connection, when transformed back to d_r, best corresponds with the portion of d_r that starts at $s_{r,1}$ and goes all the way to $s_{r,5}$. Thus, we use our similarity metric (SM) to score the similarity between u_1+" "+u_2 (our generated connection) and the reference portion in d_r such that $v_2 = SM(u_1$+" "+$u_2, t_{r,3}$+" "+$s_{r,3}$+" "+$t_{r,9})$.

Yet, as mentioned, a few overlapping exceptions for scoring connections should be discussed, e.g., Cap Connections, Unmatched Connections, Patching Connections, and Inversely Ordered Connections. However, we discuss these exception cases in the appendix due to a lack of space.

3.4 Step-4: Scoring

For someone reading d_g, one can see how the average of these ease of transformation scores for each connection maps onto how well d_g recreates d_r. However, not all these scores contribute equally to the narrative recreation. For example, both the cap and patch connection scores are weighted differently (explained in the appendix due to lack of space), being scaled by h_c and h_p, respectively. And to keep our ExSiM score normalized, let us suppose m_i equals the maximum score v_i could have achieved after being scaled by the hyperparameters it was scaled by, if any. So, in all, we score each connection and then combine them into our final storyline recreation score using a weighted average.

$$ExSiM(d_r, d_g) = \frac{\sum_{i=0}^{k} v_i}{\sum_{i=0}^{k} m_i} \quad (3)$$

Thus, our ExSiM vector of similarity metrics can be completed to allow for greater explainability. On top of the four segment-matching metrics mentioned earlier and the storyline recreation score computed above, we may now append four more. For the local similarity scores and their positions along their documents, we simply take the ease of transformation score for each connection and record the percentage of sentences that occur before it. Lastly, for our information preservation, we append the average score of matched connections, and for hallucination, we append the average score for patching connections.

4 Explainability of ExSiM

The ultimate benefit of ExSiM is that while providing a useful similarity score as above, it can also produce a host of auxiliary metric values that provide other insights into the similarity between two

documents. Although we believe more values and insights could be derived, we will only list the metrics that are relevant to the qualitative experiment done in section 6.2:

- **Information Preservation and Hallucination:** ExSiM can provide insights on how much information is preserved and how much hallucinated by using data gathered during the matching and storyline recreation steps. For information preservation, ExSiM keeps track of what percentage of reference sentences were matched and the average matched connection score. If these are both low, then that means there are many reference sentences being skipped over. For hallucination, ExSiM computes what percentage of generated sentences were matched and the average patching connection score (explained in the appendix). If these are both low, then that means there are large swaths of the generated document that are not represented in the reference document and thus hallucinated.
- **Similarity Variance Along Documents:** Consider the non-commutative problem we have been discussing with reference d_r and generated d_g. Since we are scoring every connection in d_g, and each connection can be described as what percentage of sentences it occurs after, then for each ease of transformation score, we can associate it with a percentage detailing how far along d_g it occurs. Furthermore, one could create a graph visualizing a set of documents D_G and where they tend to be more or less similar to the correct documents D_R. For example, one could imagine a generative NLP model performing worse towards the end of its generation, and our model could provide evidence for or against this worry. On the other hand, this can be done in the opposite direction, wherein one can see what part of the documents in D_R the model tends to do the worst in recreating.
- **Counting Fused and Split Sentences:** Given our segment matching algorithm, one can see the tendency for whether many sentences are concatenated in d_r, d_g, or both. If many sentences are concatenated to make segments in d_r, then our generative model fuses many sentences together. If the concatenations are occurring d_g, then that means our model is splitting many sentences apart. And, of course, one can find further nuance than this in the actual counts.

5 Experiments

To test the accuracy of our ExSiM similarity metric, we conducted two sets of experiments as follows.

Undirected Similarity Assessment Experiments: *Wikipedia Triplets* is a dataset created to test document similarity metrics (Dai et al., 2015). They have generated a dataset where the rows are triplets of Wikipedia links such that the first two articles are more similar than the second is to the third. The dataset has two portions, a synthetic and handpicked section. For the synthetic triplets, they generated 19,876 rows of which 11,566 still have live pages that we were able to use. These pages which were used as documents had on average approximately 113 words and 6 sentences each. For the handpicked triplets which they verified are aligned with human intuition, of the original 172 triplets there are 151 live pages. These pages had on average 337 words and 16 sentences each.

They boast scores that edge out our own on their dataset, but it is important to note that not only are many of the pages they used dead but also the pages themselves have changed significantly in the last eight years. However, these changes should not impact the utility of the dataset since the ideas behind the articles are the same, we merely posit that what scores models should be expected to attain has likely shifted.

Directed Similarity Assessment Experiments: As for *Directed Similarity Assessment* , we tasked five Auburn University undergraduate students with annotating 20 alternative narrative pairs constructed in the following manner. We took the first five articles from the CNN Daily Mail data set as provided by (See et al., 2017) and (Hermann et al., 2015). For the articles we used, they had on average approximately 330 words and 23 sentences. Then for each article, we shuffled the sentences before feeding them into four different models that attempted to reorder the sentences back into their correct order. Then for each of the five ground truth articles, the students were asked to rate the similarity of each reordered document to the original. They were likewise asked to rate the similarity of the sentence ordering of the generated documents to the original. The questions we asked can be found in appendix C.

The four reordering models were Facebook's BART (Lewis et al., 2019), OpenAI's GPT-3.5 (Brown et al., 2020), ReBART (Chowdhury et al., 2021), and DistilBART (Shleifer and Rush, 2020).

6 Results

For our implementation, ExSiM used as its building block an SBERT-based sentence transformer model "all-MiniLM-L6-v2" from (Reimers and Gurevych, 2019). We chose this transformer to generate embeddings for our cosine similarity SSM because it performed the best on all three datasets. We tested it against other such sentence transformers like SBERT below and also against averaged word embeddings from GloVE. To tokenize our documents into sentences, we used spacCy 2 (Honnibal and Montani, 2017) and, more specifically, the English transformer pipeline with Roberta-base[1]. As for our hyper-parameters, we set $h_c = h_p = 1$ in experiments.

In terms of the models we tested our SMU-made DSM against, we for one used "sentence-transformers/paraphrase-distilroberta-base-v1" model which we denote as SBERT. And we also used "all-MiniLM-L6-v2" which we denote as MiniLM. For Deberta and Roberta, we used BERTScore as set out by Zhang et al. (2020b). For the GloVE implementation, we used Řehůřek and Sojka (2010).

6.1 Quantitative Results

6.1.1 Undirected Similarity Assessment

	Wikipedia	
	Synthetic	Handpicked
Deberta E_D	73.0%	80.3%
Roberta E_D	76.0%	84.2%
Avg. SBERT E_S	76.2%	92.1%
Avg. MiniLM E_S	77.1%	**94.0%**
Avg. GloVe E_W	70.7%	88.7%
ExSiM*	77.7%	89.4%
ExSiM	**77.7%**	91.4%

Table 1: Wikipedia Dataset Results. * matches between concatenations disallowed. E_D, E_S, and E_W mean document, sentence, and word encoders.

Table 1 lists the accuracy of each respective model's generated similarity scores. A model's scores are computed between the first two documents in each triplet and the last two, and the model is counted as generating the correct scores if the similarity score between the first two documents is higher than the score between the last two. Of further note, this task is commutative since the ordering of documents should cause no change in the similarity scores. Thus, we used the commutative variant of ExSiM (details in section 3.3).

One can see that ExSiM achieves comparable performance with state-of-the-art document similarity metrics. It edges out all the other metrics on the synthetic task while beating most on the handpicked but ultimately coming short of the impressive score from "Avg. MiniLM E_S." Now, we are not contending these results, in and of themselves, to be decisive. Instead, we argue that these results show the explanatory power of ExSiM. With that perspective, its other benefits show its worth:

- *ExSiM shows superior explainability to these other metrics*, because instead of being a black box, one can understand exactly where these scores come from down to the lower-level similarity metric.
- *This explainability allows other metrics to be derived*, as discussed in section 4 and used in section 6.2.

	Correlation with Human Annotations	
	Similarity	Ordering
Deberta E_D	0.632	0.589
Avg. SBERT E_S	0.526	0.547
Avg. MiniLM E_S	0.537	0.558
Avg. GloVe E_W	0.537	0.537
ExSiM (Commutative)	0.621	0.579
ExSiM (Non-Commutative)	**0.768**	0.716
ExSiM* (Commutative)	0.579	0.589
ExSiM* (Non-Commutative)	0.747	**0.8**

Table 2: Correlation (Kendall's tau) between Human and Directed Similarity Assessment. * matches between concatenations disallowed. E_D, E_S, and E_W mean document, sentence, and word encoders.

6.1.2 Directed Similarity Assessment

Table 2 summarizes the results of our Directed Similarity Assessment experiments. Looking at Table 2, one can see the correlation between any given models' similarity scores and the human-annotated ground truths. The inter-annotator agreement, computed using Kendall Tau, was 0.68 on the question of ordering similarity and 0.58 for overall similarity. These correlations were computed using Kendall Tau as they are ultimately ranking of similarities between ground truth documents and reordered documents. For a table that uses the Spearman ranking correlation metric instead, see table 4 in appendix D.

[1] https://huggingface.co/spacy/en_core_web_lg

	Average per Document					
	Fuses	Splits	d_r Matched	Matched Scores	d_g Matched	Patching Scores
BART	3.2	2.6	81.3%	0.231	100%	N/A
GPT-3.5	3.2	2.8	93.4%	0.230	95%	0.037
ReBART	4.4	2.2	100%	0.228	68.6%	0.208
DistilBART	3.0	3.0	71.4%	0.212	100%	N/A

Table 3: Auxiliary metrics derived from ExSiM

Here, ExSiM boasts better performance and we attribute this to the task being on-commutative wherein the correct document is treated differently than the generated document it's being compared to. This pans out through the high variance in correlation between our commutative and non-commutative models. Clearly, the non-commutative models correlate better with human intuitions than any other model, especially on the question of how well documents were ordered.

One can also see how ExSiM varies in performance based on whether it was permitted to match segments that were both concatenations. Those that were allowed showed better correlation on the question of general similarity whilst those that were not allowed correlated better on the ordering criterion.

The great performance on the ordering section of the human annotations is of particular importance. For one, this section has higher inter-annotater agreement giving better credence to the data. Furthermore, this correlation bodes well for our claim that ExSiM serves as a better metric in terms of its sensitivity to intra-document order.

6.2 Qualitative Examples

In order to present the potential use of ExSiM's auxiliary metrics as given in a vector, we have decided to use the same dataset used in our directed similarity assessments from section 6.1.2.

For sentence fusing/splitting and ordering, we show the auxiliary metric values in Table 3. Here, one can see that ReBART tended to be worse about fusing sentences while otherwise, the models tended to have similar habits about fusing/splitting sentences.

More interestingly, let us discuss the rest of the metrics that describe information preservation and hallucination as mentioned in section 4. The d_r and d_g matched columns tell us what percentage of the respective documents are matched to the other. The matched scores column has the averaged scores of matched connections and likewise for patching. And for the patching (unmatched) scores, the N/A values occur because these models never had a patch because they always matched all their sentences.

As discussed, for investigating information preservation we can see if a model is doing poorly if it has a low matching rate for d_r and low matching scores. Given this, it becomes DistilBART does a poor job at maintaining all the information in the reference document. And not very surprisingly, GPT-3.5 performed the best.

As for hallucinations, the models that had 100% match rates for d_g likely did not hallucinate much. And when we compare the two models that generated longer texts than the reference texts (GPT-3.5 and ReBART), it seems neither hallucinated much either. GPT-3.5 tended to maintain a similar number of sentences but when it did go over, the patching score is awful meaning that when GPT added text it was highly hallucinatory. ReBART has the opposite case where it had many extra sentences but the patching scores are pretty high meaning many of these appended sentences are relevant. Therefore it may have only really hallucinated in terms of volume.

We were able to find all these insights automatically using ExSiM.

7 Conclusion

This paper presents ExSiM, a novel metric that offers a score vector rather than a singular similarity score for evaluating NLG tasks. Each element of this vector represents a specific aspect of the overall similarity assessment, offering an intuitive method for explanation.

We believe ExSiM shows great promise as a timeless methodology that will allow the upgrading of contemporary and future semantic similarity metrics. Explainability is critically important, especially in an age rife with black boxes. Understanding what we use not only allows methodical improvement but is superior when in actual use because of the explanation and intuition it engenders.

8 Limitations

One fundamental limitation is the axiomatic assumptions we took about the language which may limit the use of ExSiM on a language-by-language basis, like assuming that sentences only have at most two clauses.

But beyond that, most limitations provide avenues to follow for future work. For one, we would like to attempt iterating ExSiM over itself multiple times to produce a similarity metric for long-form documents. There are already datasets out there for testing the long document similarity metrics we would produce, like (Ginzburg et al., 2021). ExSiM could also boast better performance through hyper-parameter optimization, improvements on the matching algorithms, and other tinkering with the methodology.

There are also some caveats that may assuage worries about the ExSiMs' comparable performance in the Undirected Similarity Assessment.

- *Our hyper-parameters were chosen intuitively*, we chose for both our cap importance (h_c) and patch importance (h_p) hyper-parameters to equal 1 because we supposed that to be the best.
- *Our model was designed to perform one linguistic level-up*, meaning it is little wonder that our ExSiM performs better on the shorter single-paragraph synthetic tasks while lagging behind on the longer multi-paragraph handpicked tasks. It is likely that our metric ought to be used differently, using linguistic stacking to achieve improved performance on longer documents. But as is, this remains a future direction for our work.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Somnath Basu Roy Chowdhury, Faeze Brahman, and Snigdha Chaturvedi. 2021. Is everything in order? a simple way to order sentences.

Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2748–2760. Association for Computational Linguistics.

Andrew M. Dai, Christopher Olah, and Quoc V. Le. 2015. Document embedding with paragraph vectors.

Alexander R. Fabbri, Wojciech Kryscinski, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Trans. Assoc. Comput. Linguistics*, 9:391–409.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *CoRR*, abs/2302.04166.

Nicholas Gahman and Vinayak Elangovan. 2023. A comparison of document similarity algorithms.

Kavita Ganesan. 2018. ROUGE 2.0: Updated and improved measures for evaluation of summarization tasks. *CoRR*, abs/1803.01937.

Mingqi Gao, Jie Ruan, Renliang Sun, Xunjian Yin, Shiping Yang, and Xiaojun Wan. 2023. Human-like summarization evaluation with chatgpt. *CoRR*, abs/2304.02554.

Dvir Ginzburg, Itzik Malkiel, Oren Barkan, Avi Caciularu, and Noam Koenigstein. 2021. Self-supervised document similarity ranking via contextualized language models and hierarchical inference. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 128–137. The Association for Computational Linguistics.

Joaquin Gómez and Pere-Pau Vázquez. 2022. An empirical evaluation of document embeddings and similarity metrics for scientific articles. *Applied Sciences*, 12(11).

Hardy, Shashi Narayan, and Andreas Vlachos. 2019. Highres: Highlight-based reference-less evaluation of summarization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3381–3392. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2(2).

Philippe Laban, Wojciech Kryscinski, Divyansh Agarwal, Alexander R. Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. 2023. Llms as factual reasoners: Insights from existing benchmarks and beyond. *CoRR*, abs/2305.14540.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

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

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

Chin-Yew Lin. 2004b. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using GPT-4 with better human alignment. *CoRR*, abs/2303.16634.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Comput. Linguistics*, 39(2):267–300.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1925–1930. The Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Daniel Saggau, Mina Rezaei, Bernd Bisch, and Ilias Chalkidis. 2023. Efficient document embeddings via self-contrastive bregman divergence learning.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7881–7892. Association for Computational Linguistics.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond K. Wong, and Fang Chen. 2018. A graph-theoretic summary evaluation for rouge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 762–767. Association for Computational Linguistics.

Sam Shleifer and Alexander M. Rush. 2020. Pre-trained summarization distillation.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 148–153, Denver, Colorado. Association for Computational Linguistics.

Simeng Sun and Ani Nenkova. 2019. The feasibility of embedding based automatic evaluation for single document summarization. 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 1216–1221. Association for Computational Linguistics.

Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is chatgpt a good NLG evaluator? A preliminary study. *CoRR*, abs/2303.04048.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 899–909, Melbourne, Australia. Association for Computational Linguistics.

Chengwei Wei, Bin Wang, and C. C. Jay Kuo. 2022. Synwmd: Syntax-aware word mover's distance for sentence similarity evaluation.

Stratos Xenouleas, Prodromos Malakasiotis, Marianna Apidianaki, and Ion Androutsopoulos. 2019. SUM-QE: a bert-based summary quality estimation model. In *Proceedings of Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*, pages 6004–6010. Association for Computational Linguistics.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2023. Adaptive chameleon or stubborn sloth: Unraveling the behavior of large language models in knowledge clashes. *CoRR*, abs/2305.13300.

An Yang, Kai Liu, Jing Liu, Yajuan Lyu, and Sujian Li. 2018. Adaptations of ROUGE and BLEU to better evaluate machine reading comprehension task. In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 98–104. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *CoRR*, abs/2106.11520.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with bert.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 563–578. Association for Computational Linguistics.

Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2023–2038. Association for Computational Linguistics.

A Matching Edge Cases

There are edge cases that this approach does not properly handle, and that is cases when the transforming from one document to another would be best described as involving two sentences splitting then both of them fusing two other sentences respectively. But for the simplicity of the current approach, these rare cases being maltreated seemed warranted. Furthermore, these mismatches are ameliorated when later on ExSiM further combines adjacent matches which will likely recombine these separated clauses.

B Connection Exception Cases

- **Cap Connections:** We know readers care how documents begin and end and so we introduced cap connections. And we can and do score them as prototypical connections above, but critically it is not apparent *a priori* the relative importance of cap connections to typical connections. Thus, we have introduced a hyper-parameter h_c that denotes the scaling importance and acts as a scalar multiplier to whatever our cap ease of transformation scores are. For an example, in Figure 2 we can see a that the beginning cap connection in d_g matches well with the beginning of d_r. Therefore in this case, $v_0 = h_c \cdot SM(u_0, t_{r,0})$
- **Unmatched Connections:** What of the case when for a connection in d_g, the second segment was not matched to d_r? For an example, one can see the connection between u_2 and u_3 in Figure 2. Here, we declare it an unmatched match and give it an ease of transformation score of 0, i.e., $v_3 = 0$. But more than that, we declare this connection to be in need of a *patch*. This is because when a reader finds a connection that is not represented in the reference document, they do not merely forget about it and move on. They attempt to patch over the narrative hole. And if there are multiple unmatched connections in a row, they are all part of one hole in need of a patch.
- **Patching Connections:** When a narrative hole begins as mentioned, it becomes patched when we find find the next matched segment in d_g. The reason we are now able to patch over the hole is because we can form a patching connection between the last matched segment (which may in fact be the beginning as seen in Figure 3 which is not really a segment but behaves as one) and the

just now matched segment. For an example, let us see what happens when we score the connection between u_3 and the end of d_g. First, this is a cap so we will multiply our score by h_c. Second, the second segment in this connection is the end of d_g which cannot constitute an unmatched segment because it is matched by definition to the end of d_r. Therefore, the narrative hole starts at the matched u_2 and stops here at the end. Therefore our patch is u_2+" "+u_3 and it corresponds to where the start of the patch matched to and where the stopping place matched to. Furthermore, as discussed for caps, there is no clear *a priori* intuition on how patches should be scored in comparison to typical connections. Thus, we introduce hyper-parameter h_p to be a scaling factor. Therefore this particular patching capped connections, it as scored such that $v_4 = h_c \cdot h_p \cdot SM(u_2$+" "+$u_3, t_{r,9})$.

- **Inversely Ordered Connections:** There is the last exceptional case when we are given a matched connection but the start and stop d_g segments occur in reverse order of their matched segments in d_r. In this case, we determined that readers will only refer back to the reference segment that matches to the stop segment. This is because the inverse ordering completely dismantles the narrative between the segments and so the generated connection can most intuitively transformed back into the first occurring segment that it matched with.

C Human Annotation Questions

The form started with:

"Given "CorrectDocst.txt" and "ReorderedDocs.txt", please do the following. For each set (there are 5), first read that sets correct document. This is the ordering of ideas and semantic meaning that the reordering models are trying to recreate.

Then after reading each of these correct orderings, read the four reordered documents that were generated by four different models. Then for each one, grade it below based on how similar the ordering is, then how similar they are on a document level.

If you have any questions at all, please reach out."

Then for each set of four documents, we asked:

"Did the first model match the ordering well?" and "Did the first model match the overall meaning well?"

We allowed them to answer on a scale 1-5.

D Human Annotations Spearman Scores

See Table 4, the results are similar and warrant the same analysis.

	Correlation with Human Annotations	
	Similarity	Ordering
Deberta E_D	0.700	0.677
Avg. SBERT E_S	0.593	0.586
Avg. MiniLM E_S	0.622	0.623
Avg. GloVe E_W	0.641	0.606
ExSiM (Commutative)	0.691	0.671
ExSiM (Non-Commutative)	**0.864**	0.830
ExSiM* (Commutative)	0.668	0.678
ExSiM* (Non-Commutative)	0.852	**0.885**

Table 4: Correlation (Spearman) between Human and Directed Similarity Assessment. * matches between concatenations disallowed