GOT: Testing for Originality in Natural Language Generation

Jennifer Brooks
Department of Computer Science
The George Washington University
Washington, DC
jtbrooks@gwu.edu

Abdou Youssef
Department of Computer Science
The George Washington University
Washington, DC
ayoussef@gwu.edu

Abstract

We propose an approach to automatically test for originality in generation tasks where no standard automatic measures exist. Our proposal addresses original uses of language, not necessarily original ideas. We provide an algorithm for our approach and a run-time analysis. The algorithm, which finds all of the original fragments in a ground-truth corpus and can reveal whether a generated fragment copies an original without attribution, has a run-time complexity of $\theta(n \log n)$ where n is the number of sentences in the ground truth.

1 Introduction

This research addresses an ethical consideration for Natural Language Generation, namely, plagiarism. The Oxford English Dictionary defines *original* (adjective) as "present or existing from the beginning; first or earliest" and "created directly and personally by a particular artist; not a copy or imitation". But, if we apply the definitions of "original" to *language*, then there are two ways in which a piece of generated text may be original. For one, the text may express an "original idea", such as Einstein did in 1905 with "$E = mc^2$". On the other hand, a non-original idea may be expressed in an original way, via, for example, figurative language. Our proposed approach addresses original uses of language. It does not necessarily address original ideas.

How do we protect intellectual property when it comes to language generators that are trained on a world-wide-web of data? Our language generators have to be held accountable. They should also be protected. What if a language generator generates an original analogy? What if it writes a poem that is so great that it ends up in the history books? Multiple language generators may be trained on the same ground truth (e.g., Wikipedia) with the same embedding vectors (e.g., BERT (Devlin et al., 2018) and GPT (Vaswani et al., 2017; Radford et al., 2018)) and the same technologies (deep neural networks, LSTM cells (Hochreiter and Schmidhuber, 1997), transformers (Vaswani et al., 2017)). It will become a question of "Whose generator said it first?" With automatic language generation, we need a way to automatically measure, store, and reference original ideas and language. We propose one possible solution to these originality-related problems.

For the purposes of our analyses, we define *ground truth* as the set of sentences that are compared with the generated sentences. The ground truth may be larger than the training set, but should include the training set. The gound truth would also, ideally, grow. For example, the ground truth could start out as the training set, but as new sentences are generated with a trained model, then the new sentences may be added to the ground truth. We also claim that generated sentences should only be added to the ground truth if they are original or include citations where appropriate.

2 Background

Our criteria and basis for evaluating measurements of originality are:

1. Can we tell whether a generated sentence is an original use of language?

2. Can we tell whether the sentence contains a fragment from the ground truth that is a candidate for protection as intellectual property?

Therefore, when measuring generation originality by comparing the generated sentence with the sentences in the ground truth, then the answers to numbers 1 and 2 above are binary. Either the generated sentence is an original use of language or it is not. Either the generation is at risk of plagiarism or

Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021), pages 68–72

it is not. However, if we consider that the ground truth may not be representative of all the sentences that have ever been generated, then there is a measure of uncertainty that may be added to the binary outcome.

There are no standard automatic measures for novelty and originality in stylized language generation (Mou and Vechtomova, 2020). High perplexity (PPL) and a low BLEU (Papineni et al., 2002) score may suggest novelty, but they are not sufficient for testing for originality. High PPL and a low BLEU score may be achieved when there is little overlap between the generated language and the ground truth, but nonesense and off-topic sentences are rewarded. While nonesense sentences may be novel, they may be grammatically incorrect, and sentences that are grammatically correct will likely have some overlap with fragments (n-grams) in the ground truth, such as using phrases like "she said that". So, we want a generation originality test that doesn't penalize n-gram overlap. (An original use of language may combine common n-grams in a new way.) We also want a generation originality test that flags potential plagiarism of original fragments in the ground truth, which neither BLEU nor PPL does.

We propose a generation originality test (GOT) that addresses original uses of language. It does not necessarily address original ideas. GOT is equally appropriate for stylized text generation, where novelty is desirable, and for other generation tasks where there is not an imposed style but the generation is open-ended, including summarization tasks.

3 Proposed Approach

Our proposed generation originality test (GOT) determines whether:

1. any fragment in a generated sentence equals an "original" fragment in the ground truth, in which case the generation may be in violation of a copyright law, if no citation of the original source is included; or,

2. the generated sentence is "original", per Definition 1, below.

Definition 1 (Original Sentence)**.** *A sentence, whether generated or in the ground truth, of n tokens is original if there exists an original k-gram within the sentence for some k≤n. The originality of k-grams is defined next.*

The definition of originality of a fragment (or k-gram) depends on whether we are referring to a generated fragment or to a fragment in the ground truth. Generated fragments are tested against the ground truth. If the generated fragment does not appear in the ground truth, then the generated fragment is considered original. If it appears once in the ground truth, then it is considered not original and so a citation may be needed. See Table 1 for a summary of the criterion for each type of fragment to be true. In Table 1, C equals the number of times that fragment appears in the ground truth.

	Type	Criterion
Ground Truth Fragment	Original	$C = 1$
	Not Original	$C \geq 2$
Generated Fragment	Original	$C = 0$
	Not Original, Citation Needed	$C = 1$
	Not Original, No Citation Needed	$C \geq 2$

Table 1: Criterion per fragment type, where C is the number of times the fragment appears in the ground truth. Note, C is always with respect to counts in the ground truth, even when evaluating generated fragments.

Ground-truth fragments that appear once and only once in the ground truth are considered original.[1] Likewise, fragments that appear more than once in the ground truth are considered "not original". For example, "lengthened shadow" appeared twice in our ground truth and so it is not considered an original phrase in the ground truth. Combining non-original fragments to generate a new idea or analogy, however, could be considered an original use of language. For example, "the writer is the lengthened shadow of a man" contains the fragments "the writer is" and "the lengthened shadow" and "of a man" which are not original fragments in our ground truth. However, the way in which they are combined in this example creates an original use of language – in this case, a metaphor. (Examples of fragments that appeared many times in our

[1]For simplicity of explanation, we qualify a fragment as "original", and therefore a candidate for protection of intellectual property, if it appears "once and only once" in the ground truth. However, with very large datasets, it may be necessary to relax the criteria from "once and only once" to a relatively small number of occurrences, in order to consider a fragment a candidate for protection of intellectual property.

training set are “it is” and “human life”.)

Here is one possible use of GOT. If a generated sentence contains a fragment that appears once and only once in the ground truth (after duplicate sentences are removed from the ground truth), then the generated sentence may be discarded because it contains a fragment from the ground truth that is a candidate for protection as intellectual property. In other words, the sentence may be in violation of a copyright law. Otherwise, the sentence could include a citation of the source for the original fragment.

The definition of ground-truth original fragments actually calls for more nuance, which we will elaborate and explain how to compute next. We maintain a count per fragment that is incremented each time the fragment appears in a new sentence in a new document or by a different author (if the author can be determined in both instances) in the ground truth. In other words, if a fragment in the ground truth is repeated in the same document, or by the same author across documents, then the count for that fragment is incremented only once. (Therefore, an author, if known, should also be stored for each fragment, at least until the count for that fragment is greater than 1. When the count for a fragment is greater than 1, then it has already been determined that the fragment was seen a second time in a different document by a different known, or unknown, author.) The count for a fragment will be 1 if it occurs just once in the ground truth, or if all of its occurrences are in the same document or by the same author; otherwise, the count will be greater than 1. Now, a ground-truth fragment is said to be original if and only if its count is 1.

See Algorithm 1 for psuedo-code to test for originality and find all original fragements in a dataset.

To examine fragments, we use a window length of wl varying between 2 and the sentence length, where wl is the number of words in the fragment. If the first or last word in the window is a determinant (e.g., ‘a’ or ‘the’), any use of the verbs *to be* and *to have* (‘is’, ‘are’, ‘am’, ‘was’, ‘were’, ‘has’, ‘had’, ‘have’), punctuation mark, or preposition/subordinating conjunction (e.g., ‘to’, ‘of’, or ‘from’), the window is moved one step to the right. (Shortening the window to get rid of the determinant, special verb, special character, or preposition would result in a window size already covered in the previous step.) All words and characters are allowed in the other positions of the window, so, for example, a comma or preposition may appear in the middle of a window of size 3 or more.

3.1 Runtime Complexity

The following complexity analysis is with respect to Algorithm 1. We are representing F and O with balanced binary search trees (e.g., red-black tree (Guibas and Sedgewick, 1978; OKASAKI, 1999)) where the comparator is lexicographic ordering. Searching, insertion and deletion in such trees take $\theta(\log n)$ comparisons. Since the length of fragments is assumed to be constant on average, then each comparison takes constant time, implying that each search/insert/delete operation in O and F take $\theta(\log n)$ time.

Given our representation of F and O with balanced binary search trees, consider the following time complexity analysis:

- Let n = number of sentences in the dataset. The first for-loop (line 1) iterates n times.
- Let c = the average length (i.e., number of tokens) of a sentence in our ground truth. We found that c = 25, a fairly small constant. Therefore, the two for-loops in Steps 4 and 5 iterate on average a constant number of times.
- The binary search in F (line 10) has a runtime complexity of $\theta(\log n)$.
- Depending on the result of the binary search of F (line 10) there may be an insertion to F (line 14) which has a runtime complexity of $\theta(\log n)$.
- Then the number of calculations in lines 1-20 is the following function of n: $2c^2 n \log n$.
- The code segment of lines 21-26 takes $\theta(n)$ time because the number of wl-token fragments in the ground truth dataset (of n sentences where each sentence consists of c tokens on average) is at most cn.
- Therefore, the runtime complexity is: $\theta(n \log n)$.

This algorithm would be executed before generation tasks, but may also be executed whenever the

[2]If the first or last word in the window is a determinant (e.g., ‘a’ or ‘the’), special verb (‘is’, ‘are’, ‘am’, ‘was’, ‘were’, ‘has’, ‘had’, ‘have’), punctuation mark, or preposition/subordinating conjunction (e.g., ‘to’, ‘of’, or ‘from’), the window is moved one step to the right.

Algorithm 1 Find Original Fragments in the Ground Truth

Require: Input S, the sentences in the ground truth to evaluate
Require: Input F, list of fragments already discovered, may be empty set;
Require: Input $CountPerFrag(f)$, for all $f \in \mathcal{F}$
Require: O, list of original fragments ▷ Count per $o \in \mathcal{O}$ should always be 1

1: **for each** $s \in \mathcal{S}$ **do**
2: $l =$ number of tokens in sentence s
3: $sentParts =$ set of tokens in s
4: **for each** wl in range 2 to l **do** ▷ $wl =$ length of window
5: **for each** i in range 0 to $l - wl + 1$ **do** ▷ assume zero-based indexing
6: **if** $sentParts[i]$ or $sentParts[i + wl - 1] =$ special token[2] **then**
7: Continue to next i
8: **else**
9: $frag = sentParts[i : i + wl]$
10: **if** $frag \in \mathcal{F}$ **then** ▷ binary search of F
11: $CountPerFrag[frag] = CountPerFrag[frag] + 1$
12: Break from for-loop in line 5
13: **else** ▷ frag was not found in F
14: Add $frag$ to F
15: $CountPerFrag[frag] = 1$
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21: Set O to the empty set;
22: **for each** $frag$ in F **do**
23: **if** $CountPerFrag[frag] == 1$ **then**
24: Add $frag$ to O;
25: **end if**
26: **end for**

reference set changes or is updated (for example, based on generated language).

4 Example: Results on One Application

To see how GOT performed on a generation task, we applied it to a metaphor generator that we built, based on an RNN (Elman, 1990) architecture with LSTM cells (Hochreiter and Schmidhuber, 1997) for training a language model on the language of metaphors, using only metaphors and their topics as input. (A topic was inserted at the beginning of each input sentence.)

The model was trained to predict the next word in the sentences from our ground truth—a set of 22,113 quotes, where each quote contains at least one metaphor and is labeled with a topic. There are 1,684 unique topics (e.g., "animals", "fear", "fishing", "grandparents", "happiness", "motives", "politics", and more examples listed in Table 2) and the dataset is currently available to the public online as part of "Dr. Mardy's Dictionary of Metaphorical Quotations" (Grothe, 2008).

To the trained language model, we apply an inference engine that uses weighted random choice with a "constraining factor" to encourage language coherence and originality in the output, and patterns of metaphors to encourage the generation of grammatically correct metaphors (Brooks and Youssef, 2020). The constraining factor, c (for $c \geq 1$), causes the inference engine to select—with a probability of $\frac{1}{c}$—the most likely word to appear next. Otherwise, and with a probability of $1 - \frac{1}{c}$, the inference engine will make a weighted random selection. Selecting the most likely next word encourages language coherencey in the output, while weighted random selection encourages originality. (We found that a constraining factor of 3 or 4 worked best with our model.)

A generated sentence failed the GOT if a fragment of at least 2 words appeared as an "original" fragment in the training set; that is, if the fragment appeared just once in the ground truth. Using our metaphor generator, we generated 500 metaphors from randomly chosen topics. Applying GOT on each of the 500 generated metaphors, we found that only 32 repeated an "original" fragment from the training set. From this experiment, we conclude that out of the 500 generated metaphors, 468 of them, or just over 93%, can be considered original. (Table 2 provides examples from our metaphor generator on randomly generated topics.)

Topic	Generated Metaphor
tears	The arrested waters shone and danced.
fathers	Expectations are premeditated resentments.
character	Today is the companion of genius.
friends	Assumptions are the termites of relationships.
writers	The writer is the lengthened shadow of a man.
world	This world is the rainbow of us.
truth	The brain is the eden of a star.
innocence	The cure for silence is the salt of speech.
imagination	Success is the only deadline.

Table 2: Examples of Generated Metaphors

5 Conclusion

Our approach to originality testing includes two contributions:

- An automatic test, where no standard existed, for originality in generated language
- An automatic test, where no standard existed, for identifying where generators are in violation of copying an original use of language without attribution

The first contribution tells us whether a generation is an original use of language. The second contribution tells us whether a generation is, at least, not at risk of committing plagiarism. For example, the sentence "A bird built a nest" is not an original use of language; however, it is at least probably not in violation of plagiarism since it does not contain a fragment that is so rare that it should be protected as an original use of language.

References

Jennifer Brooks and Abdou Youssef. 2020. Discriminative pattern mining for natural language metaphor generation. In *Proceedings of the Discriminative Pattern Mining Workshop*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jeffrey L. Elman. 1990. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211.

Mardy Grothe. 2008. *I Never Metaphor I Didn't Like: A Comprehensive Compilation of History's Greatest Analogies, Metaphors, and Similes*. Harper Collins.

Leo J. Guibas and Robert Sedgewick. 1978. A dichromatic framework for balanced trees. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 8–21.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Lili Mou and Olga Vechtomova. 2020. Stylized text generation: Approaches and applications. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 19–22, Online. Association for Computational Linguistics.

CHRIS OKASAKI. 1999. Red-black trees in a functional setting. *Journal of Functional Programming*, 9(4):471–477.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.