Ethical Citation Assistant

by
Beatrice Yaa Yeboah Cobbinah

Contents

Introduction

The concept of citation is an important one in an education and business setting. When we consider the growing value of data, it is understandable how and why there has been a growing importance in acknowledging and honouring the work various people do. In this light, it is then understandable why it is important to cite people's work and acknowledge people's hard work when you do use their work.

However, the process of citation is not always an easy one. While other people may be well versed in citating, for many people this process can be tedious and be an almost impossible task to achieve. From, genuinely missing out on certain cited works, to wrongly citing and tagging the different cited works, there is the need for any assistance that can be given.

Objectives / User Requirement

1. This software will facilitate the comparison of text uploaded with content already published in scientific and technical platforms including websites, books and online repositories.
2. This software will identify direct or paraphrased quotations in an uploaded text that match with publicised work.
3. This software will identify other published works that are using the same or similar language, concept or ideology.
4. This software will generate a report based on identified similarity and cited works.
5. The software will in no way store or retain a user's work after assistance has been provided.

State of the Art

Citation Assistance is often offered through different means. It is often associated with Plagiarism detection and text similarities. There are several plagiarism detection software some of them are discussed below.

Turnitin: It is no shock that this is one of the first software to be discussed, having become one of the most popular plagiarism detectors, Turnitin is an Internet-based plagiarism detection service that sell its licenses to universities and high schools who then use the software as a service (SaaS) website to check submitted documents against its database and the content of other websites with the aim of identifying plagiarism. Results can identify similarities with existing sources and can also be used in formative assessment to help students learn to avoid plagiarism and improve their writing.

All the other plagiarism detectors achieve the same or a similar purpose. Some of the more popular ones are listed below. –

Grammerly, DupliChecker, PlagTracker PlagScan and many more.

Methodology

The Search Component

In order to allow a wider and online version of this Assistant, Google Search Application Programming Interface (API) was used to allow connections and searches over the internet. SERP API was used as well, but due to its relevant limitation it was not used for the final display. Google Search API allowed summarized text or keywords to be searched on the web and similar or identical documents to it be found. Likewise, Google Programmable Engine allowed preference be given to certain websites like Wikipedia, science direct and others which are more likely to have published works as compared to other random websites. The output from the search API, provides the result for a USER REQUIREMENT that is- finding similar or related documents.

Google Programmable Search Engine (formerly known as Google Custom Search and Google Co-op) is a platform provided by Google that allows web developers to feature specialized information in web searches, refine and categorize queries and create customized search engines, based on Google Search. The service allows users to narrow the 11.5 billion indexed webpages down to a topical group of pages relevant to the creator's needs.

Keyword and Key Text

Opening the Assistant to the web is of course an essential thing, but it will be impossible to achieve if we have to search with the entire text provided by a user. This inspired the next section of this project.

In this part of the project, the first approach considered was a keyword search. The idea behind this was to simply extract two or three words that defined the entire context of the text submitted by the user and with these words we can find document with similar understanding.

Several techniques were considered however two main libraries used will be discussed in this section.

1. NLTK Rapid Automatic Keyword Extraction (RAKE): This is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analysing the frequency of word appearance and its co-occurrence with other words in the text.
2. DisitilBERT with Sentence Transformer library: DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark.
   The BERT architecture will be discussed further in the upcoming topics, however below are some of the outputs of both RAKE And DistilBERT after extracting keywords from a text.

```
['iteratively propagates information',
 'evolving visual features',
 'enjoying fast learning',
 'world indoor',
 'vice versa',
 'theart results',
 'simulated multi',
 'rich real',
 'object environments',
 'model ',
 'latent variables',
 'image quality',
 'compositional representations',
 'careful evaluation',
 'better dataefficiency',
 'achieves state',
 'outdoor scenes',
 'scenes',
 'terms',
```

Output from RAKE extraction

```
int(keywords)
```

```
multiplicative integration allows', 'encourage emergence compositional', 'resolution synthesis iteratively', 'synthesis iterati
ly propagates', 'efficient type transformer']
```

Output from DistilBert extraction

While these and the several other methods proved useful in extracting the keywords they didn't necessarily achieve our objective in excellence ie. Finding similar text. The text found though

similar were not necessarily identical in context and seemed to have a number of miss and fail. This inspired using a passage summarizer to summarize the full text into one line or two and then the summarized line was then searched.

Using a Passage summarizer:

Just like in finding the keywords, several methods were used some of which include but not limited to:

1. Using Genisim summarizer library
2. Using Transformer libraries' pipeline
3. Using T5ForConditionalGeneration, T5Tokenizer from the Transformer library
4. Using Summarizer library and several others.

While all these libraries saw significant performance, quite a number of them were considerably affected when the text was long and when proper cleaning was not done on the text. While the data cleaning could be worked on it also altered the final output a little when cleaning required removal of stopwords or punctuations.

Hence, for the summarization the Latent Semantic Analysis Summarizer (**LSA)** was used. The LSA summarizer proved to have a better performance because latent semantic analysis technique processing analyses relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis). A matrix containing word counts per document (rows represent unique words and columns represent each document) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Using this SVD concept, large texts can be summarized efficiently without losing the main integrity.

The image below shows the summarized sentence for the Text - Erika Rappaport Object Lessons and Colonial Histories Inventing the Jubilee of Indian Tea.txt

```
#Using the LSASummarizer library, We will summarize the text upload for comparison.
#LSA Summarizer allows the factorization, hence large text can be summarized faster
parser = PlaintextParser.from_string(textatten,Tokenizer("english"))
summarizer_lsa = LsaSummarizer()
# Summarize using sumy LSA
summary =summarizer_lsa(parser.document,2)
lsa_summary=""
for sentence in summary:
    lsa_summary+=str(sentence)
print(lsa_summary)
```

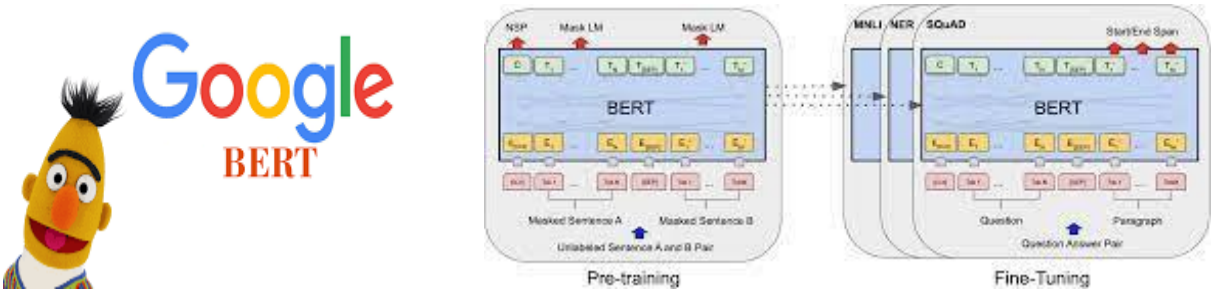
```
British consumers were thus to prefer Indian tea not out of patriotic fervor per se, but because they should naturally desire n
```

The Citation Component

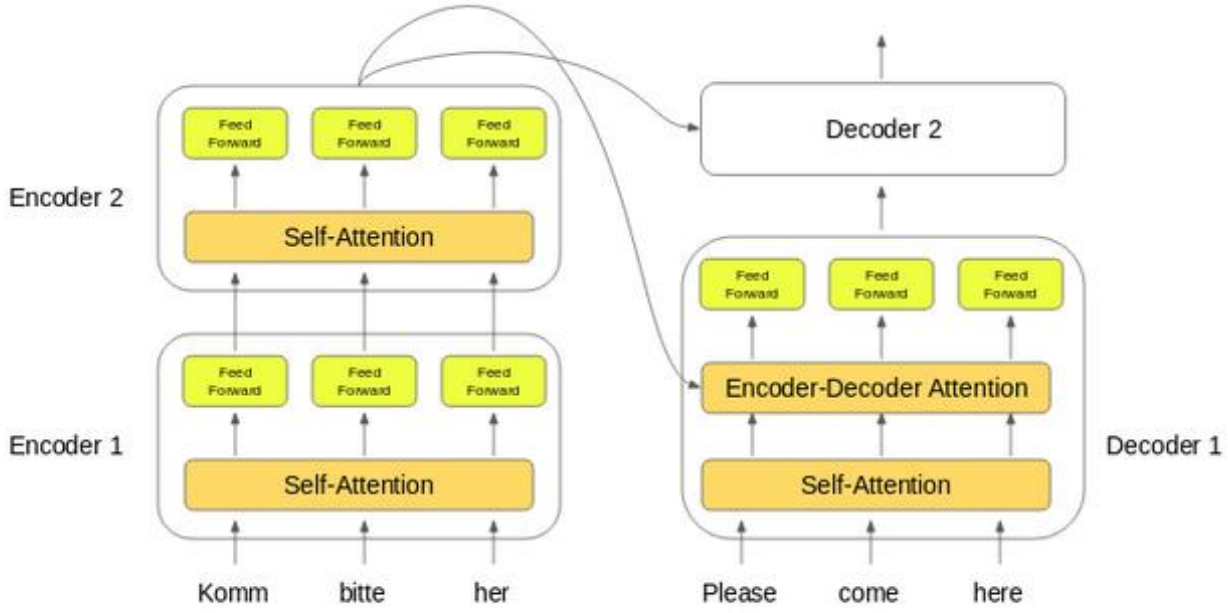
For the actual citation, we will be using two main algorithms.

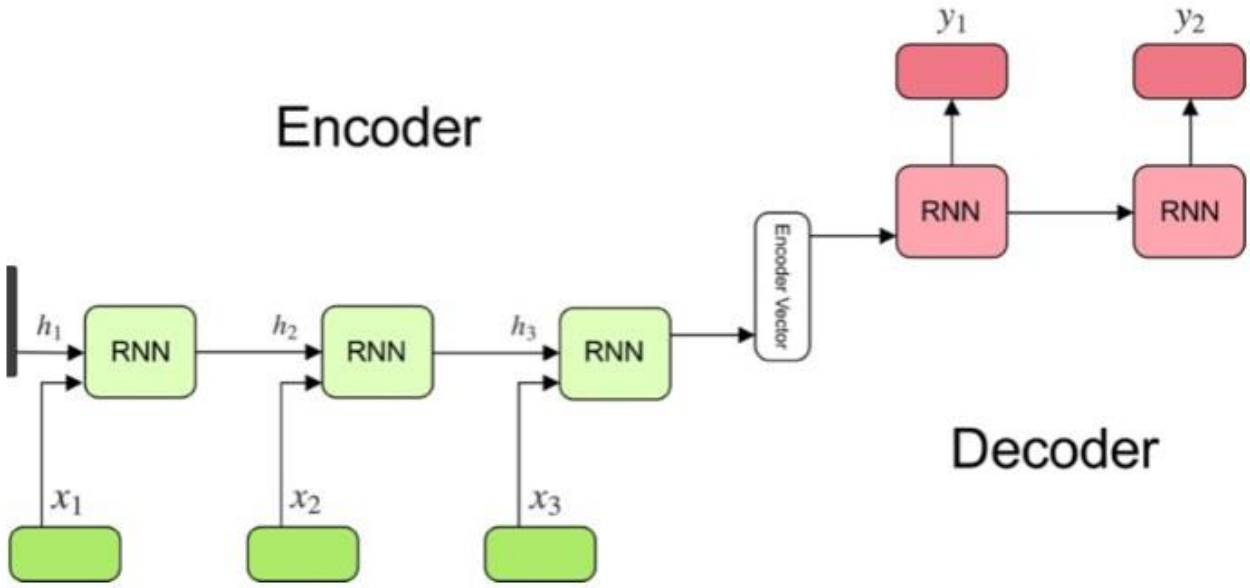
- Bidirectional Encoder Representation from Transformers (BERT):
- Cosine Similarity

1. BERT: This is a Transformer-based technique for language processing pretraining that was developed by Google. It was created and published in 2018 by Jacob Devlin and his colleagues from Google. The original English-language BERT has two models: - the BERT(base): 12 Encoders with 12 bidirectional self-attention heads, and the BERT(LARGE): 24 Encoders with 24 bidirectional self-attention heads. Both models are pre-trained from unlabeled data extracted from the BooksCorpus with 800M words and Wikipedia with 2,500M words.



The Transformer is a deep learning model introduced in 2017 that utilizes the mechanism of **attention**, weighing the influence of different parts of the input data. For example, if the input data is a natural language sentence, the Transformer does not need to process the beginning of it before the end. Transformers are designed to handle sequential input data however the process does not need to be done in a sequence. Instead, the context is identified and the meaning of each word in relation to the context is preserved. Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.





The BERT model used in this project was the base model with 12 Encoders (layers or transformer blocks) and 12 bidirectional attention heads and 110 million parameters. This means that a word starts with its embedding representation from the embedding layer. Every layer does some multi-headed attention computation on the word representation of the previous layer to create a new intermediate representation. All these intermediate representations are of the same size.

Using the Sentence Transformer library, we set the model to the BERT base model and we then create our word embeddings. For this project, because we would like to conserve the meaning of each word on a sentence level, we would first tokenize the sentence into sentences using NLTK's sentence tokenizer.

After these embeddings have been created, we need to find a means to compare and identify how similar or different these are. The method used was the Cosine Similarity.

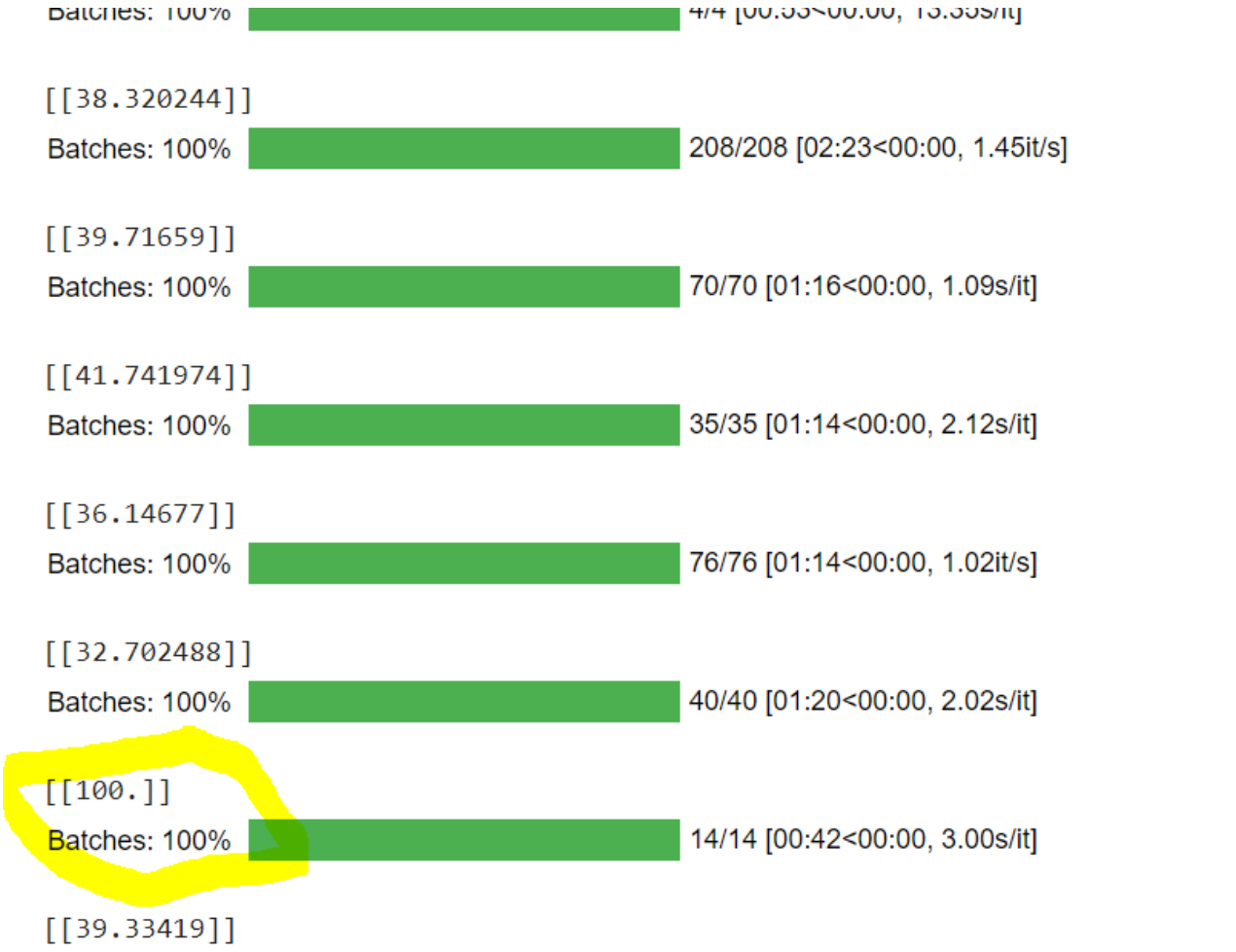
Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

After this is performed, based on how similar or different text is in comparison to other text and the area of similarity, the point of citation is noticed and the text it is cited from is also noted.

Experimentation and Observation.

While preliminary work was done with just the first five links returned after our summarized sentence is searched the bulk of this project was done with a database created.

First, a text found in this database was run and two outputs were generated. One is the overall similarity score between that text and all the text in the database but specifically the identical text and second the line by line comparison between the text and other text in the database.



```
[[0.99999994 0.5643122  0.42146313 ... 0.35663822 0.19377151 0.6031613 ]
 [0.5643122  1.         0.54403406 ... 0.33779398 0.27794272 0.5956769 ]
 [0.42146313 0.54403406 0.99999994 ... 0.35586116 0.29006428 0.27631056]
 ...
 [0.35663822 0.33779398 0.35586116 ... 1.         0.6930569  0.45465696]
 [0.19377151 0.27794272 0.29006428 ... 0.6930569  1.0000002  0.31996465]
 [0.6031613  0.5956769  0.27631056 ... 0.45465696 0.31996465 1.        ]]
```

Batches: 100% 22/22 [00:58<00:00, 2.66s/it]

Next, we did some experiments with one line from the earth text. First to identify when there is a match and second to identify how the similarity score is affected when the sentence is paraphrased slightly and heavily.

```
·th.txt
).50778866 0.41121352 0.50722545 0.31692114 0.34296387 0.4091797
).22033796 0.48777196 0.37668344 0.47760183 0.36264095 0.5778944
).3506487  0.5081847  0.38160142 0.3731354  0.39211586 0.34258586
).38657945 0.30849394 0.34103262 0.6068698  0.44629884 0.45855504
).5612576  0.44646877 0.4953931  0.98573786]]
```

tches: 100% 4/4 [00:20<00:00, 5.09s/it]

```
·th.txt
).5135803  0.41251576 0.5112641  0.3408358  0.37912446 0.40606374
).17474493 0.51006025 0.37108672 0.4879992  0.39911062 0.5665458
).36585814 0.4547378  0.3809467  0.3844629  0.40631676 0.3511634
).4130764  0.34048414 0.34242007 0.5558383  0.45925358 0.42073697
).56429493 0.4255757  0.50799227 0.95332545]]
```

tches: 100% 4/4 [00:18<00:00, 4.69s/it]

In the images above, you discover how the similarity score changes as the sentence is paraphrased but the similarity still stands out as compared to similarity score with other sentences.

Finally when the sentence is paraphrased heavily, the similarity score drops to 0.81 but is still well above other unsimilar sentences.

```
earth.txt
[[0.2908904  0.272767   0.35175902 0.23270747 0.28067356 0.31759304
  0.1899782  0.3205531  0.28355482 0.53662187 0.24667448 0.33806887
  0.2641599  0.29892316 0.26568753 0.24092934 0.3005196  0.30196685
  0.31311008 0.21099046 0.23496825 0.41433042 0.28227425 0.3112123
  0.42280185 0.34691885 0.35661766 0.8122294 ]]
Batches: 100% 4/4 [00:22<00:00, 5.74s/it]
```

Putting this in context of a full text, we created a text with lines from different documents and tried to assess the performance. But this time we set certain parameters.

First, we only returned points of match that had similarity scores up and above 0.85 for at least one line.

```
Batches: 100% 4/4 [03:01<00:00, 45.33s/it]

earth.txt
Tuple of arrays returned :  (array([0, 0, 1, 1, 1, 2, 3, 4, 5, 6, 7, 8]), array([16, 19,  1,  5, 20, 21, 22,
Batches: 100% 4/4 [00:38<00:00, 9.71s/it]

moralsandethics.txt
Tuple of arrays returned :  (array([ 9, 10, 11, 12, 13, 14]), array([ 7,  8,  9, 10, 11, 12]))
Batches: 100% 40/40 [02:52<00:00, 4.32s/it]

kkkk.txt
Tuple of arrays returned :  (array([10, 15, 15, 15, 16, 17, 18, 19, 20, 21, 22, 23, 23, 23, 23, 24, 25,
       26, 27, 27, 28, 29, 30, 32, 32, 32, 32]), array([188,  41, 199, 288,  42,  43,  44,  45,  46,  47,  48,
        75,  83,  50,  51,  52,  53,  89,  54,  55,  56,   6, 136, 137,
       155]))
Batches: 100% 208/208 [02:24<00:00, 1.44it/s]

Neural.txt
Tuple of arrays returned :  (array([15, 31, 32, 33]), array([276, 235,  18,  19]))
Batches: 100% 35/35 [00:31<00:00, 1.09it/s]
```

Second, we only returned points of match with similarity scores 0.8 and at least two lines of match.

```
Batches: 100% ████████████████████ 71/71 [17:52<00:00, 15.10s/it]

Erika Rappaport, “Object Lessons and Colonial Histories: Inventing ...
0.6314110475427964
Tuple of arrays returned :  (array([], dtype=int64), array([], dtype=int64))
Batches: 100% ████████████████████ 796/796 [17:09<00:00, 1.29s/it]

The Interpretation of Cultures: Selected Essays
0.7180364272173714
Tuple of arrays returned :  (array([12, 13, 30, 30]), array([2294, 4784,  274,  338]))
Batches: 100% ████████████████████ 1275/1275 [15:25<00:00, 1.38it/s]

Chapter 19: Industrialization and Nationalism, 1800-1870
0.7024958554436179
Tuple of arrays returned :  (array([], dtype=int64), array([], dtype=int64))
Batches: 100% ████████████████████ 1297/1297 [26:19<00:00, 1.22s/it]

University of Warwick institutional repository: http://go.warwick.ac.uk ...
0.4718190361471737
Tuple of arrays returned :  (array([], dtype=int64), array([], dtype=int64))
```

The concept of these two options is that, while citations may not always be in multiple lines but could easily be in one the measure for identifying a citation should be a bit stricter while on the on other had the hand the presence of several lines of match indicates a very high likelihood of a citation.

At the end of the comparison, the name of text with matches is returned as well as the lines on which they matched in both the original and the cited text and an overall similarity score is generated to identify how similar these two texts are.

While the 5 most similar text are also returned.

```
a,b =areas_sim['earth.txt']
for u in range(len(a)):
  l=a[u]
  print(sentencesCit[l])
```

```
 Continent map of late Triassic fossil sites
 Vertebrate fossils from the Late Triassic have been found at a number of sites around the world, some of which are marked (bla
 Continent map of late Triassic fossil sites
 Vertebrate fossils from the Late Triassic have been found at a number of sites around the world, some of which are marked (bla
New dating of rocks at sites in South America and Greenland pinpoint when long-necked dinosaurs known as sauropodomorphs migrat
New dating of rocks at sites in South America and Greenland pinpoint when long-necked dinosaurs known as sauropodomorphs migrat
New dating of rocks at sites in South America and Greenland pinpoint when long-necked dinosaurs known as sauropodomorphs migrat
DENNIS KENT AND LARS CLEMMENSEN
 That more precise date for the sauropodomorphs’ migration may explain why it took them so long to start the trek north – and h
Around the time that sauropodomorphs appeared in Greenland, carbon dioxide levels plummeted within a few million years to 2,000
The reason for this drop in carbon dioxide – which appears in climate records from South America and Greenland – is unknown, bu
“We have evidence for all of these events, but the confluence in timing is what is remarkable here,” says Morgan Schaller, a ge
These new findings, he says, also help solve the mystery of why plant eaters stayed put during a time that meat eaters roamed t
“This study reminds us that we can’t understand evolution without understanding climate and environment,” says Steve Brusatte,
“Even the biggest and most awesome creatures that ever lived were still kept in check by the whims of climate change.”

 Morality
 Many people find morality extremely useful.
```

```
print(areas_sim)
```

```
{'earth.txt': (array([0, 0, 1, 1, 1, 2, 3, 4, 5, 6, 7, 8]), array([16, 19,  1,  5, 20, 21, 22, 23, 24, 25, 26, 27])), 'moralsar
       26, 27, 27, 28, 29, 30, 32, 32, 32, 32]), array([188,  41, 199, 288,  42,  43,  44,  45,  46,  47,  48,  38,  49,
        75,  83,  50,  51,  52,  53,  89,  54,  55,  56,   6, 136, 137,
       155])), 'Neural.txt': (array([15, 31, 32, 33]), array([276, 235,  18,  19]))}
```

Further work.

1. The concept of privacy should be looked into further. It was a user requirement that no aspect of the uploaded text be recorded. While this was done, there is a need to further protect the user's uploaded text.
2. In other for this project to perform properly, it should be run always against the web, which provides a larger scope and database for comparison. Unfortunately, this may introduce significant challenges but it is an essential part that should be considered,