DLT2: Dating Latin Texts with Deep Learning Techniques

Mitchell Allen and Paul Hoffmann

University of Colorado - Boulder

Abstract. In this paper, we introduce DLT2, a model that can date Latin texts through deep learning techniques in order to assist archaeologists, classicists, and other specialists. Latin is a classical language with a long, rich history, and as such, there are many significant texts for which a date cannot be determined precisely, if at all. The model is trained on a corpus of Latin texts with dates of writing that can be estimated within a 50-year window. These texts are divided into eight distinct "eras" based on recognized developments in the historical development of Latin and its societal contexts. The model is then able to suggest the most likely era for an undated Latin text. After ensuring that metadata such as dates, authors' names, and titles are removed from the beginning of each document, DLT2 is able to achieve an overall prediction accuracy of 30.88% in classifying texts with known dates, with some era-specific accuracy scores reaching as high as 66.85%. ...

Keywords: deep learning, text dating, natural language processing, NLP, Latin, corpus linguistics

1 Introduction

1.1 Motivation

Latin is a classical language that was spoken in Rome. Due to the Roman Empire's expansion, it spread throughout Europe as well as parts of Asia and Africa. The language was widely employed as a top choice for international communication until the mid-18th century, when it was gradually supplanted by English, French, and others. It is still used today by the Catholic Church, and it has influenced many modern languages and academic disciplines through loanwords and specialized terminology. However, despite its long and influential history, the date when many important Latin texts were written is uncertain or completely unknown. If these texts could be dated with a higher degree of precision, new implications could be drawn that would have a sizable impact on many diverse areas of study, including history, anthropology, linguistics, and literature, as well as other domains.

There are some broad orthographical and morphological changes in Latin that can be traced to particular periods in history, and these, in conjunction with content clues regarding the text's context, can currently be used to estimate the date of a text within several centuries, but a more sophisticated model may be able to pick up on even more nuanced changes in vocabulary, syntax, and other linguistic features in order to provide a more precise dating. We believe that such a tool could build off of the core functionality of dating undated texts to fuel other related applications, such as detecting forgeries and identifying unknown authors. However, these remain outside the scope of this project. The core focus of DLT2 at present is to provide an accurate estimation of the date of composition for a given work in Latin.

2 Related Work

2.1 Dating Texts

As far as we are aware, deep learning has not been used before to date Latin texts, although there have been several projects attempting to date texts in other languages, including Chinese[1], Sanskrit[2], and Ancient Hebrew.[3]

Tian achieves 60% accuracy in classifying Chinese texts into four era classifications, although the later two are clearly distinguishable, whereas the earlier two are more often confused. [1] This may indicate that differences between the first two eras are not as prominent, and they should potentially be reclassified or merged by sinologists in order to more accurately reflect the linguistic reality. DLT2 also struggles more with particular eras, which will be discussed further in Section 5.

The most prominent of these efforts in other languages may be Deep Mind's "Ithaca" for Ancient Greek, which can date texts within 30 years of their actual written date on average, as well as provide an estimated geolocation for inscriptions and restore gaps in texts with missing or damaged manuscripts.[4] Although these functionalities are beyond the scope of the current project, they give the closest approximation of the benefits that DLT2 could provide if it were to be eventually developed to its full potential.

2.2 Latin NLP

We had formerly intended to lemmatize our corpus, which would have required the extension of the current functionality of several NLP tools, such as CLTK[5] and Lamonpy[6], that are specifically designed to process Latin. However, we have chosen to modify our approach so that only tokenization is required, and the functionality of existing tools will accomplish the needed tasks for DLT2.[7] If a future iteration of DLT2 were to require lemmatization, the Latin lemmatizer developed by John Snow Labs for use with Apache Spark could serve as an alternative to modifying other prebuilt tools.[8]

Our corpus data will be sourced from the Latin Library[9], an online repository that contains over 2,200 Latin texts in the public domain, roughly 1,900 of which can be dated precisely. This collection covers nearly all relevant Latin texts aside from fragments and very short songs or poems, so increasing this data will require artificial upsampling on our part. To improve our model's text processing capabilities, DLT2 will incorporate pretrained, 100-dimension word embeddings from the LiLa: Linking Latin project.[10]

3 Methods

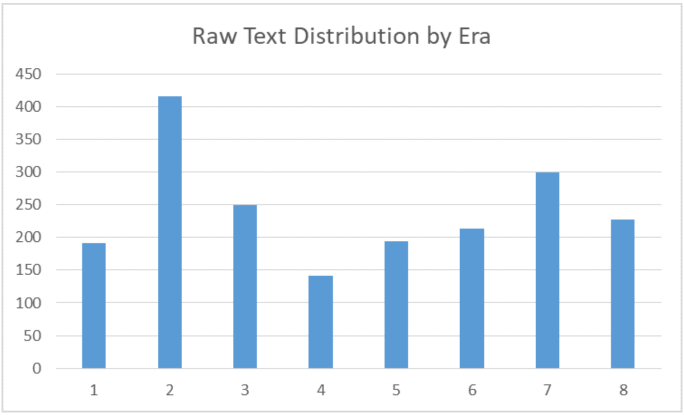
3.1 Corpus

Our corpus, sourced from the Latin Library, contains 2,220 texts, which constitute the vast majority of the Latin texts available from antiquity as well as numerous texts from Latin's later history.[9] Based on their respective dates of composition, each text is categorized into one of the eight eras enumerated below. These historical divisions approximate those established by classicists due to sociological changes in the cultural contexts in which Latin was spoken, and these contextual changes typically engendered some degree of change within the language itself.

Old Latin	700 BC - 75 BC
Caesarean Latin	75 BC - 25 AD
Classical Latin	25 AD - 200 AD
Imperial Latin	200 AD - 400 AD
Late Latin	400 AD - 500 AD
Vulgar Latin	500 AD - 900 AD
Medieval Latin	900 AD - 1300 AD
New Latin	1300 AD - Present

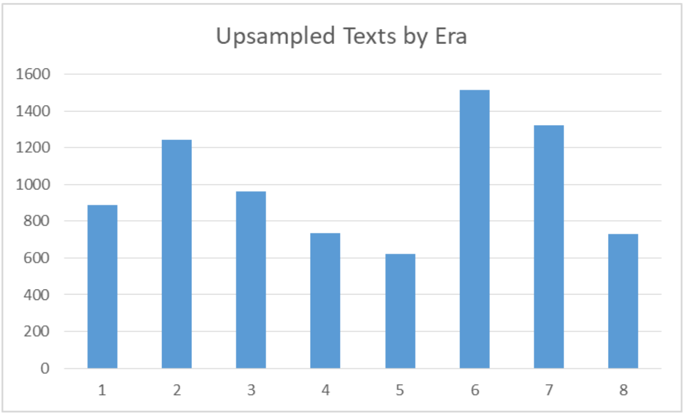
3.2 Data Preparation

Raw Dataset From this baseline corpus, we removed any texts with unknown dates or proposed dates that ranged larger than 100 years. This eliminated 265 texts from the original corpus. The remaining 1,935 texts were processed with regex to remove extraneous words, lines, and punctuation. The bar graph below details the number of texts in the corpus that could be categorized into each era, with 1 on the left corresponding to Old Latin and progressing through later eras rightward until New Latin at 8.



Balanced Dataset However, using this imbalanced data led to the model underfitting and predicting Caesarean Latin (Era 2) for every text. The significantly larger portion of texts from Caesarean Latin meant that DLT2 could still achieve a relatively high level of accuracy by doing so. In order to prevent this underfitting, we removed a random selection of texts from each of the more populated eras until each of the eight eras contained an equal 141 texts apiece. This allowed us to use a balanced dataset of 1,128 texts and force DLT2 to rely on deeper information for its classifications.

Upsampled Dataset In order to increase the number of texts available in each era, we also created an upsampled dataset. We filtered out the first 100 characters from each text in order to remove metadata such as names of authors, dates, and titles that were typically present at the beginning of the document, albeit not in any standard format. After these were removed, each text was divided into 10,000-character chunks, and any leftover chunks that were less than 10,000 characters were discarded. Each of these chunks was saved into a new text file, thereby increasing the number of unique texts that the model could analyze for each era while still ensuring that each file contained unique and authentic Latin writing. Being able to utilize authentic and unique spans of Latin text rendered this approach preferable to other means of oversampling, such as simply repeating duplicated texts or attempting to artificially generate original texts from existing data. This new split resulted in 8,022 total texts, distribution for this upsampled dataset is depicted below. Interestingly, Vulgar Latin (Era 6) and Medieval Latin (Era 7) are now more heavily populated than Caesarean Latin (Era 2).



Rebalanced Dataset Again, in order to guard against underfitting, we removed a random selection of texts from each of the more populated eras until each of the eight eras contained an equal 623 texts apiece, totaling 4,984 texts. This rebalanced dataset allowed us to supply the model with a larger sample size from which to draw its observations.

3.3 Data Processing

We created a CSV file that contained two columns: one with file names of each of the texts and the other with a number between 1 and 8 depending on its respective era. From there, we created an algorithm that pre-processed each text. The algorithm used the Numpy, Pandas, and Regex packages for Python. The algorithm read in the CSV file and saved only the column of names of the Latin text as Pandas arrays called Texts and Names. The Texts array gets overwritten after the algorithm finds each iteration of Latin text and saves the whole text into the appropriate array location as a string.

After this, we remove the first 100 characters of each text, since that contains the title, author, and sometimes the date of the texts. This step is only necessary for the Raw and Balanced Datasets due to these characters already being removed during the creation of the Upsampled Dataset. We do not want the model to rely on those pieces of information but instead on actual linguistic details present in the body of the text. From there, we pass each text into a Regex function that cleans the texts by removing brackets, numbers, tabs, and punctuation.

Following our initial trials with the Raw and Balanced Datasets, we determined that a larger sample size would likely be helpful for DLT2's classification efforts. However, since our corpus already contains nearly every Latin text that would be useful for our purposes, we came up with a solution to split longer texts into 10,000-character chunks, as discussed in Section 3.2.3, which resulted in creating a corpus that was four times greater than what we started with. With our new corpus created, the algorithm updated the array called Names with the newly created sub-texts and saved it into a new CSV file, "Rank", that will be fed into our model.

3.4 Hardware

Processing power has proved to be a major limiting factor for the model's performance. While running the model locally, a Lenovo P50 GPU: Nvida Quadro M1000m was used. In order to enhance the amount of data and number of parameters later on, we also used a Google Cloud GPU: T4.

TensorFlow proved to be a major hindrance due to the complexity of setting it up correctly with graphic drivers. To run it on our local machine, we needed to become NVIDIA Developer members in order to download the right Cuda and cuDNN. Also, when running the model on Google Cloud, we ran into problems using the latest Cuda version, 11.6. To fix this, we had to create a whole new virtual machine that used Cuda 11.0. We also needed to call some TensorFlow packages differently due to the version.

3.5 Building the Model

The model algorithm reads in the CSV file called "Rank" the same way the pre-processing algorithm read in the other CSV file. Each text was then passed through a Tokenizer and padded in the sequence function to ensure that each text's matrix was the same shape. The respective labels, dates in this case, were passed in via one-hot encoding.

Creating the model proved to be the most difficult part due to the fact that we are dealing with long strings of text. As mentioned in the Hardware section, limiting the number of characters fed into the model became necessary due to maxing out GPU memory. Also, the batch size needed to be small due to the same issue; we opted to keep it at 32 for all of our trials.

After initially trying many simple models with only Dense layers and making small adjustments, we noticed that having more layers did not make the results better. From there, we kept the model small and added other types of layers. We added an embedding layer to the beginning of the model. The embeddings we used were pre-trained using a Latin Word2Vec set from LiLa.[10] This increased our accuracy, which makes sense due to the word embeddings containing real-valued vectors in a vector space that represent individual words. Each word is mapped to one vector, and the vector values are pre-learned and connect well in neural network models like ours.

We wanted to take another step to improve accuracy by adding an LSTM layer. Since our problem is based in NLP, LSTM is a great fit due to its superiority in signal processing and capability of resolving the vanishing gradient. The LSTM layer did improve our results and kept the model small with fewer layers. Changing the basic LSTM to a bidirectional LSTM also led to large improvements due to the additional training of the data.

After several iterations of adjusting and fine-tuning these layers, we ended up with a four-layer model. The first layer is the embedding layer of size 300, which feeds into two bidirectional LSTM layers of size 1050. The last layer is a fully connected layer with the Softmax activation function of size 9, due to the nature of our project as categorical learning.

4 Experimental Design

The texts from the corpus are shuffled and split into a 70% training set and a 30% test set. Our loss function uses "Categorical Crossentropy", and the optimizer is "Adam". When training the model, we kept the epochs to 5 and batch size to 32. The batch size is small because of our hardware limitations. If we wanted to use a batch size of 128, we would need to have access to at least 30 GB of video memory. We also kept the number of epochs generally low due to the amount of time the model requires to run. For instance, 5 epochs takes about two hours to run.

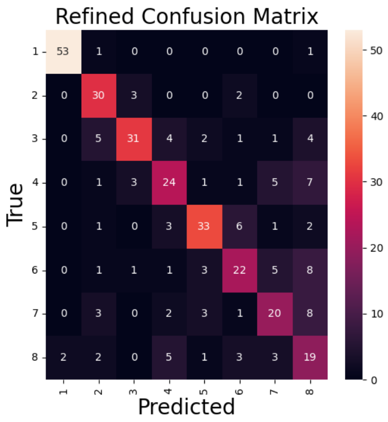
When searching for the model we wanted to use for full data set, we looked at test accuracy and as we ran every trial as this measures how often the model has correctly identified the era in which a particular text was written. Printing a confusion matrix helped us see how the model performed in predicting each category, which alerted us to underfitting in earlier iterations of the model, as stated in Section 4.2. This also allows us to analyze which eras are most often confused for others. This can be further broken down into determining whether misclassification attempts are "near misses" categorized into chronologically adjacent eras, or if these incorrect classifications are more evenly spread across history.

We kept the data input simple for the initial model trials: 100 characters, 25 epochs, and a batch size of 32 with the Balanced Dataset. After we realized that the first 100 characters of many texts provided excessively helpful metadata, we removed these characters and focused on the next 10,000 characters with the same batch size of 32, 25 epochs, and the Balanced Dataset. We then performed a third experiment with identical hyperparameters on the Rebalanced Dataset. When these results suggested overfitting, we tried a wide variety of different hyperparameters that did not lead to any success: the model was still underfitting despite the dataset being balanced. However, we did finally experience greater success after lowering the character window to 1,000 in our last set of experiments.

5 Experimental Results

5.1 Initial Trial: Balanced Dataset, 100 Character Window

The confusion matrix below shows the results from our best model during the first set of trials on the Balanced Dataset. These trials only looked at the first 100 characters, and as such we believe the success shown here to be artificially inflated by extratextual information.



Trial 1 Scores	
Test loss:	1.8099
Test accuracy:	68.44
Test f1 score:	0.69
Test precision:	0.72
Test recall:	0.66

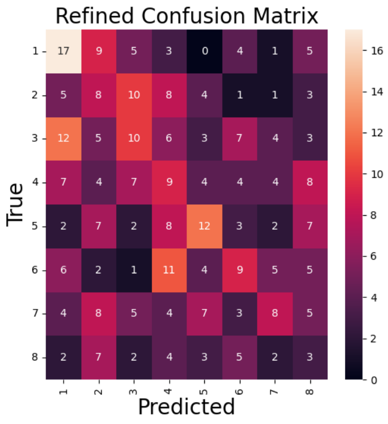
It is helpful to note that for many eras, even when it classifies a text incorrectly, it is only off by an era or two. This seems to suggest that the model has successfully learned some salient features in the text, but the window it has narrowed the text down to remains somewhat broader than the given era still, forcing it to make a guess.

Later eras of Latin likely see higher confusion rates due to the language itself becoming less frequently spoken. After the decline of the Roman Empire, Latin became increasingly used only in academic, ecclesiastical, and literary contexts that typically aimed to imitate or revive forms of Latin from earlier eras. Thus, it is likely that texts from these eras bear inconsistent characteristics that at times may resemble almost any of the earlier eras of Latin. These inconsistent and artificial stylistic tendencies thereby seem to confuse the model, leading to lower accuracy scores for the later Latin texts.

The first 100 characters at the beginning of each text proved to be too helpful, since they usually contained the author's name. Since authors often wrote multiple texts within the corpus, and these would naturally fall within the same era, we believe the model is leaning too heavily on this information instead of picking up on diachronic linguistic cues. Thus, due to the presence of such helpful metadata within the window that was analyzed, we needed to exclude the first 100 characters from our evaluation window in the next round of trials.

5.2 Second Trial: Balanced Dataset, 10,000 Characters

For this round of experiments, we trained the model on the first 10,000 characters of the texts in the Balanced Dataset after the first 100 characters were excluded to remove the metadata. The confusion matrix from our best model during these trials is shown below.

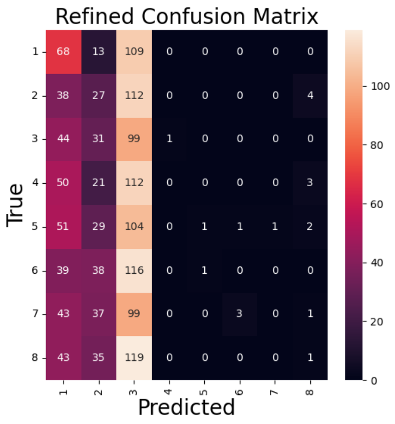


Overall, this model's 24% accuracy beat the random baseline of 12.5%, although this is inflated by a strong performance in classifying Old Latin, which is arguably the most distinct of the eight era. Late Latin performs the best of the rest, but none of these do very well still, so there doesn't seem to be much in the way of meaningful conclusions that can be drawn from this on its own.

Since the trials with this approach produced a 60% training accuracy and 24% test accuracy, we suspected that increasing our data would improve our model by giving it a larger sample size to work with, so we plan to split texts into 10,000-character chunks, provided they were long enough, in order to upsample our text amounts, particularly for the less populated eras.

5.3 Third Trial: Rebalanced Dataset, 10,000 Characters

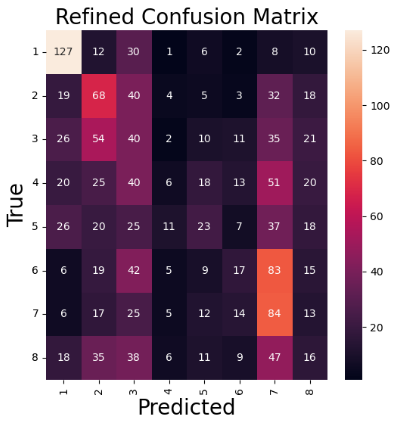
For this trial we used the Rebalanced Dataset to supply the model with 4,984 texts of size 10,000 characters. We again trained for 25 epochs with a batch size of 32. However, we were surprised to find that the model began underfitting again despite the dataset being distributed equally. This can be seen in the confusion matrix below.



Our test accuracy of 13% is roughly equivalent to what we would expect in completely random guessing, so these are very poor results for this model. Since the model is at least guessing Old Latin and Caesarean Latin in addition to a heavy tendency toward Classical Latin, it may be that these eras are much more distinct from the rest, and the model is simply becoming overwhelmed by the data being too large at this point. This is reinforced by the fact that, despite attempting to make wide-ranging adjustments to hyperparameters, these results only improved in the following trial after the character count was restricted again.

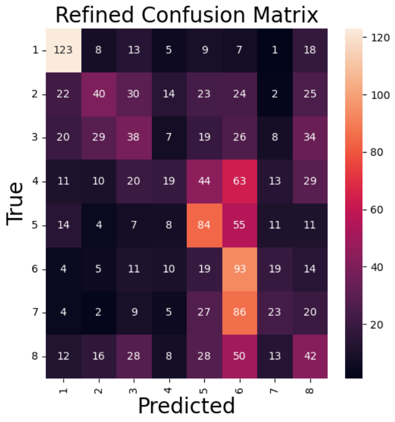
5.4 Fourth Trial: Rebalanced Dataset, 1,000 Characters

Keeping all else the same as the third trial, we lowered the number of epochs to 5 to increase computing speed and restricted the 10,000-character window to 5,000 and 2,000 characters, seeing only slight improvements. However, when we limited the window to 1,000 characters, the model suddenly achieved much more success, as seen below.



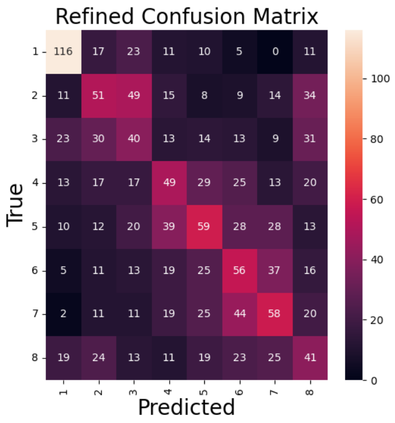
Trial 4 Scores, 5 Epochs	
Test loss:	2.0144
Test accuracy:	25.47
Test f1 score:	0.13
Test precision:	0.45
Test recall:	0.08

With this initial success, we again kept everything else the same but increased the epochs back to the standard amount of 25, which generated the results below.



Trial 4 Scores, 25 Epochs	
Test loss:	2.3589
Test accuracy:	30.88
Test f1 score:	0.25
Test precision:	0.35
Test recall:	0.20

These results are much more promising, especially for Old Latin, which achieves 66.85% accuracy on its own. Missed guesses appear to be much more closely clustered to the true era, particularly for Medieval Latin, which may suggest that such a classification ought to be merged with Vulgar Latin into a broader category of its own. In a final experiment, we kept everything else the same but increased the epochs to 50, which generated the results below.



Trial 4 Scores, 50 Epochs	
Test loss:	4.8734
Test accuracy:	31.42
Test f1 score:	0.31
Test precision:	0.32
Test recall:	0.30

Increasing the number of epochs does lead to a slight improvement, although it doubles the amount of training time that must be spent. In future research, we would likely remain at 25 epochs until we felt we had a finalized model to warrant to longer training time. We would also like to perform another round of upsampling in the future to divide all texts into these 1,000-character chunks and further boost the sample size for the model's training. Based on the trends observed in these four major rounds of trials, we believe that this would lead to even stronger classification performance from DLT2.

6 Conclusions

6.1 Summary

We noticed that using shorter texts and a greater number of texts generally led to better model results. If we had more time, it would be very interesting to continue playing with the amount of characters to see if we could achieve even better results with this approach. We would also like to figure out why is ”less is more” in this case. It may be due to our data type, long form texts, consisting of extremely diverse verbal information that provides too much complexity for our model when processed in larger windows. Regardless, we are pleased with this initial foray into automated Latin text dating, and we are optimistic for its future improvement.

6.2 Ethical Implications

A tool such as DLT2 can have an outsized impact on society at large through its assistance of a wide range of specialists across many domains. If undated Latin texts could be dated with a higher degree of precision, these details would enable new conclusions to be drawn with much greater certainty about the nature of their composition and the broader contexts in which they were written. As such, every effort has been made to ensure that open source tools have been utilized in the development of DLT2 so that it too may render a readily accessible service to other future researchers.

References

1. Tian, Z.: Automatically dating classical chinese texts: Preliminary study on biji and buddhist texts (2022)
2. Hellwig, O.: Dating sanskrit texts using linguistic features and neural networks. Indogermanische Forschungen **124**(1) (2019) 1–46
3. Liebeskind, C., Liebeskind, S.: Deep learning for period classification of historical hebrew texts. Journal of Data Mining Digital Humanities (2020)
4. Assael, Y., Sommerschield, T., Shillingford, B., Bordbar, M., Pavlopoulos, J., Chatzipanagiotou, M., Androutsopoulos, I., Prag, J., de Freitas, N.: Restoring and attributing ancient texts using deep neural networks. Nature **603** (2022) 280–283
5. Johnson, K.P., Burns, P.J., Stewart, J., Cook, T., Besnier, C., Mattingly, W.J.B.: The Classical Language Toolkit: An NLP framework for pre-modern languages. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, Online, Association for Computational Linguistics (August 2021) 20–29
6. bab2min: bab2min/lamonpy: 0.2.0 (October 2020)
7. Chollet, F.: keras. https://github.com/fchollet/keras (2015)
8. Labs, J.S.: Latin lemmatizer. https://github.com/johnsnowlabs/spark-nlp/tree/master/docs/$_posts/maziyarpanahi/2020-07-29-lemma_la.md$(2020)
9. Carey, W.L.: The latin library. Website
10. Sprugnoli, R., Passarotti, Marco Moretti, G.: Vir is to moderatus as mulier is to intemperans - lemma embeddings for latin. CLiC-it 2019 - Sixth Italian Conference on Computational Linguistics (2019)