Article

Discovering the arrow of time in machine learning

J. Kasmire [1,†,‡] and Anran Zhao [1,‡]

[1] UK Data Service and Cathie Marsh Institute, University of Manchester, UK ; j.kasmire@manchester.ac.uk
[2] UK Data Service and Cathie Marsh Institute, University of Manchester, UK; ofzhaoar@gmail.com
* Correspondence: j.kasmire@manchester.ac.uk
† Current address: Humanities Bridgeford Street, University of Manchester, Oxford Road, Manchester, M13 9PL
‡ These authors contributed equally to this work.

Abstract: Machine learning (ML) is increasingly useful as data grows in volume and accessibility as it can perform tasks (e.g. categorisation, decision making, anomaly detection, etc.) through experience and without explicit instruction, even when the data are too vast, complex, highly variable, full of errors to be analysed in other ways [1,2]. Thus, ML is great for natural language, images, or other complex and messy data available in large and growing volumes. Selecting a ML algorithm depends on many factors as algorithms vary in supervision needed, tolerable error levels, and ability to account for order or temporal context, among many other things. Importantly, ML methods for explicitly ordered or time-dependent data struggle with errors or data asymmetry. Most data are at least implicitly ordered, potentially allowing a hidden 'arrow of time' to affect non-temporal ML performance. This research explores the interaction of ML and implicit order by training two ML algorithms on Twitter data before performing automatic classification tasks under conditions that balance volume and complexity of data. Results show that performance was affected, suggesting that researchers should carefully consider time when selecting appropriate ML algorithms, even when time is only implicitly included.

Keywords: machine learning; time; naive bayes classification; recurrent neural networks, Twitter; social media data; automatic classification

1. Introduction

Machine learning (ML) is a broad church. Supervised ML is very popular for exploring the relationship between clearly defined input and output variables. ML methods include: linear regression which handles continuous output by finding a line or plane that best fits the data, logistic regression which makes classification predictions for binary output, and support vector machines which classify by finding a hyperplane or boundary that maximises the margin between two or more classes. As an example, a supervised ML algorithm might be trained to classify incoming email based on the sender, subject line or message content (input) that are accurately labelled as SPAM or NOT-SPAM (output). In contrast, unsupervised ML models attempt to draw inferences and find patterns from unlabelled data sets, such as finding neighbourhood clusters within demographic information. Common clustering methods include k-means clustering, which groups input into a pre-determined k number of groups, and hierarchical clustering, which arranges observations into tree-like graphs called dendrograms. For instance, an unsupervised ML algorithm might be asked to classify research articles into disciplines according to their similarity. There are also semi-supervised approaches to machine learning, so the distinction is not always clear cut.

ML algorithms and tasks are diverse, but there are some general features that make ML useful. For example, supervised ML algorithms allow data to be classified when the volume or complexity exceeds the capacity of manual classification or non-learning algorithmic exploration while unsupervised ML algorithms are extremely useful for real-time data exploration and pattern detection in data that changes too quickly or too unpredictably for conventional exploration. ML (supervised and unsupervised) are often understood to perform better with more data. Nevertheless, there are trade-offs as unrepresentative data

Citation: Kasmire, J.; Zhao, A. Discovering the arrow of time in machine learning. *Preprints* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 August 2021 doi:10.20944/preprints202108.0516.v1

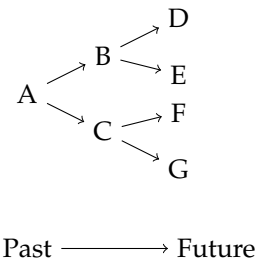

Figure 1. Path dependency illustrated in a simple graph; travel between points only moves in one direction (toward the future) after which previously accessible points (alternate futures) are longer available.

can lead to 'over-fitting' and poor generalisability, which is when a ML algorithm performs very well on the training data but poorly on other data. ML is also typically understood to be tolerant of errors in the data, with some approaches showing acceptable performance despite error rates up to 39 percent [1,2]. ML methods usually require substantial calculation, but this can be managed through careful selection of algorithms and setups [3]. For these reasons, ML is understood to be a good tool for the large, complex and messy data that is ballooning too rapidly for manual or conventional means of exploration, classification or analysis.

This paper begins with some background on time, machine learning and the intersection of the two. Following this is a clear description of the research question to be addressed and the methods and data used to address it. Next is the research results and a discussion of how the results relate to the research question. Finally, a brief discussion of what the results might mean for the wider ML research context and what potential next steps for this research topic.

1.1. The "arrow of time"

The "arrow of time" concept comes from 1927 theoretical physics lecture and describes the apparent one-way flow of time observed in irreversible processes at macroscopic levels. Although concluding that the arrow of time is a consequence of entropy, it was also described as a matter of perception, reasoning and statistical approaches [4]. Put another way, the one-way flow of time was understood to have a thermodynamic origin and psychological consequences. Thus, the arrow of time applies to irreversible processes that are both physical (e.g. an egg cannot be unscrambled) and economic or social (e.g. decisions cannot be undone and done differently instead), often known as path dependency or lock-in [5].

More recent work on the arrow of time maintains that people perceive and experience time as flowing in one direction, but highlights how the concept has continued to develop. Entropy is now linked to information and complexity [6] although the concepts and their applications are not interchangeable [7]. Currently, time's arrow is thought to be a key driver of thermodynamic processes, biochemical processes like evolution via natural selection, and informational processes in both the natural world and cultural systems like language [6,8].

Setting aside the complicated concepts of information (especially as they related to entropy and thermodynamics), the arrow of time can be found in language; utterances [1] are spoken or written at a specific point in time. Further, language sequentially orders its components (symbols, semantic features, words, etc.). These temporal aspects of language are not always strict, allowing some meaning to be communicated 'out of order'. Still, exactly what is communicated may not be "the same" as what would have been communicated if the utterances had been experienced at a different time or in another order. For example, classic literature is read and reread continuously with frequent new

[1] Utterances are usually understood to be exclusively spoken, but this research counts tweets as utterances.

 doi:10.20944/preprints202108.0516.v1

interpretations, all of which are probably unlike the contemporary interpretations from when the literature was originally available. Another example would be how readers may or may not struggle to understand the plot of a book series were they to read the sequel before the original. Likewise, some sentences can be understood more clearly than others when the words are jumbled; 'eats mouse cheese' is relatively clear while 'eats man shark' remains ambiguous.

The difference in interpretations or in interpretability described above derives from shared knowledge or common ground, which accumulates gradually as a consequence of experience [9]. Once learned, common ground allows communicators to recruit previously gained knowledge into helping interpret current utterances. Thus, books in a series can be read out of order if they deal with well-known topics, plots, tropes, events or characters, even if they are not always interpreted in the same way. Likewise, individuals could use common ground to assume that the mouse is eating the cheese in the first jumbled sentence (rather than the cheese eating the mouse) but might not be able to make assumptions about the second without more context (as humans and sharks are both capable of eating each other).

But common ground is also built between individuals through repeated interactions [9] and within a single interaction as participants take turns to speak and check their understanding [10]. Meaning is even built within a single utterance as linguistic input is incrementally processed word-by-word [10], which can create problems when an incremental understanding is violated by a 'garden path' sentence [11]. As a consequence, language use and comprehension are affected by prior exposure to language, most obviously in childhood language development where studies show neurological differences as a result of language exchanges and linguistic turn-taking [12].

1.2. Time in ML

ML includes supervised and unsupervised methods and can perform tasks such as prediction, classification, clustering, search and retrieval or pattern discovery [13]. Familiar examples of applying ML to temporal or otherwise ordered data include predictive text functions that suggest complete words based on the sequence of letters entered so far and storm warning systems based on weather data. Comparative studies have found that different ML algorithms perform differently on time-series classification [14] and explicitly temporal tasks [15]. Neural network ML models generally outperformed classic linear analysis techniques and an analysis of 13 more modern ML time series methods showed multilayer perceptron and then Gaussian processes were the best across a range of categories and data features [16]. Clearly, ML can be very sensitive to time, with some ML algorithms working with time better than others. Consequently, there are well-established data sets, benchmarks and approaches for temporal data ML research.

Specifically time-sensitive ML approaches have additional problems or weaknesses that other ML methods can avoid. The problems of over-fitting and generalisability are more complicated because the training data must be representative in relation to time as well as all the other ways that it must be representative. The added dimension makes time-sensitive ML approaches less tolerant of errors or missing values than other ML methods and with greater risks of over-fitting to one time frame. This means that time-sensitive ML algorithms typically have additional data requirements, including that it must be organised into uniformly sized time intervals. Although not insurmountable, the stricter data requirements mean that time-sensitive ML is more difficult to apply, and so less popular, unless the data to be analysed or research question are specifically temporal.

Most of the data of interest for ML exploration and analysis is vast, complex and messy. Importantly, that data usually also has a temporal context. First, much of that data is time-stamped or otherwise ordered, although this is not always seen as important. Second, data is unevenly distributed over time as users, platforms and records are added or shared, seriously complicating temporal representativeness. Additionally, features within that data are also unevenly distributed over time. To illustrate, 10-15% of medical diagnoses at any

 doi:10.20944/preprints202108.0516.v1

given point in time are estimated to be wrong [17], although which are erroneous is not known at the time. Some incorrect diagnoses are identified through later testing, condition progression, and medical innovations or discoveries, but the re-diagnosis is also subject to the 10-15% misdiagnosis rate. Further complicating the issue, a correct diagnosis according to the diagnostic criteria of the time could be a misdiagnosis according to later criteria. This becomes clear when we recognise that autism is not childhood schizophrenia [18] or that status lymphaticus is not a real condition [19]. Third, and perhaps most importantly, new data is not created independently of existing data. Instead, data is often generated as the result of learning from previously generated data. For example, users might change what content they share on a social media platform after observing what content is popular or well-received. When considered together, all of this suggests that the data that invites ML analysis is likely to be time-asymmetrical in ways to which many ML algorithms are not well suited.

Researchers have begun to ask how time and ML algorithms interact, both to correct for problems caused by temporal features as well as to capitalise on these features. For example, sentiment analysis of Twitter data is well-established as a ML problem of interest, but accounting for temporal aspects of the data allowed researchers to analyse sentiment in 'real time' tweets [20,21] while combining sentiment analysis of tweets with geo-tagging enabled researchers to build a spatial and temporal map of New York [22]. Further, using social interactions with temporal features allowed researchers to significantly improve stress detection in young people [23], potentially allowed problems to be detected early enough for effective intervention. Researchers have clearly noticed the ways that ML algorithms are affected by time and are beginning to investigate these interactions and even to put them to practical use. Thus, it is important to understand which algorithms are sensitive to implicit temporal features and in what ways these temporal sensitivities affect the ML performance.

1.3. Research Question

This research addresses several related questions. First, is ML performance influenced, biased or complicated by data with subtle or implicit representations of the flow of time? Put another way, will ML algorithms that assume time-independence or time-symmetrical data perform differently when that assumption is violated? This could have consequences for how an appropriate ML algorithm is chosen according to the data available or the task to be performed.

A second problem would revolve around identifying, quantifying or managing the ML effects of data sets that only implicitly capture the one-directional flow of time. Of course, researchers could discard or reshape the data to meet the more stringent requirements of explicitly time-sensitive ML methods, although this would lose many of the benefits of ML approaches. For example, there may be ways to automatically weight or present the data during training, which may help balance the risks between over-fitting and implicit temporal distortion. Alternatively, iterative learning could be introduced so that algorithms learn from already-trained models as well as from the data itself. By approximating the way that the people generating the data learn from past data, such iterative learning may account for the time-asymmetry of the data.

The research seeks to answer whether or not the 'arrow of time' can be observed in ML algorithm performance under various training regimes on data with only implicit inclusion of time. In effect, the research question asks whether 'more data is always better for ML performance?' or whether 'data that spans significant time frames will capture an implicit arrow of time that can distort ML performance?'. To answer that, this research collects tweets from 9 different time frames and trains ML models to predict which time frame those tweets come from. the ML models are trained on subsets of 3 time frames, 6 time frames and all 9 time frames. If the accepted ML concept that 'more training data is always better' is true, then the performance should be equal or better when the models are trained on 6 or 9 time frames in comparison to when they are trained on only 3 time

 doi:10.20944/preprints202108.0516.v1

frames. However, if the data does capture some implicit temporal features that are not well accounted for in these ML models and that disrupt its learning, then the performance of the models trained on more time frames should be worse than those trained on fewer time frames.

Importantly, this research is about a classification task and *not* about a time-series prediction task. Thus, the ML models are not explicitly considering temporal features or order, and time is only present within the model in an implicit way, embedded within the data. In essence, this means that the research question can be rephrased as 'are non-temporal ML models influenced by purely implicit temporal features?'.

2. Materials and Methods

First, two supervised ML algorithms are explored, compared and contrasted. Unsupervised ML algorithms are also interesting, but as a first step in the exploration of implicit temporal representations on ML performance, this research focuses on supervised ML only. Second, the data collection, preparation and application of the two selected methods are described in detail and a GitHub repository with the data sets and code is linked [24]. Finally, the experimental conditions and results are discussed, with clear descriptions of the specific training and testing regimes for each model and the accuracy and precision for each.

Two ML models will be explored in this research: naive Bayes classifiers (NBC) and recurrent neural networks (RNN). Both are supervised ML models, meaning that they learn to perform their tasks through explicit training. This is done by providing the ML models with one or more training data sets that consists of at least one 'input' and at least one 'output' so that they can learn to correctly predict the output from the input. Whether or not the models have learned to do this correctly is achieved through one or more test data sets that have the same structure as the training data sets but that do not contain the same entries. Generally, supervised ML models achieve this by taking one large data set and dividing it so that the majority is allocated to the training data set and the rest to the test data set with no overlap. 'Correct' learning is generally understood as the most accurate and precise performance over the test data as possible, although some research projects may instead seek to specifically focus on minimising either false-positives or false-negatives.

While both models take the same basic approach of learning from explicit training sets before being tested, exactly how they learn and operate is unique to each model.

2.1. Naive Bayes Classification and Recurrent Neural Networks

NBC models learn to associate the inputs with the outputs by creating and subsequently adjusting simple weighted associations. For example, a NBC might be given a training data set of emails (input) which are tagged as 'SPAM' or 'NOT-SPAM' (output). Through training, the NBC would first associate the words, phrases or structures of the emails with their 'SPAM' or 'NOT-SPAM' tags and would then refine the weights of these associations. By the end of training, the words, phrases and structures that are only or mostly found in 'SPAM' emails (or vice versa) would be used to predict whether a new email (input in the test data set) is likely to be SPAM or not. Many NBC models seek to be as accurate and precise as possible, although some models may instead seek to focus on ensuring that all SPAM is correctly identified as SPAM even if this means that some NOT-SPAM are also labelled as SPAM.

RNN and other neural network models also learn to predict an output by learning from the training data by feeding activation through a network or graph of connected nodes and layers. The input is provided to the first layer which then activates the subsequent layer according to the connections and weights, proceeding in this way with the activity of the final layer being interpreted as the predicted probability of output classes. The model then compares the predicted output to the real output in the training data and uses the difference to adjust the network connections and weights so as to reduce the error. Simple

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 August 2021 doi:10.20944/preprints202108.0516.v1

or feed-forward neural networks only accept a single input at a time so the information only flows in one direction but RNN models allow layers to feed their activation back to the previous layer allowing the input at any given step to contain some of the input from the previous step. For example, a feed-forward neural net might be fed a word one letter at a time with the aim of predicting the next letter; if fed 'n', 'e', 'u', 'r', and 'a', it would predict 'n' (statistically the most common letter to follow 'a'). But a RNN with the same task and inputs would predict 'l' because it would look at *all* of the letters in the input sequence rather than only the most recent.

For the research described in this article, the data set has two parts: the cleaned full text of tweets and when the tweet was first made (in year-month format). Both models will be passed the tweets as input and are asked to predict when the tweet was made as output. As a first exploration of the research question, both models will be judged on simple accuracy with no focus on the issue of false positives or false negatives. However, the research question requires more than simply training and testing the models on all of the data. Instead, the ML models are trained and tested on subsets of the test data that encompass 3 time periods, 6 time periods or 9 time periods. This allows the researchers to see if the models perform differently when the training sets are larger but also contain more implicit temporal features. In this way, the models can help illuminate whether more data is better (regardless of the temporal nature of that data) or whether language change or specific events captured within the time frames distort the way ML models perform.

2.2. Data collection

The data set is created via the Tweepy[25] package and Twitter application programming interface (API). The Tweepy *search_full_archive* and *cursor* methods take a start date, stop date, search keywords and an environment parameters as input and returns up to a specified maximum of truncated tweets with tweet ID, sender ID, tweet sent time and other details as output. The Tweepy *status_lookup* method then iterates over the IDs of the truncated tweets, 100 at a time, updating to a full-text version with the option to add other relevant information (e.g. whether the tweet was in reply to another, whether it was favourited, from what source the tweet was sent, etc.) as needed.

For this research, the Tweepy methods described above were used to create nine batches of 1200 tweets. Each batch was collected with the keywords 'vaccine' and 'UK' and each covered a six-month interval between April 2017 and April 2021 for a total of 10800 tweets. For example, one batch retrieved 1200 tweets published between 1 April and 20 April 2021 containing the keywords 'vaccine' and 'UK'.

It is important to note that this method requires a Twitter academic research Twitter developer account, which equips the user with the necessary API tokens to access Twitter materials and contents that are older than seven days. Applying for an academic developer account is free but requires explaining the purpose and planned use of the Twitter materials. Further, the *search_full_archive* developer environment within Tweepy must be set and labelled in the Twitter developer account portal.

2.3. Data preparation

The Tweepy method above returns a specific 'Twitter search result' type object within Python and includes many details that are not considered important for this research question or are not in the desired format or order. Thus, the output from each search is transformed into a table that retains only the 'full_text' and 'created_at' columns, with all others being dropped. The nine individual tables are then concatenated to create a single table with all of the data, after which the 'Create_at' values are re-coded from date and time (down to the second) to simple year-month values. The combined data is then saved and exported as a .csv in preparation for Natural Language Processing (NLP) steps.

The NLP steps begin with the *clean* method from 'Preprocessor'[26] which removes URLs, hashtags, mentions, and reserved words (RT for retweet and FAV for favourite), emojis and smileys. Then, the *sub* method from 'Re'[27] removes punctuation and any

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 August 2021 doi:10.20944/preprints202108.0516.v1

words that appear in the English stop words list from the 'NLTK.corpus' package[28] (e.g. determiners, prepositions, pronouns, etc.). The tweets are then passed through the *word_tokenize*, *pos_tag*, *get_wordnet_pos* and *lemmatize* methods from the 'WordNet' functions from the 'NLTK' package. This changes words to simple lemma versions according to how they were used in the original text so that 'further' and 'farthest' are both changed to 'far', 'boxes' is changed to 'box', and 'recording' is changed to 'record' if it was originally used as a verb but to 'recording' if it was originally used as a noun.

Finally, the data set is copied and edited into three new data sets. Data set 1 contains only entries from the three most recent time periods (2020.4 to 2021.4) and data set 2 contains only entries from the six most recent time periods (2018.10 to 2021.4). Data set 3 contains all entries covering the full nine time periods (2017.4 to 2021.4). These three data sets are used to explore how the ML model performance is influenced when the training data is larger but also more temporally complicated. Each is converted to a list within python and separated into test and train data sets with an 80-20 split.

2.4. Modelling

The NBC model is straightforward; lemmatised tweets are converted to textblobs via the *TextBlob* method from the 'textblob'[29] package. Following this, three empty NBC models are trained by applying the *NaiveBayesClassifier* method (via 'textblob.classifiers' tool from 'textblob' package), one each on the training portions of Data sets 1, 2, and 3. The trained NBC models are then tested on the corresponding test portions of the same data sets, so that the NBC trained on the train data from Data set 1 is tested on the test data from Data set 1 and so on.

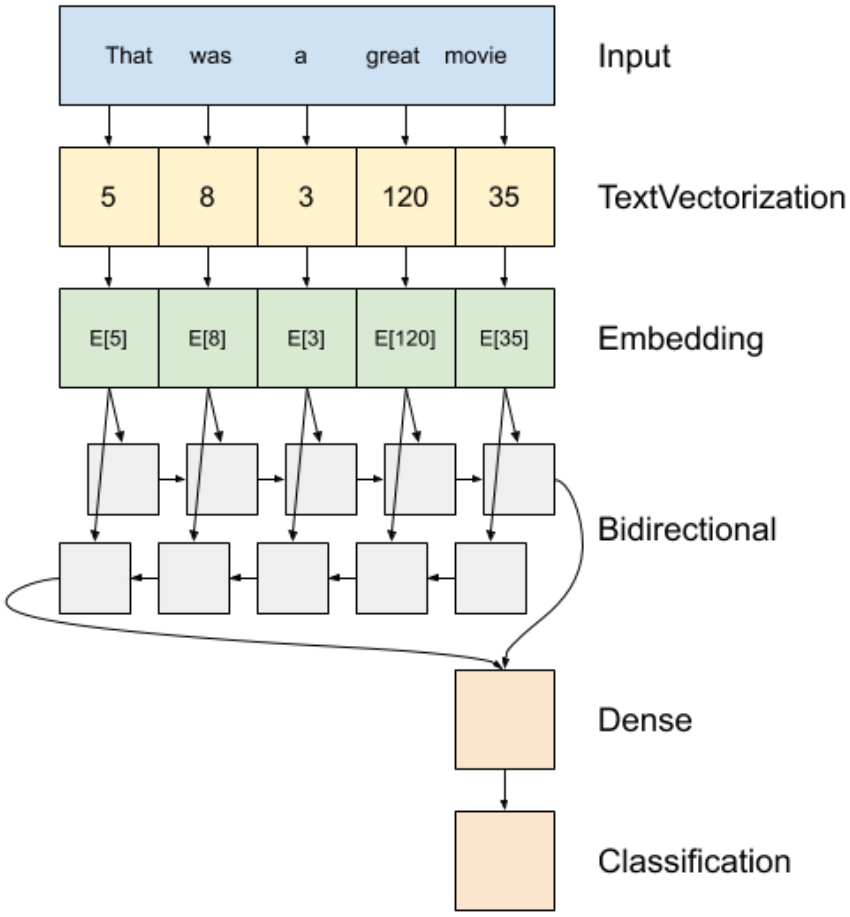

Figure 2. The RNN model has five layers, each of which serves a specific function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 August 2021 doi:10.20944/preprints202108.0516.v1

The RNN is more complicated but mostly uses the 'Keras'[30] and 'Tensorflow'[31] python libraries. First, the lemmatised data sets are encoded as tensorflow-type data set through *tf.data.Dataset.from_tensor_slices* function and the 'created_at' dates are converted to binary features representing each time period through the *get_dummies* method in Pandas library. Following this preparation step, the training and testing data sets are attached to the RNN with the *batch* and *prefetch* functions and is passed through all 5 layers of the RNN 20 times (epochs = 20). The RNN, and its 5 layers, is depicted in figure 2. The first layer uses *experimental.preprocessing.TextVectorization* and *adapt* methods from 'Tensorflow' to convert the input to a word index sequence that includes the frequency of each word within the specific data set. The second layer transforms the word index sequences to trainable vector sequences which, with sufficient training, turns different word sequences with similar meanings into similar vector sequences. The third layer is a *tf.keras.layers.Bidirectional* wrapper, which propagates the input forward and backwards through the RNN with long short-term memory (LSTM) architecture. The fourth and fifth layers convert the trained vector sequences into prediction through *tf.keras.layers.Dense* functions. These prediction vectors have as many elements as there are classes (e.g. Data set 1 with 3 time periods has a three-element layer) with each element in this layer belonging to one and only one of the possible output values. The element with the highest score is interpreted as the predicted probability of output classes. The fourth layer applies the common *ReLu* activation function with 64 nodes, while the fifth and final layer adopts the *softmax* activation function to perform the final multi-class prediction. All five layers are compiled using a *categorical_crossentropy* loss function and an *adam* optimiser function.

3. Results

3.1. Exploring language used

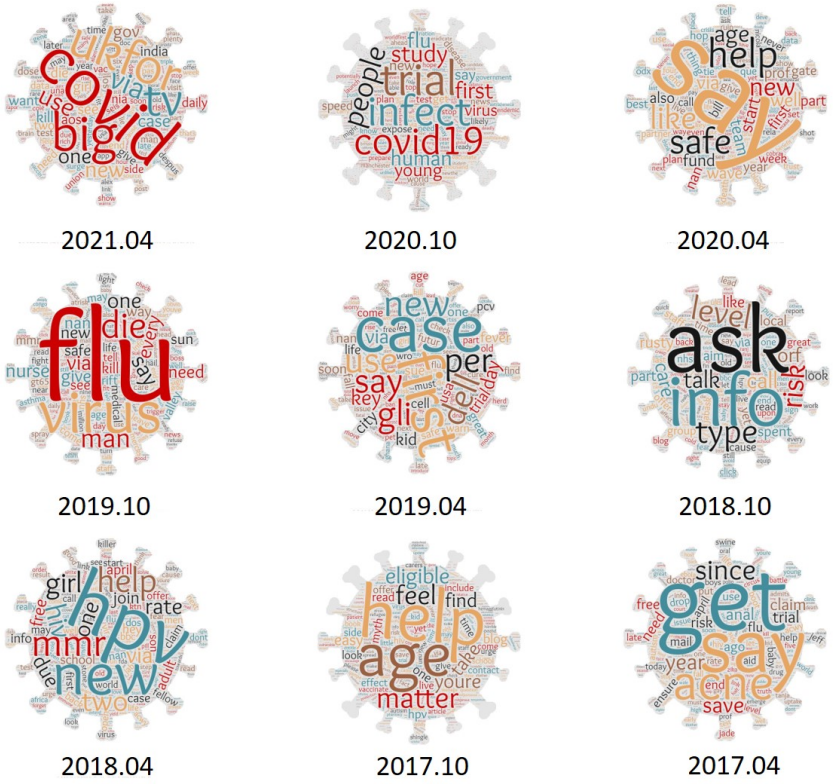

Figure 3. Word clouds for each time period, labelled by the closing date of that time period.

It hepls to begin by exploring the language within each time period. To this end, figure 3 presents word clouds for each time period in the data sets. The most frequent words are

 doi:10.20944/preprints202108.0516.v1

included, with the size of each word representing its frequency. Each of the time periods is clearly unique and no two word clouds are the same although there are many words that appear in the word clouds for more than one time period.

Comparing the word clouds shows differences that may be linked to time in ways that are more or less obvious. For example, the most recent time periods feature words like 'covid' or 'covid 19' quite strongly. The global pandemic (still on going at time of writing) has clearly influenced online discussions about vaccines and vaccine development. Neither of these words feature in older time periods since they are neologisms. It is clear that a global event can create or popularise new words and that the appears and use of those words is related to the timing of the event and the subsequent discussions.

Similarly, 'hpv', 'girl' and 'mmr' feature strongly in the 2018.04 time period but are not as prominent in other time periods. There was no global pandemic in 2018, so it is less clear why this time period should display these distinct word frequencies. A bit of research showed that 2018 saw the HPV vaccine offered to preteen and teenage boys as well as girls and also featured unusually high rates of infection for measles, mumps and rubella. Both of these events generated news articles, online discussions and new behaviours, all of which are linked to the timing of policy changes and epidemiological spreading patterns. While neither of these vaccine-related events in 2018 was so memorable that the authors didn't have to go looking for an explanation, they were nevertheless sufficient to have an temporally limited impact on Twitter discussions.

3.2. Overall model performance

Importantly, both the NBC and RNN models demonstrated good overall performance on categorising the tweets by time period; the least accurate performances of any ML model in any experimental condition was just below 80%. Thus, both of the selected ML models are capable of capturing the characteristics of the tweets and using those characteristics to predict which time period category the tweets came from. It is important to reiterate that this task treats the time periods as simple categories into which the data must be classified. The time period categories in this case are equivalent to any other categories that might be used to train the ML models, such as sentiment, author, style or anything else. This means that the ML classifiers used in these experiments have no explicit approach to time or order. Despite not having any explicit approach to time or order, there were some performance differences that suggest both ML models are sensitive to the implicit 'arrow of time'.

Accuracy for both models, and test loss for the RNN models, are reported in table 1. Accuracy, as described previously, is a simple measure of whether the ML model predicts the correct time period for each tweet in the test data. Loss is the difference between the correct output vector and the predicted output vector. To clarify, the ML models need to categorise tweets into one of three time periods when processing Data set 1. Each tweet will have a correct output vector with two zeros and a single 1, such as *[1 0 0]* which would apply to any tweet that belongs to the first of the three time periods. If the ML model examines the tweet it might predict an output vector of [.8 .2 .0]. That can be interpreted to mean that the model is 80% certain that the tweet belongs to the first time period but a 20% chance that it could belong to the second time period. In this case, the ML model would have predicted the time period accurately, as the 80% certainty applies to the correct time period, but would have a loss score of .2 because of the differences between the predicted and correct output vectors.

NBC models are trained with a single exposure to the training data and then tested once on a different set of test data, from which they do not learn. In contrast, RNN models are trained with multiple epochs or 'passes' through both the training and test data, although they only learn from the training data and not from the test data. This means that the accuracy and loss for RNN models can be reported as the final values on the last pass through the test data. Accuracy and loss for RNN models can also be reported by showing the accuracy and loss values for each pass through the training and test data as well, creating a function that shows how they learn from epoch to epoch. This allows a

 doi:10.20944/preprints202108.0516.v1

	NBC	RNN	
Data Set	Accuracy score	Test Loss	Accuracy score
1	0.90	0.39	0.89
2	0.84	0.17	0.85
3	0.79	0.13	0.81

Table 1: Test loss and accuracy scores for both ML models according to data set

more nuanced understanding of how the RNN model performance and demonstrates how the final values are reached gradually after the learning has plateaued.

3.3. NCB performance

The most accurate NBC model performance was over 90% and came from Data set 1. This was the data subset with only the three most recent time periods and thus the least data with which to train the ML model. Accuracy for the NBC model falls as more data is added, with Data set 2 showing 84% accuracy and Data set 3 falling just below 80% accuracy. Loss was not a metric generated by the NBC model used in this experiment, so accuracy is the only way to judge performance for this ML model.

3.4. RNN performance

The RNN models show a similar pattern; accuracy drops as the volume of data and number of time periods in each data set increases. Data set 1, with only three time periods, was almost 90% accurate while Data set 2 was over 84% accurate and Data set 3 was just over 81% accurate. Figure 4 shows how the accuracy and loss functions changed through learning in multiple epochs, with blue lines showing the accuracy and loss values from the training data and orange lines showing the values for the test data.

4. Discussion

An important result from both models is that accuracy was best for Data set 1 which contained fewer time periods and less data overall. This result is counter-intuitive in light of the commonly accepted idea that ML performance is improved by more data. Instead, this result supports the hypothesis that ML models which are not designed to account for or deal with temporal features or ordered data can still be influenced by an implicit representation of time. Interestingly, the fact that the NBC model showed this result as well suggests that even very simple non-temporal models can be disrupted by subtle or implicit temporal features.

At the same time, the drop in accuracy as data volume increased was not equal between the two models. This suggests that not all ML models are equally disrupted by or equally adept at accounting for implicit representations of time. As the total accuracy drop was less pronounced for the RNN than it was for the NBC, RNN models may be less susceptible to subtle temporal effects or are better at capturing the complexity captured by data that spans greater time frames.

The learning functions for the RNN models show that the accuracy and loss for training and test data after each epoch are close, but not identical. This means that the models are well fitted to the data without suffering from over- or under-fitting. These learning functions also reveal that handling more and more temporally complex data necessitates more iterations; learning for Data set 1 levelled off around the 10 epoch mark but the much larger Data set 3 required at least 20 epochs.

The results presented here could be explored further through other ML models, different or larger data sets (potentially with more specific geographical filtering), classification tasks that do not rely on predicting when a tweet was created, and other training regimes. One particularly interesting training regime to explore would be ‘curriculum learning’ in which training data is fed into the model in a deliberate order (e.g. easy or straightforward

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 August 2021 doi:10.20944/preprints202108.0516.v1

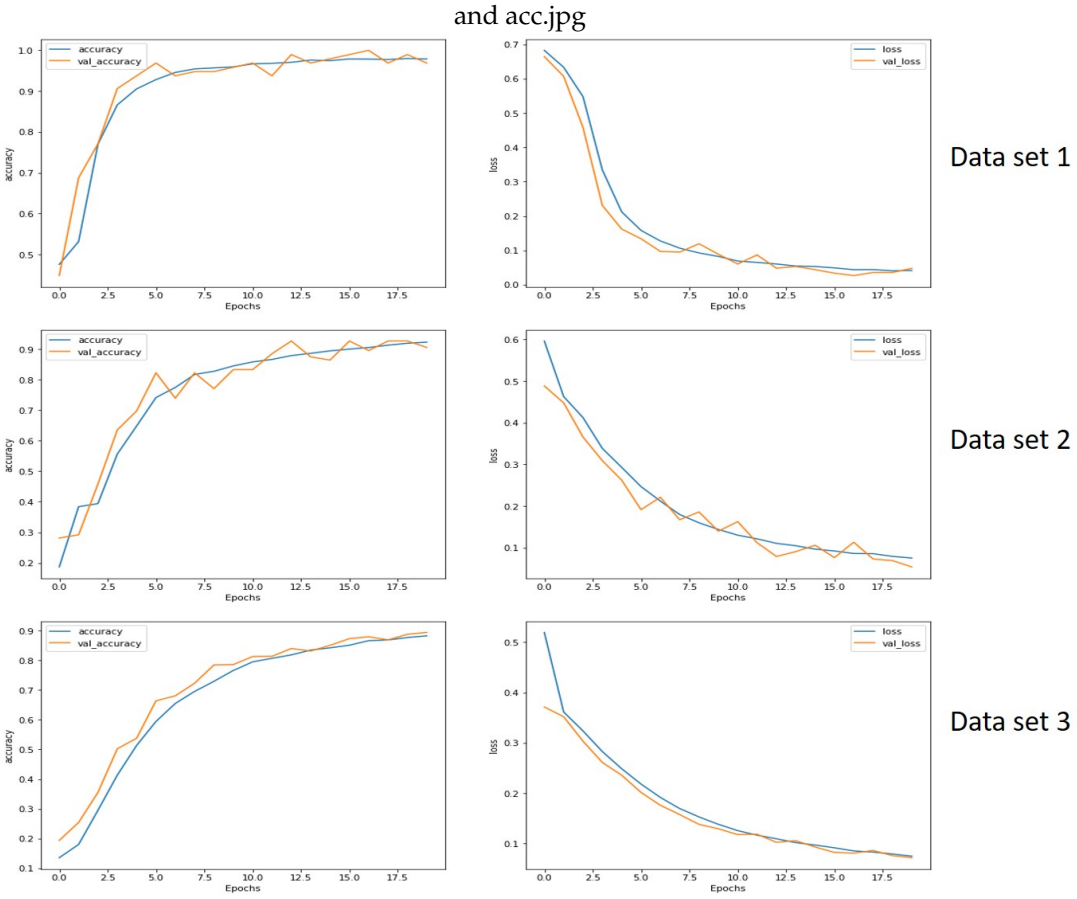

Figure 4. Loss and accuracy functions from the RNN

training data first and then hard or more ambiguous training data later). [32]. In this way, additional interactions between ML and implicit representations of the 'arrow of time' may be discovered which might shed more light on the links between data complexity or temporal features and performance that can be achieved under different training regimes.

5. Conclusions

The main outcome of this research is that both ML models examined here contradict generally accepted ideas about ML; that more data is better and that classification tasks are not affected by time. Both models show that performance as measured by accuracy was not improved by making more data available. Instead, the results suggest that when the data contains implicit temporal features, more data disrupts the capacity to classify. Effectively, implicit temporal data seems to disrupt non-temporal ML models.

Importantly, this research should be understood as a first step or 'proof of concept' rather than an exhaustive exploration of how ML and time interact. Classifying tweets into time periods is not a particularly important task, especially when Twitter data can easily be exported with a time stamp. Nevertheless, there are many important classification tasks that already routinely employ ML models, such as predicting diagnoses from medical images or identifying risk categories for individuals from their social media activity. These, and indeed most, classification tasks do not typically make use of time in the data or the analysis. Despite this, time is present in training data because it is created at a specific point in time (and within the technological and social context of that time). Likewise, the categories into which data is classified are also created, modified, or no longer used within clear temporal contexts. Thus, even without such classification tasks are inevitably influenced by time, even if the temporal representation is subtle or implicit. Importantly,

the results of this research strongly suggest that the performance of ML classification tasks is affected by such subtle or implicit temporal features.

Currently, researchers weigh up many factors when selecting the appropriate ML model to use on a given research topic . These often include data volume, data complexity, generalisability, calculation costs, error tolerance and many more because the characteristics of their data must be balanced against the research project aims. Time is only included in these factors to consider when the data or research question is explicitly temporal or ordered. Thus, the main conclusion of this research is that time should be added to the set of factors that researchers consider when choosing which ML models to use because an unacknowledged mismatch between the temporal features of the data and the ML algorithm can have consequences on the total performance or training needs.

As it stands, the authors can only say that implicit time seems to matter for some classification tasks so researchers should be careful when selecting a ML model in light of the data and project aims. Thus, the authors only suggest that they consider temporal features and the potential disruptions to ML performance from a mismatch between non-temporal methods and implicitly temporal data when they select a ML algorithm. With further research, the authors would like to produce some heuristics to help in this selection process.

Author Contributions: Conceptualisation, J.K. and A.Z.; methodology, J.K. and A.Z.; software, A.Z.; validation, J.K. and A.Z.; formal analysis, A.Z.; writing—original draft preparation, J.K.; writing—review and editing, J.K. and A.Z.; visualisation, J.K. and A.Z.;All authors have read and agreed to the published version of the manuscript."

Funding: Funded by UKRI through the ESRC under grant number ES/P008437/1, with contributions from our partners.

Data Availability Statement: The code and data used in this analysis are available online at .

Acknowledgments: In this section you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brunk, C.A.; Pazzani, M.J. An investigation of noise-tolerant relational concept learning algorithms. In *Machine Learning Proceedings 1991*; Elsevier, 1991; pp. 389–393.
2. Kaiser, T.M.; Burger, P.B. Error tolerance of machine learning algorithms across contemporary biological targets. *Molecules* **2019**, *24*, 2115.
3. Başţanlar, Y.; Özuysal, M. Introduction to machine learning. In *miRNomics: MicroRNA Biology and Computational Analysis*; Springer, 2014; pp. 105–128.
4. Eddington, A. *The nature of the physical world: THE GIFFORD LECTURES 1927*; Vol. 23, BoD–Books on Demand, 2019.
5. Page, S.E.; others. Path dependence. *Quarterly Journal of Political Science* **2006**, *1*, 87–115.
6. Mikhailovsky, G.E.; Levich, A.P. Entropy, information and complexity or which aims the arrow of time? *Entropy* **2015**, *17*, 4863–4890.
7. Ben-Naim, A. Entropy, Shannon's measure of information and Boltzmann's H-theorem. *Entropy* **2017**, *19*, 48.
8. Febres, G.; Jaffe, K. A fundamental scale of descriptions for analyzing information content of communication systems. *Entropy* **2015**, *17*, 1606–1633.
9. CLARK, E. *Common Ground*; Vol. 87, Wiley Online Library, 2015.
10. Eshghi, A.; Howes, C.; Gregoromichelaki, E.; Hough, J.; Purver, M. Feedback in conversation as incremental semantic update. Proceedings of the 11th International Conference on Computational Semantics, 2015, pp. 261–271.
11. Ferreira, F.; Henderson, J.M. Recovery from misanalyses of garden-path sentences. *Journal of Memory and Language* **1991**, *30*, 725–745.

12. Romeo, R.R.; Leonard, J.A.; Robinson, S.T.; West, M.R.; Mackey, A.P.; Rowe, M.L.; Gabrieli, J.D. Beyond the 30-million-word gap: Children's conversational exposure is associated with language-related brain function. *Psychological science* **2018**, *29*, 700–710.
13. Laxman, S.; Sastry, P.S. A survey of temporal data mining. *Sadhana* **2006**, *31*, 173–198.
14. Bagnall, A.; Bostrom, A.; Large, J.; Lines, J. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. Extended version. arXiv 2016. *arXiv preprint arXiv:1602.01711* **2016**.
15. Wang, S.; Cao, J.; Yu, P. Deep learning for spatio-temporal data mining: A survey. *IEEE Transactions on Knowledge and Data Engineering* **2020**.
16. Ahmed, N.K.; Atiya, A.F.; Gayar, N.E.; El-Shishiny, H. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews* **2010**, *29*, 594–621.
17. Graber, M.L. The incidence of diagnostic error in medicine. *BMJ quality & safety* **2013**, *22*, ii21–ii27.
18. Kanner, L.; others. Autistic disturbances of affective contact. *Nervous child* **1943**, *2*, 217–250.
19. Dodwell, H. "Status Lymphaticus," the Growth of a Myth. *British medical journal* **1954**, *1*, 149.
20. Goel, A.; Gautam, J.; Kumar, S. Real time sentiment analysis of tweets using Naive Bayes. 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016, pp. 257–261. doi:10.1109/NGCT.2016.7877424.
21. Prakruthi, V.; Sindhu, D.; Anupama Kumar, D.S. Real Time Sentiment Analysis Of Twitter Posts. 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), 2018, pp. 29–34. doi:10.1109/CSITSS.2018.8768774.
22. Bertrand, K.Z.; Bialik, M.; Virdee, K.; Gros, A.; Bar-Yam, Y. Sentiment in new york city: A high resolution spatial and temporal view. *arXiv preprint arXiv:1308.5010* **2013**.
23. Zhao, L.; Jia, J.; Feng, L. Teenagers' stress detection based on time-sensitive micro-blog comment/response actions. IFIP International Conference on Artificial Intelligence in Theory and Practice. Springer, 2015, pp. 26–36.
24. Zhao, A.; Kasmire, J. ICTeSSH-Arrow-of-Time, 2021.
25. Roesslein, J. Tweepy: Twitter for Python! *URL: https://github.com/tweepy/tweepy* **2020**.
26. Python Package Index - PyPI.
27. Van Rossum, G. *The Python Library Reference, release 3.8.2*; Python Software Foundation, 2020.
28. Bird, S.; Klein, E.; Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit*; " O'Reilly Media, Inc.", 2009.
29. Loria, S. textblob Documentation. *Release 0.15* **2018**, *2*.
30. Chollet, F.; others. Keras, 2015.
31. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.
32. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum Learning. Proceedings of the 26th Annual International Conference on Machine Learning; Association for Computing Machinery: New York, NY, USA, 2009; ICML '09, p. 41–48. doi:10.1145/1553374.1553380.

 doi:10.20944/preprints202108.0516.v1