Sig-Networks Toolkit: Signature Networks for Longitudinal Language Modelling

Talia Tseriotou[1*], **Ryan Sze-Yin Chan**[2*], **Adam Tsakalidis**[1,2], **Iman Munire Bilal**[3], **Elena Kochkina**[1‡], **Terry Lyons**[2,4], **Maria Liakata**[1,2,3]
[1]Queen Mary University of London, [2]The Alan Turing Institute,
[3]University of Warwick, [4]University of Oxford
`t.tseriotou@qmul.ac.uk; rchan@turing.ac.uk`

Abstract

We present an open-source, pip installable toolkit, **Sig-Networks**, the first of its kind for longitudinal language modelling. A central focus is the incorporation of Signature-based Neural Network models, which have recently shown success in temporal tasks. We apply and extend published research providing a full suite of signature-based models. Their components can be used as PyTorch building blocks in future architectures. Sig-Networks enables task-agnostic dataset plug-in, seamless pre-processing for sequential data, parameter flexibility, automated tuning across a range of models. We examine signature networks under three different NLP tasks of varying temporal granularity: counselling conversations, rumour stance switch and mood changes in social media threads, showing SOTA performance in all three, and provide guidance for future tasks. We release the Toolkit as a PyTorch package[1] with an introductory video [2], Git repositories for preprocessing[3] and modelling[4] including sample notebooks on the modeled NLP tasks.

1 Introduction

Existing work on temporal and longitudinal modelling has largely focused on models that are task-oriented, including tracking mood changes in users' linguistic content (Tsakalidis et al., 2022b,a), temporal clinical document classification (Ng et al., 2023), suicidal ideation detection on social media (Cao et al., 2019; Sawhney et al., 2021), real-time rumour detection (Liu et al., 2015; Kochkina et al., 2023). Transformer-based models struggle to outperform more traditional RNNs in such tasks, highlighting their limitations in temporal settings (Mullenbach et al., 2018; Yuan et al., 2022). Inspired by the success of models with short- and long-term processing capabilities (Didolkar et al., 2022; Tseriotou et al., 2023) in producing compressed temporal representations, we develop a toolkit that applies Signature Network models (Tseriotou et al., 2023) to various longitudinal tasks. Path signatures are capable of efficient and compressed encoding of sequential data, sequential pooling in neural models, enhancement of short-term dependencies in linguistic timelines and encoding agnostic to task and time irregularities. We make the following contributions:

- We release an open-source pip installable toolkit for longitudinal NLP tasks, **Sig-Networks**, including examples on several tasks to facilitate usability and reproducibility.
- For data preprocessing for the Signature Networks models (Tseriotou et al., 2023), we introduce another pip installable library `nlpsig` which receives as input streams of textual data and returns streams of embeddings which can be fed into the models we discuss in this paper.
- We showcase SOTA performance on three longitudinal tasks with different levels of temporal granularity, including a new task and dataset – longitudinal rumour stance, based on rumour stance classification (Zubiaga et al., 2016; Kochkina et al., 2018). We highlight best practices for adaptation to new tasks.
- Our toolkit allows for flexible adaptation to new datasets, preprocessing steps, hyperparameter choices, external feature selection and benchmarking across several baselines. We provide the option of flexible building blocks such as Signature Window Network Units (Tseriotou et al., 2023) and their extensions, which can be used as a layer integrated in a new PyTorch model or as a stand-alone model for sequential NLP tasks. We share NLP-based examples via notebooks, where users can easily plug in their own datasets.

---

* Indicates equal contribution.
‡Work done while at Queen Mary University of London.
[1]`https://pypi.org/project/sig-networks/`
[2]`http://youtu.be/lrjkdfYf8Lo`
[3]`https://github.com/datasig-ac-uk/nlpsig/`
[4]`https://github.com/ttseriotou/sig-networks/`

2 Related Work

Longitudinal NLP modelling has been sporadically explored in tasks like semantic change detection (Bamler and Mandt, 2017; Yao et al., 2018; Tsakalidis and Liakata, 2020; Montariol et al., 2021; Rosin and Radinsky, 2022) or dynamic topic modelling (He et al., 2014; Gou et al., 2018; Dieng et al., 2019; Grootendorst, 2022). Such approaches have limited generalisability as they track the evolution of specific topics over long-periods of time. Social media data have given rise to longitudinal tasks such as mental health monitoring (Sawhney et al., 2021; Tsakalidis et al., 2022a), stance detection and rumour verification (Kochkina et al., 2018; Chen et al., 2018; Kumar and Carley, 2019) requiring more fine-grained temporal modelling . Other tasks, like healthcare patient notes (Ng et al., 2023) and dialogue act classification (Liu et al., 2017; He et al., 2021) are also longitudinal in nature.

Path Signature (Chen, 1958; Lyons, 1998) is a collection of iterated integrals studied in the context of solving differential equations driven by irregular signals. It provides a summary of complex un-parameterised data streams through an infinite graded sequence of important statistics. Thus, it produces a collection of statistics efficiently summarising important information about the path. Signatures are deemed invaluable in machine learning (Levin et al., 2013) as sequential feature transformers (Yang et al., 2016; Xie et al., 2017; Yang et al., 2017; Lyons et al., 2014; Perez Arribas et al., 2018; Morrill et al., 2020), or integrated components of neural models (Bonnier et al., 2019; Liao et al., 2021; Tseriotou et al., 2023). However, they have only been sparsely explored within NLP (Wang et al., 2019, 2021; Biyong et al., 2020), addressing only sequentiality or temporality. Motivated by the wide range of longitudinal NLP tasks and the work by Tseriotou et al. (2023) we present a toolkit for neural sequential path signatures models achieving SOTA performance in a range of such tasks.

Libraries for computing path signatures include `roughpy`, `esig`, `iisignature` (Reizenstein and Graham, 2020), `signatory` (Kidger and Lyons, 2021) and `signax` (see links in Appendix E). Sig-Networks is a PyTorch library using `signatory` for differentiable computations of the signature and log-signature transforms on GPU. Currently only `signax` provides an alternative implementation of differentiable computations of signatures using JAX (Bradbury et al., 2018).

3 Methodological foundations

3.1 Task Formulation and Background

Longitudinal Task Formulation. We use the following terminology throughout the paper:

- *Data Point*: d_i, is a single piece of information at a given time, i.e. a post, tweet or utterance.
- *Data Stream*: $S^{[t_1,t_m]}$, is a series of chronologically ordered data points $\{d_1, \ldots, d_m\}$ at times $\{t_1, \ldots, t_m\}$, i.e. a timeline or a conversation.

For each d_i, we consider its historical data stream. We divide our models in two categories: (a) *window-* and (b) *unit-based*. In (a) we assume a window of $|w|$ most recent historical data points of d_i, H_i=$\{d_{i-(w-1)}, \ldots, d_i\}$, as our modeling sequence. In (b), we follow Tseriotou et al. (2023) to construct n history windows, each of length $|w|$, shifted by k points.[5] The modeling sequence is given by $H_i = \{h_{i_1}, ..., h_{i_{n-1}}, h_{i_n}\}$ with the qth unit (of w posts) defined as h_{i_q}= $\{p_{i-(n-q)k-(w-1)}, p_{i-(n-q)k-(w-2)}, ..., p_{i-(n-q)k}\}$.

Path Signatures Preliminaries. In our formulation, the textual data stream is the equivalent of the path P over an interval $[t_1, t_m]$ and the signature $S(P)$ is a pooling layer providing a transformed representation for these sequential data. The signature is a collection of all r iterated integrals along dimensions c: $S(P)_{t_1,t_m} = (1, S(P)^1_{t_1,t_m}, ..., S(P)^c_{t_1,t_m}, S(P)^{1,1}_{t_1,t_m}, S(P)^{1,2}_{t_1,t_m}, ... , S(P)^{c,c}_{t_1,t_m}, ..., S(P)^{i_1,i_2,\cdots,i_r}_{t_1,t_m}, ...)$. Since the iterated integrals can go up to infinite dimensions, a degree of truncation N (i.e. up to N-folded integrals) is commonly used. A higher N leads to a larger feature space. *Log*-signatures' output feature space increases less rapidly with input dimensions c, and depth N, allowing a more compressed representation. Sig-Networks allows for the selection of the desired N and the implementation of signatures or log-signatures. We use N=3 and log-signatures which achieved the best performance.

3.2 System Overview

Fig. 1 shows the overview of our Sig-Networks toolkit. The system receives a task-agnostic dataset of linguistic data streams. These can optionally include a set of pre-computed linguistic embeddings for each data point (e.g. post), timestamps and non-linguistic external features. Linguistic embeddings can also be computed by the system

[5] The total number of modeled data points is $k*n+(w-k)$.

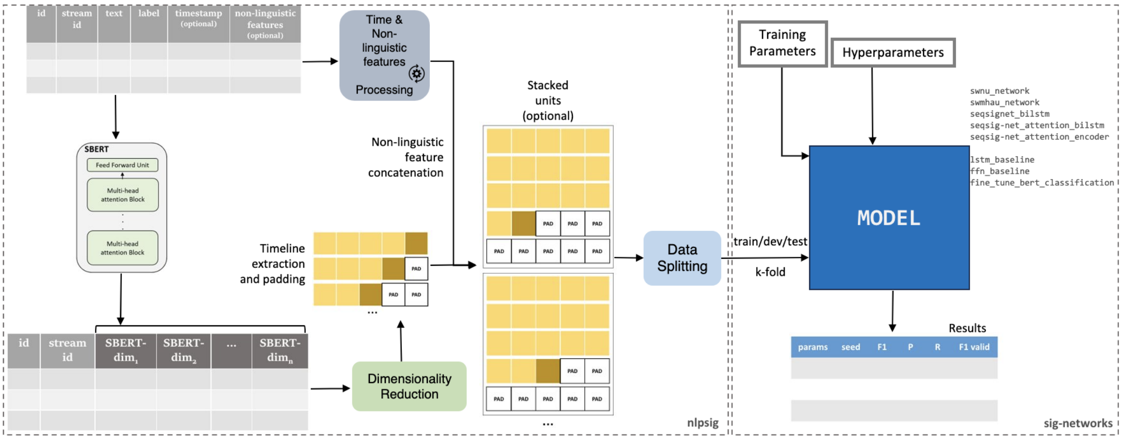

Figure 1: Sig-Networks Tooklit Overview.

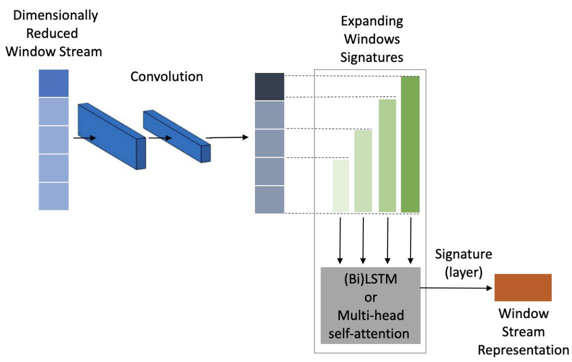

Figure 2: Signature Window Unit and its variations.

and then dimensionally reduced using a selected method (§3.3). Timestamps can be processed to produce and normalise time-related features. The data points are then chronologically ordered and padded based on either a *window*- or a *unit-basis*. Data splitting for model training is performed by the relevant module (§4.1), providing a range of options (including k-fold, stratification). A range of baseline or Signature Network models are available for training (§3.4,4.3,4.2) through user defined parameters, integrating hyperparameter tuning functions for task-based optimal parameter selection.

3.3 Feature Encoding

Each data point is encoded in a high-dimensional space using SentenceBERT (SBERT) (Reimers and Gurevych, 2019) to derive semantically meaningful embeddings. Our toolkit provides different sentence encoding options (§4.1) [6] and multiple options for dimensionality reduction (§4.1). We found UMAP to perform slightly better. Sig-Networks also caters for time-related and external feature incorporation. There is a range of timestamp-derived features and normalisation methods, according to different task characteristics (§4.1 & Appendix C). External information and domain-specific features can be either included as part of the stream feature space, c, or concatenated at the output of the model.

[6]We recommend 384-dim embeddings to facilitate dimensionality reduction required for input to signature transforms.

3.4 Signature Network Models

The Signature Network model family forms an extension of the work by Tseriotou et al. (2023) on combining signatures with neural networks for longitudinal language modeling. We present a range of models (§5.2) based on the foundational Signature Window Network Unit (SWNU), which models the granular linguistic progression in a stream: it reduces a short input stream via a conv-1d layer operation, applies an LSTM on signatures on locally expanding windows of the stream and produces a stream representation via a signature pooling layer.

SWNU implementation is flexible, allowing selection between LSTM *vs* BiLSTM, convolution-1d layer *vs* convolution neural network (CNN), and the option to stack multiple such units to form a deeper network. Importantly, we also introduce a variant of SWNU ('SW-Attn'), replacing LSTM with a Multi-head self-attention with an add & norm operation and a linear layer (Fig. 2).

Furthermore, the toolkit allows for the flexible use of Seq-Sig-Net (the best performing model by Tseriotou et al. (2023)), which sequentially models SWNU units through a BiLSTM, preserving the local sequential information and capturing long-term dependencies. Further available variants of Seq-Sig-Net include SW-Attn+BiLSTM (replacing SWNU with a SW-Attn unit) and SW-

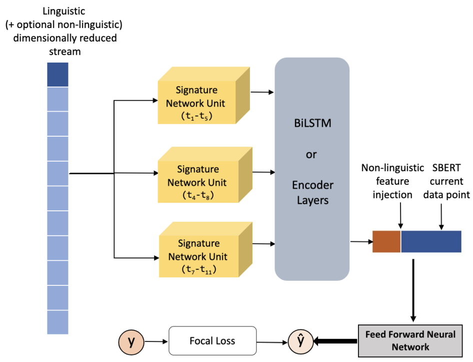

Figure 3: Seq-Sig-Net and its variations using SWNU (yellow, see Fig. 2) on a sample length of 11 points.

Attn+Encoder (replacing BiLSTM with stacked Encoder layers on top of learnable unit embeddings). The final representation is pooled through a trainable [CLS] token. The number of stacked layers is user defined (see Fig. 3). For all Sig-Network models, we follow the same formulation as Tseriotou et al. (2023), by concatenating the SBERT vectors of the current data point with the learnable stream representation and passing it through a feed-forward network for classification using focal loss (Lin et al., 2017). The system provides flexibility with respect to the number of hidden layers and the optional addition of external features. It also provides separate classes for the signature units so they can be incorporated in new architectures.

4 System Components

As shown in Fig. 1, the toolkit is split up into two pip installable Python libraries: `nlpsig` (SBERT vector extraction, data pre-processing including dimensionality reduction of the SBERT streams, constructing model inputs) and `sig-networks` (PyTorch implementations of our models, functions for model training/evaluation).

4.1 Data Preparation Modules in `nlpsig`

These modules perform data loading and preprocessing. `nlpsig` allows for loading pre-computed embeddings for the data points or calculating them using any pretrained or custom model from the `sentence-transformer` and `transformer` libraries via the `nlpsig.encode_text` modules.[7]

Utilising signatures typically requires dimensionality reduction of the data point embeddings (§3.3). `nlpsig` provides several options via the `nlpsig.DimReduce`[8] class: UMAP (McInnes et al., 2018), Gaussian Random Projections (Bingham and Mannila, 2001; Achlioptas, 2003), PPA-PCA (Mu and Viswanath, 2018), PPA-PCA-PPA (Raunak et al., 2019). The `nlpsig.PrepareData`[9] class is used to process the data and obtain streams of dimension-reduced embeddings as input to the Signature Network family of models (see §3.4).

If the dataset includes timestamps, we automatically compute several time-derived variables with different standardisation options. External non-linguistic features can also be included in the dataset and model. The toolkit provides the flexibility of including these features as part of the path stream and/or concatenated in the output with the SBERT representation of the current data point (see Appendix C). There are wrapper functions in the `sig-networks` package (`sig_networks.obtain_SWNU_input`, `sig_networks.obtain_SeqSigNet_input`) to easily obtain the padded input for each model. Since the `nlpsig` library allows for more flexibility in constructing streams of embeddings, customisation of these wrapper functions is encouraged for different datasets or tasks.

4.2 Training

Through `nlpsig.classification_utils`, the toolkit allows for k-fold cross validation or a single train/test split.[10] Splits can be completely random, stratified (for streams via `split_ids`), or predefined (via `split_indices`). If a subset of the dataset is leveraged for classification (e.g. single-speaker classification in dialogue), the user can define such indices in `path_indices`. For training, the user can select the loss function (cross-entropy, focal loss), a validation metric and specify the early stopping patience. Off the shelf hyperparameter tuning functions are available via grid search.

4.3 Model Modules

These modules allow for flexible training for each model, with the function names presented in Fig. 1. PyTorch classes for the building blocks of our models are provided separately to encourage their novel integration in other systems (e.g. see Appendix G). The toolkit can be used to benchmark datasets using: BERT, feedforward network with(out) historical stream information and BiLSTM. For Sig-

[7] encode_text

[8] dimensionality_reduction

[9] data_preparation

[10] classification_utils

Network family models, we provide options for choosing: 1. N, truncation degree, 2. signatures or log-signatures, 3. pooling options in the units, 4. LSTM or BiLSTM in SWNU, 5. dimensionality reduction of Conv-1d or CNN and their dimensions in the unit, 6. combination method of historical signature modelled stream with current SBERT data point and external features, 7. number of encoder layers, 8. path chronological reversion. Importantly, the user can assess their task of interest and define the window size w, number of units n, and shift k (see § 6). After model tuning one can access the trained model object, a set of results for all seeds and hyperparameters, and a set of results for the best hyperparameters.

5 Experiments

5.1 Tasks and Datasets

We demonstrate the applicability of Sig-Networks across three longitudinal sequential classification tasks of different temporal granularity. For all tasks we consider the current data point, its timestamp and its historical stream.

Moments of Change (MoC). Given sequences of users' posts, MoC identification involves the assessment of a user's self-disclosed mood conveyed in each post with respect to the user's recent history as one of 3 classes: *Switches* (IS): sudden mood shift from positive to negative, or vice versa; *Escalations* (IE): gradual mood progression from neutral/positive to more positive, or from neutral/negative to more negative; or *None* (O): no change in mood (Tsakalidis et al., 2022b). Dataset is ***TalkLife MoC***: 18,702 posts (500 user timelines; 1-124 posts each). Annotation was performed on the post-level with access to the entire timeline.

Counselling Dialogue Classification. Given the data stream of utterances during a counselling dialogue between therapist and client, the task is to categorise client's utterances into one of 3 classes: *Change*: client seems convinced towards positive behaviour change; *Sustain*: client shows resistance to change; *Neutral*: client shows neither leaning nor resistance towards change. We utilise therapist and client utterances in the stream, while classifying only client utterances. Dataset is ***Anno-MI*** (Wu et al., 2022): 133 motivational interviews (MI), 9,699 utterances (4,817 client utterances), sourced from effective and ineffective MI videos on YouTube & Vimeo. The videos were professionally transcribed and annotated by MI practitioners.

Stance Switch Detection. The Stance Switch Detection task tracks the ratio of support/opposition towards the topic of a conversation at each point in time and captures switches in overall stance. This is a binary classification of each post in a conversation stream into: *Switch*: switch between the total number of oppositions (querying or denying) and supports or vice versa; or *No Switch*: either the absence of a switch or cases where the numbers of supporting and opposing posts are equal. For this task we introduce a new dataset, ***Longitudinal Rumour Stance (LRS)***, a longitudinal version of the RumourEval-2017 dataset (Gorrell et al., 2019). It consists of Twitter conversations around newsworthy events. The source tweet of the conversation conveys a rumourous claim, discussed by tweets in the stream. In 325 conversations 5,568 posts are labelled based on their stance towards the claim in the corresponding source tweet as either *Supporting*, *Denying*, *Questioning* or *Commenting*. We convert conversation structure and labels into a Longitudinal Stance Switch Detection task. Conversations are converted from tree-structured into linear timelines to obtain chronologically ordered lists. Then we convert the original stance labels into *Switch* and *No Switch* categories based on the numbers of supporting tweets versus denying and questioning ones at each point in time.

5.2 Models and Baselines

We perform 5-fold cross-validation, repeatedly with 3 seeds (see Appendix A for full details) and compare against the following baselines:

BERT(focal/ce): data point-level (stream-agnostic) BERT (Devlin et al., 2018) fine-tuned using the alpha-weighted focal loss (Lin et al., 2017) or cross-entropy, respectively.

FFN: data point-level Feedforward Network (FFN) operating on SBERT of the current point.

FFN History: stream-level FFN operating on the concatenated SBERT vectors of the current point and the average of its historical stream.

BiLSTM with a single layer operating on a specified number of historical data points.

Our Sig-Networks Family Models are:

SWNU (Tseriotou et al., 2023) uses expanding signature windows fed into an LSTM. We modify the unit to use a BiLSTM and improved padding.

SW-Attn: Same as SWNU but with Multi-head attention instead of an LSTM.

Model	Anno-MI (3-class)			LRS (2-class)			TalkLife (3-class)		
BERT (focal)	.519			.589			.531		
BERT (ce)	.501			.596			.521		
FFN	.512			.581			.534		
FFN History	.520			.625			.537		
BiLSTM ($w = 5$)	.517			.637			.544		
SWNU ($w = 5$)	.522			.670			**.563**		
SW-Attn ($w = 5$)	.515			.667			.556		
History Length **#units (w=5, k=3)**	**11** **3**	**20** **6**	**35** **11**	**11** **3**	**20** **6**	**35** **11**	**11** **3**	**20** **6**	**35** **11**
BiLSTM	.518	.507	.510	.657	.648	.648	.539	.533	.525
SWNU	.522	.512	.493	.671	.654	$\underline{.673}$	.550	.537	.539
SW-Attn	.517	.508	.508	.659	.665	.661	.547	.541	.539
Seq-Sig-Net	**.525**	$\underline{.523}$	.517	.672	**.678**	.654	**.563**	$\underline{.561}$	.559
SW-Attn+BiLSTM	.511	.514	.515	.663	.657	.660	.554	.557	.550
SW-Attn+Encoder	.498	.506	.505	.664	.657	.662	.552	.552	.545

Table 1: Results (macro-avg F1) of the Sig-Networks toolkit models on our three tasks for different History Lengths. **Best** and second best scores are highlighted.

Seq-Sig-Net: Sequential Network of SWNU units using a BiLSTM as in Tseriotou et al. (2023).

SW-Attn+BiLSTM: Seq-Sig-Net with SW-Attn unit instead of SWNU.

SW-Attn+Encoder SW-Attn+BiLSTM with two Encoder layers using unit-level learnable embeddings instead of the BiLSTM .

6 Results and Discussion

Performance comparison. Signature Network models show top performance, with Seq-Sig-Net achieving SOTA or on-par performance with SWNU across all tasks (see Table 1, detailed in Appendix B). On LRS the best model is Seq-Sig-Net with window length w=20 posts (F1=.678), while on Anno-MI the best model is also Seq-Sig-Net but for w=11 (F1=.525). In TalkLife, Seq-Sig-Net and SWNU both reach top performance (F1=.563). The difference of optimal window length across tasks relates to the characteristics of each dataset (see Table 2 and next paragraph). Additionally, the performance of BiLSTM peaks within the same range of history length, different for each task, denoting the best performing models depend on the temporal granularity of the task. Lastly, SW-Attn+Encoder performs better than SW-Attn+BiLSTM regardless of task and history length, further highlighting the importance of sequential modelling for these tasks.

Dataset	Anno-MI		Longitudinal Rumour Stance	TalkLife MoC	
	Change	**Sustain**	**Switch**	**Switch**	**Escalation**
Mean Point Time Diff.	5sec		1hr 26min 40sec	6hr 51min 11sec	
Median Point Time Diff.	3sec		1min 39sec	59min 38sec	
Mean consecutive events	2.21	1.68	8.52	1.58	4.12
Median consecutive events	1	1	4	1	3
Mean no. of events in stream	8.86	4.05	6.45	1.77	4.03
Median no. of events in stream	5	3	0	1	1

Table 2: Dataset Statistics on time and event length.

Seq-Sig-Net outperforms all models across tasks in modeling long-term effects, making it particularly appealing for highly longitudinal tasks; SWNU has the best performance when modeling short linguistic streams. In LRS and TalkLife Sig-Networks outperforms all baselines, for each history length. For Anno-MI, the least longitudinal task due to the short mean/median consecutive sequences of Change/Sustain utterances (see Table 2), we conjecture that most of the performance gain in including historical dialogue information is due to adding more context rather than sequential modelling. This is apparent from the small performance gains of Seq-Sig-Net models compared to FFN History and BERT (focal) versus the much starker performance improvement in the other tasks.

Time-scale analysis. The degree of temporal granularity across datasets ranges from seconds in Anno-MI, minutes in LRS and hours in TalkLife (Table 2), showing the generalisability of Signature-Networks. TalkLife has an average of 1.58/4.12 consecutive Switches/Escalations and a similar average of such events (1.77/4.03 respectively) in each data stream, meaning that the task benefits from good granularity on short modeling windows. This can be provided by both SWNU (window of 5 posts) and a short Seq-Sig-Net of 3 units. Anno-MI presents even shorter sequences of consecutive Change/Sustain intentions (2.21/1.68), but the average number of such events in each conversation is higher (8.86/4.05), therefore benefiting from being less sequential in terms of short-term dependencies but being more sequence dependent on series of windows. Finally, LRS is the most longitudinal task in our experiments showing the highest mean number of consecutive switches (8.52), therefore benefiting from more units in Seq-Sig-Net.

7 Conclusion

We present the Sig-Networks toolkit, which allows for flexible modeling of longitudinal NLP classification tasks using Signature-based Network models (Tseriotou et al., 2023), proposing improvements and variants. We test our system on three NLP classification tasks of different domains and temporal granularity and show SOTA performance against competitive baselines, while also shedding light into temporal characteristics which affect optimal model selection. The Toolkit is made available as a PyTorch package with examples, making it easy to plug-in new datasets for future model extensions.

Limitations

While the Sig-Networks library provides sequential models with very competitive performance on longitudinal NLP tasks, it comes with limitations. Firstly, it requires basic knowledge of Python, since it is available as a PyTorch library, and assumes integration in PyTorch systems. Additionally, its use on classification tasks requires labeled data, which can be expensive to obtain for tasks that require expert annotation. Although our tasks under examination are in English, we believe that this work is extensible to other languages. Since one of the initial steps for obtaining linguistic representations involves the use of a pretrained language model, we expect lower quality for low-resource languages where such pretrained models have poor performance or are non-existent.

Hyperparameter tuning including time feature selection, given that the timestamps are available, is often key in achieving competitive classification performance. We provide guidelines and expect the users to perform a thorough grid search if needed to reach a competitive performance. Lastly, we understand that our data point-level evaluation, which assesses predictions at each point in the stream in silo, can be lacking pattern identification on a stream level. We plan to address stream-level evaluation using the settings from Tsakalidis et al. (2022b) in future work and we encourage users to cross-check performance with stream-level metrics.

Ethics Statement

The current project focusses on providing a toolkit for facilitating research and applications in longitudinal modelling. This is showcased in three tasks, two of which employ existing datasets (TalkLife and AnnoMI) and one is a re-interpretation of an existing public dataset (LRS).

Since the TalkLife dataset involves sensitive user generated social media content, Ethics approval was received from the Institutional Review Board (IRB) of the corresponding ethics board of the University of Warwick prior to engaging in longitudinal modelling with this dataset. Thorough data analysis, data sharing policies to protect sensitive information and data anonymisation were used to address ethical considerations around the nature of such data (Mao et al., 2011; Keküllüoglu et al., 2020). Access to TalkLife's data was obtained through the submission of a project proposal and the approval of the corresponding license by TalkLife[11]. TalkLife data were maintained and experiments were ran through a secure server accessible only by our group members. While we release code examples and results, we do not release any data, labels, models or preprocessing associated with TalkLife data in our git repository.

The AnnoMI dataset is publicly available and is based on transcribed videos of therapy sessions which are enacted.

The LRS dataset is a re-interpretation of the RumourEval 2017 dataset to reflect switches in stance over time. RumourEval-2017 is a well established dataset for stance and rumour verification. The longitudinal stance extension of the dataset allows studying the changes in public stance over time.

Developing methods for longitudinal modeling is an important research direction for better interpretation of events. Potential risks from the application of our work in being able to identify moments of change in individuals' timelines are akin to those in earlier work on personal event identification from social media and the detection of suicidal ideation. Potential mitigation strategies include restricting access to the code base trained on TalkLife and annotation labels used for evaluation.

References

Dimitris Achlioptas. 2003. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *International conference on Machine learning*, pages 380–389. PMLR.

Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250.

John Pougué Biyong, Bo Wang, Terry Lyons, and Alejo J Nevado-Holgado. 2020. Information extraction from swedish medical prescriptions with sig-transformer encoder. *arXiv preprint arXiv:2010.04897*.

Patric Bonnier, Patrick Kidger, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. 2019. Deep signature transforms. *arXiv preprint arXiv:1905.08494*.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake

[11] https://www.talklife.com/

VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Lei Cao, Huijun Zhang, Ling Feng, Zihan Wei, Xin Wang, Ningyun Li, and Xiaohao He. 2019. Latent suicide risk detection on microblog via suicide-oriented word embeddings and layered attention. *arXiv preprint arXiv:1910.12038*.

Kuo-Tsai Chen. 1958. Integration of paths–a faithful representation of paths by noncommutative formal power series. *Transactions of the American Mathematical Society*, 89(2):395–407.

Tong Chen, Xue Li, Hongzhi Yin, and Jun Zhang. 2018. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. In *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2018 Workshops, BDASC, BDM, ML4Cyber, PAISI, DaMEMO, Melbourne, VIC, Australia, June 3, 2018, Revised Selected Papers 22*, pages 40–52. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Aniket Didolkar, Kshitij Gupta, Anirudh Goyal, Nitesh Bharadwaj Gundavarapu, Alex M Lamb, Nan Rosemary Ke, and Yoshua Bengio. 2022. Temporal latent bottleneck: Synthesis of fast and slow processing mechanisms in sequence learning. *Advances in Neural Information Processing Systems*, 35:10505–10520.

Adji B Dieng, Francisco JR Ruiz, and David M Blei. 2019. The dynamic embedded topic model. *arXiv preprint arXiv:1907.05545*.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. Semeval-2019 task 7: Rumoureval 2019: Determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation: NAACL HLT 2019*, pages 845–854. Association for Computational Linguistics.

Zhinan Gou, Lixin Han, Ling Sun, Jun Zhu, and Hong Yan. 2018. Constructing dynamic topic models based on variational autoencoder and factor graph. *IEEE Access*, 6:53102–53111.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.

Yulan He, Chenghua Lin, Wei Gao, and Kam-Fai Wong. 2014. Dynamic joint sentiment-topic model. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):1–21.

Zihao He, Leili Tavabi, Kristina Lerman, and Mohammad Soleymani. 2021. Speaker turn modeling for dialogue act classification. *arXiv preprint arXiv:2109.05056*.

Dilara Keküllüoglu, Walid Magdy, and Kami Vaniea. 2020. Analysing privacy leakage of life events on twitter. In *12th ACM conference on web science*, pages 287–294.

Patrick Kidger and Terry Lyons. 2021. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. In *International Conference on Learning Representations*. https://github.com/patrick-kidger/signatory.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Elena Kochkina, Tamanna Hossain, Robert L. Logan, Miguel Arana-Catania, Rob Procter, Arkaitz Zubiaga, Sameer Singh, Yulan He, and Maria Liakata. 2023. Evaluating the generalisability of neural rumour verification models. *Information Processing & Management*, 60(1):103116.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. *arXiv preprint arXiv:1806.03713*.

Sumeet Kumar and Kathleen M Carley. 2019. Tree lstms with convolution units to predict stance and rumor veracity in social media conversations. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 5047–5058.

Daniel Levin, Terry Lyons, and Hao Ni. 2013. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint arXiv:1309.0260*.

Shujian Liao, Terry Lyons, Weixin Yang, Kevin Schlegel, and Hao Ni. 2021. Logsig-rnn: A novel network for robust and efficient skeleton-based action recognition. *arXiv preprint arXiv:2110.13008*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1867–1870.

Yang Liu, Kun Han, Zhao Tan, and Yun Lei. 2017. Using context information for dialog act classification in dnn framework. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2170–2178.

Terry Lyons, Hao Ni, and Harald Oberhauser. 2014. A feature set for streams and an application to high-frequency financial tick data. In *Proceedings of the 2014 International Conference on Big Data Science and Computing*, pages 1–8.

Terry J Lyons. 1998. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 14(2):215–310.

Huina Mao, Xin Shuai, and Apu Kapadia. 2011. Loose tweets: an analysis of privacy leaks on twitter. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 1–12.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Syrielle Montariol, Matej Martinc, and Lidia Pivovarova. 2021. Scalable and interpretable semantic change detection. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4642–4652.

James H Morrill, Andrey Kormilitzin, Alejo J Nevado-Holgado, Sumanth Swaminathan, Samuel D Howison, and Terry J Lyons. 2020. Utilization of the signature method to identify the early onset of sepsis from multivariate physiological time series in critical care monitoring. *Critical Care Medicine*, 48(10):e976–e981.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. *arXiv preprint arXiv:1802.05695*.

Clarence Boon Liang Ng, Diogo Santos, and Marek Rei. 2023. Modelling temporal document sequences for clinical icd coding. *arXiv preprint arXiv:2302.12666*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Imanol Perez Arribas, Guy M Goodwin, John R Geddes, Terry Lyons, and Kate EA Saunders. 2018. A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder. *Translational psychiatry*, 8(1):274.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jeremy Reizenstein and Benjamin Graham. 2020. Algorithm 1004: The iisignature library: Efficient calculation of iterated-integral signatures and log signatures. *ACM Transactions on Mathematical Software (TOMS)*.

Guy D Rosin and Kira Radinsky. 2022. Temporal attention for language models. *arXiv preprint arXiv:2202.02093*.

Ramit Sawhney, Harshit Joshi, Rajiv Shah, and Lucie Flek. 2021. Suicide ideation detection via social and temporal user representations using hyperbolic learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2176–2190.

Adam Tsakalidis, Jenny Chim, Iman Munire Bilal, Ayah Zirikly, Dana Atzil-Slonim, Federico Nanni, Philip Resnik, Manas Gaur, Kaushik Roy, Becky Inkster, et al. 2022a. Overview of the clpsych 2022 shared task: Capturing moments of change in longitudinal user posts.

Adam Tsakalidis and Maria Liakata. 2020. Sequential modelling of the evolution of word representations for semantic change detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8485–8497.

Adam Tsakalidis, Federico Nanni, Anthony Hills, Jenny Chim, Jiayu Song, and Maria Liakata. 2022b. Identifying moments of change from longitudinal user text. *arXiv preprint arXiv:2205.05593*.

Talia Tseriotou, Adam Tsakalidis, Peter Foster, Terence Lyons, and Maria Liakata. 2023. Sequential path signature networks for personalised longitudinal language modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5016–5031.

Bo Wang, Maria Liakata, Hao Ni, Terry Lyons, Alejo J Nevado-Holgado, and Kate Saunders. 2019. A path signature approach for speech emotion recognition. In *Interspeech 2019*, pages 1661–1665. ISCA.

Bo Wang, Yue Wu, Nemanja Vaci, Maria Liakata, Terry Lyons, and Kate EA Saunders. 2021. Modelling paralinguistic properties in conversational speech to detect bipolar disorder and borderline personality disorder. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7243–7247. IEEE.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Zixiu Wu, Simone Balloccu, Vivek Kumar, Rim Helaoui, Ehud Reiter, Diego Reforgiato Recupero, and Daniele Riboni. 2022. Anno-MI: A dataset of expert-annotated counselling dialogues. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6177–6181. IEEE.

Zecheng Xie, Zenghui Sun, Lianwen Jin, Hao Ni, and Terry Lyons. 2017. Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(8):1903–1917.

Weixin Yang, Lianwen Jin, Hao Ni, and Terry Lyons. 2016. Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 4083–4088. IEEE.

Weixin Yang, Terry Lyons, Hao Ni, Cordelia Schmid, and Lianwen Jin. 2017. Developing the path signature methodology and its application to landmark-based human action recognition. *arXiv preprint arXiv:1707.03993*.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the eleventh acm international conference on web search and data mining*, pages 673–681.

Zheng Yuan, Chuanqi Tan, and Songfang Huang. 2022. Code synonyms do matter: multiple synonyms matching network for automatic icd coding. *arXiv preprint arXiv:2203.01515*.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

A Experiment setup details

We train all models using PyTorch (Paszke et al., 2019) and Huggingface Transformers (Wolf et al., 2020) for BERT, using the alpha-weighted focal loss (Lin et al., 2017), except for BERT (ce).

SBERT representations: As noted in §3.3, we use SentenceBERT (SBERT) (Reimers and Gurevych, 2019) to encode each data point to obtain semantically meaningful embeddings. To do this with our toolkit, we used the `nlpsig.SentenceEncoder`[12] class which uses the `sentence-transformers` library. For each dataset, we obtained 384-dimensional embeddings using the "all-MiniLM-L12-v2" model[13].

Model experiment settings: In each of our experiments in §5, we select the best model for each of the 5 folds using the best validation F1 macro-average score on 100 epochs with early stopping (patience set to 3). For training, we use the Adam optimiser (Kingma and Ba, 2014) with weight decay of 0.0001. For all models, we use the alpha-weighted focal loss (Lin et al., 2017) with setting $\gamma = 2$ and alpha of $\sqrt{1/p_t}$ where p_t is the probability of class t in the training data. The exception is for the BERT (ce) baseline model where we used the cross-entropy loss. For BERT, we used batch size of 8 during training due to limited GPU resources available for training on the secure data environment which hosted the TalkLife dataset. For the other models, we used batch size of 64.

For the TalkLife MoC dataset, we use the same train/test splits as in Tsakalidis et al. (2022a,b); Tseriotou et al. (2023). Furthermore, we average the F1 macro-average performance over three random seeds, $(1, 12, 123)$. For Anno-MI and Longitudinal Rumour Stance datasets, we created the five folds using the `nlpsig.Folds` class[14] class (with `random_state=0`). Each fold constructed was used as a test and the rest as the training and validation data. Validation sets were formed on 33% of the train set. When creating the folds, we stratify using the `transcript_id` for Anno-MI and the conversation ID for Rumours to ensure there was no contamination between streams.

For each model, we perform a grid search for hyperparameter selection based on the validation set performance comparing F1 macro-average. For signature window models, prior to hyperparameter search, we performed dimensionality reduction on the SBERT embeddings using UMAP (McInnes et al., 2018) with the `umap-learn` Python library. Using the `UMAP`[15] class in the library, we kept all default parameters besides `n_neighbors=50`, `min_dist=0.99` and `metric="cosine"`. In each

[12] https://nlpsig.readthedocs.io/en/latest/encode_text.html

[13] https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2

[14] https://nlpsig.readthedocs.io/en/latest/classification_utils.html

[15] https://umap-learn.readthedocs.io/en/latest/api.html

of the signature window models, we reduced the SBERT embeddings to 15 dimensions. For all models considered, the dropout rate was set to 0.1.

In the rest of this section, we state the hyperparameters choices we had for each model. Note that the full results for each model that we trained (for each hyperparameter configuration and seed) as well as the best hyperparameters for each model and dataset can be found in the GitHub repository for the project in the `examples` folder[16].

SWNU and Seq-Sig-Net: For the signature window networks which used the Signature Window Network Unit (SWNU) (§3.4, 5.2), hyperparameter selection was set through a grid search over the parameters: learning rate $\in [0.0005, 0.0003, 0.0001]$, LSTM hidden dimensions of SWNU $\in [10, 12]$, FFN hidden dimensions $\in [[32, 32], [128, 128], [512, 512]]$ where $[h_1, h_2]$ means a two hidden layer FFN of dimensions h_i in the ith layer. For Seq-Sig-Net, the BiLSTM hidden dimensions $\in [300, 400]$. We took the log-signature transform with depth (degree of truncation) 3. In each model run, the convolution-1d reduced dimensions is equal to the LSTM hidden dimensions (i.e. 10 or 12 here).

SW-Attn and Seq-Sig-Net-Attention models: For the signature window networks which used the Signature Window Attention Unit (§3.4, 5.2) hyperparameter selection was set through a grid search over the following parameters: learning rate $\in [0.0005, 0.0003, 0.0001]$, convolution-1d reduced dimensions $\in [10, 12]$, FFN hidden dimensions $\in [[32, 32], [128, 128], [512, 512]]$. We took the log-signature transform with depth (degree of truncation) 3. While the toolkit allows you to easily stack multiple SW-Attn blocks, i.e. multiple iterations of taking the expanding window signatures and multi-head attention (with add+norm and a linear layer), we only have one block, `num_layer=1`.

For models using SW-Attn units, we must choose the number of attention heads to divide the resulting number of signature channels after taking streaming signatures. For models with conv-1d reduced dimensions set to 10, `output_channels=10`, we set `num_heads=5` since after taking a log-signature of depth 3, the output has dimension 385[17]. For models with `output_channels=12`, we set `num_heads=10` since the number of log-signature channels at depth 3 for a path with 12 channels is 650.

BERT: We fine-tuned the `bert-base-uncased`[18] model on the Huggingface model hub, and used the `transformers` library and Trainer API for training the model. The only hyperparameter we performed a grid-search for was learning rate $\in [0.00005, 0.00001, 0.000001]$[19]. For BERT, we found it was important to use a much lower learning rate than the ones we used for other models due to the larger number of parameters in the model.

FFN models: For models using a Feedforward Network (FFN), either operating on the SBERT embedding of the current point (**FFN**) or operating on a concatenation of the current SBERT embedding with the mean average of its historical stream (**FFN History**), we perform a hyperparameter search over learning rate $\in [0.001, 0.0005, 0.0001]$ and hidden dimensions $\in [[64, 64], [128, 128], [256, 256], [512, 512]]$.

BiLSTM: We apply a single layer BiLSTM on a specified number of historical SBERT embeddings for the data point. We perform a grid search over learning rate $\in [0.001, 0.0005, 0.0001]$ and hidden dimension sizes $[200, 300, 400]$.

B Results

We present class-level performance for each task in Tables 3, 4 and 5.

<table>
<tr><th>Model</th><th colspan="3">Neutral(N)</th><th colspan="3">Change(C)</th><th colspan="3">Sustain(S)</th><th colspan="3">Macro-avg</th></tr>
<tr><td>BERT (focal)</td><td colspan="3">.767</td><td colspan="3">.449</td><td colspan="3">.339</td><td colspan="3">.519</td></tr>
<tr><td>BERT (ce)</td><td colspan="3">.784</td><td colspan="3">.442</td><td colspan="3">.277</td><td colspan="3">.501</td></tr>
<tr><td>FFN</td><td colspan="3">.764</td><td colspan="3">.424</td><td colspan="3">.347</td><td colspan="3">.512</td></tr>
<tr><td>FFN History</td><td colspan="3">.761</td><td colspan="3">.449</td><td colspan="3">.351</td><td colspan="3">.520</td></tr>
<tr><td>BiLSTM (w=5)</td><td colspan="3">.753</td><td colspan="3">.449</td><td colspan="3">.348</td><td colspan="3">.517</td></tr>
<tr><td>SWNU (w=5)</td><td colspan="3">.762</td><td colspan="3">.447</td><td colspan="3">.356</td><td colspan="3">.522</td></tr>
<tr><td>SW-Attn (w=5)</td><td colspan="3">.749</td><td colspan="3"><u>.450</u></td><td colspan="3">.346</td><td colspan="3">.515</td></tr>
<tr><th rowspan="2">History Length (units)</th><th colspan="4">11 (n=3)</th><th colspan="4">20 (n=6)</th><th colspan="4">35 (n=11)</th></tr>
<tr><th>N</th><th>C</th><th>S</th><th>Macro-avg</th><th>N</th><th>C</th><th>S</th><th>Macro-avg</th><th>N</th><th>C</th><th>S</th><th>Macro-avg</th></tr>
<tr><td>BiLSTM</td><td>.746</td><td>.446</td><td>.363</td><td>.518</td><td>.754</td><td>.446</td><td>.322</td><td>.507</td><td>.755</td><td>.446</td><td>.329</td><td>.510</td></tr>
<tr><td>SWNU</td><td>.761</td><td>.444</td><td><u>.360</u></td><td>.522</td><td>.759</td><td>.440</td><td>.338</td><td>.512</td><td>.752</td><td>.413</td><td>.314</td><td>.493</td></tr>
<tr><td>SW-Attn</td><td>.759</td><td><u>.450</u></td><td>.341</td><td>.517</td><td>.754</td><td>.438</td><td>.333</td><td>.508</td><td>.749</td><td>.446</td><td>.330</td><td>.508</td></tr>
<tr><td>Seq-Sig-Net</td><td><u>.769</u></td><td>.446</td><td>.359</td><td>.525</td><td><u>.769</u></td><td>.452</td><td>.347</td><td><u>.523</u></td><td>.763</td><td>.446</td><td>.342</td><td>.517</td></tr>
<tr><td>SW-Attn+BiLSTM</td><td>.750</td><td>.446</td><td>.339</td><td>.511</td><td>.757</td><td>.452</td><td>.332</td><td>.514</td><td>.763</td><td>.438</td><td>.345</td><td>.515</td></tr>
<tr><td>SW-Attn+Encoder</td><td>.765</td><td>.411</td><td>.319</td><td>.498</td><td>.767</td><td>.423</td><td>.327</td><td>.506</td><td>.763</td><td>.410</td><td>.343</td><td>.505</td></tr>
</table>

Table 3: Class-level F1 scores of the Sig-Networks toolkit models on **Anno-MI** for different History Lengths. **Best** and <u>second best</u> scores are highlighted.

C Time Feature Guidance

As mentioned in §4.1 the toolkit allows for the automatic computation of the following time-derived features if a timestamp column is provided:

- `time_encoding`: date as fraction of the year
- `time_encoding_minute`: time as fraction of minutes, ignoring the date

[16] https://github.com/ttseriotou/sig-networks/tree/main/examples

[17] `signatory.logsignature_channels(10, 3)` can be used to compute this number.

[18] https://huggingface.co/bert-base-uncased

[19] Note in `transformers` (version 4.30.2), the default learning rate is 0.00005

Model	No Switch			Switch			Macro-avg		
BERT (focal)	.724			.454			.589		
BERT (ce)	.720			.472			.596		
FFN	.704			.457			.581		
FFN History	.727			.523			.625		
BiLSTM (w=5)	.730			.545			.637		
SWNU (w=5)	**.761**			.580			.670		
SW-Attn (w=5)	**.761**			.574			.667		
History Length (units)	**11 (n=3)**			**20 (n=6)**			**35 (n=11)**		
	No Switch	**Switch**	**Macro-avg**	**No Switch**	**Switch**	**Macro-avg**	**No Switch**	**Switch**	**Macro-avg**
BiLSTM	.748	.566	.657	.740	.555	.648	.748	.548	.648
SWNU	.759	.584	.671	.736	.571	.654	.759	.587	.673
SW-Attn	.745	.573	.659	.747	.583	.665	.743	.579	.661
Seq-Sig-Net	.760	.584	.672	.754	**.602**	**.678**	.748	.559	.654
SW-Attn+BiLSTM	.742	.584	.663	.741	.573	.657	.750	.570	.660
SW-Attn+Encoder	.746	.581	.664	.742	.572	.657	.756	.569	.662

Table 4: Class-level F1 scores of the Sig-Networks toolkit models on **Longitudinal Rumour Stance** for different History Lengths. **Best** and second best scores are highlighted.

Model	IS	IE	O	Macro-avg
BERT (focal)	.283	.439	.871	.531
BERT (ce)	.229	.431	**.903**	.521
FFN	.281	.432	.890	.534
FFN History	.280	.454	.877	.537
BiLSTM (w=5)	.260	.479	.892	.544
SWNU (w=5)	.301	.494	.894	**.563**
SW-Attn (w=5)	.300	.480	.887	.556

History Length (units)	**11 (n=3)**				**20 (n=6)**				**35 (n=11)**			
	IS	**IE**	**O**	**Macro-avg**	**IS**	**IS**	**O**	**Macro-avg**	**IS**	**IE**	**O**	**Macro-avg**
BiLSTM	.252	.478	.887	.539	.244	.470	.887	.533	.225	.460	.891	.525
SWNU	.292	.471	.887	.550	.275	.448	.888	.537	.270	.457	.889	.539
SW-Attn	.286	.471	.884	.547	.286	.453	.883	.541	.289	.452	.876	.539
Seq-Sig-Net	.301	**.495**	.893	**.563**	**.304**	.487	.891	.561	.303	.480	.894	.559
SW-Attn+BiLSTM	.291	.483	.887	.554	.298	.483	.890	.557	.298	.467	.885	.550
SW-Attn+Encoder	.289	.477	.890	.552	.302	.463	.891	.552	.294	.452	.887	.545

Table 5: Class-level F1 scores of the Sig-Networks toolkit models on **TalkLife MoC** for different History Lengths. **Best** and second best scores are highlighted.

- `time_diff`: time difference between consecutive data in the stream
- `timeline_index`: index of the data point in the stream

The option to include user-processed time features is available. Optionally, the user can specify a standardisation method for each time feature from the list below:

- `None`: no transformation applied
- `z_score`: transformation by subtracting the mean and dividing by the standard deviation of the data points
- `sum_divide`: transformation by dividing by the sum of the data points
- `minmax`: transformation by subtracting the minimum of data points from the current data point and dividing by the differential of the maximum and minimum of the data points.

The above (normalised) features can be included as part of the path stream in the signature model (*in-path*) and/or concatenated with the SBERT representation of the current data point in the input to the final FFN layers in the model (*in-input*) . During the different task modeling we find particularly important the efficient incorporation of time features. Such decision is task-driven.

For Anno-MI we include the `time_encoding_minute` and `timeline_index` (without transformation) *in-path*. For Longitudinal Rumour Stance we include `time_encoding` normalised with `z_score` and `timeline_index` without normalisation both *in-path* and *in-input*. Finally for TalkLife MoC we use `time_encoding` normalised with `z_score` both *in-path* and *in-input*. Since TalkLife and Longitudinal Rumour Stance are social media datasets they can benefit from the use of *in-input* features that model the temporal semantic component of linguistic representations. We expect *in-input* features to be less beneficial for our specific dialogue task which is semantically stable with conversations being date-agnostic (but not time agnostic). At the same time in the dialogue task of Anno-MI, the use of both the `time_encoding_minute`, which ignores the date, and `timeline_index` *in-path*, allows for modeling both the temporal flow of the conversation and the position (index) of the utterance of interest in the dialogue. While Longitudinal Rumour Stance also benefits from using the `timeline_index` which identifies the position of information with respect to the initial claim, the use of `time_encoding` normalised with `z_score` is more suitable here as it makes use of the date of the comment. In TalkLife only the latter is used, without any index features. Here, since relevant context for each post under consideration occurs in short history windows, the timeline position (index) is irrelevant. By presenting how different time features benefit each task together with the intuition behind the selection process, we encourage users to consider the temporal characteristics of their task in-hand for efficient time feature selection.

D Package Environment

The experiments ran in a Python 3.8.17 environment with the key following libraries: `sig-networks` (0.2.0), `nlpsig` (0.2.2), `torch` (1.9.0), `signatory` (1.2.6.1.9.0), `sentence-transformers` (2.2.2), `transformers` (4.30.2), `accelerate` (0.20.1), `evaluate` (0.4.0), `datasets` (2.14.2), `pandas` (1.5.3), `numpy` (1.24.4), `scikit-learn` (1.3.0), `umap` (0.5.3).

E Path Signature Libraries

Library	Link
`roughpy`	https://github.com/datasig-ac-uk/RoughPy
`esig`	https://github.com/datasig-ac-uk/esig
`iisignature`	https://github.com/bottler/iisignature
`signatory`	https://github.com/patrick-kidger/signatory
`signax`	https://github.com/Anh-Tong/signax

F Infrastructure

The experiments with the Anno-MI and Longitudinal Stance datasets were ran on the *Baskerville*, a GPU Tier2 cluster developed and maintained by the University of Birmingham in a collaboration with a number of partners including The Alan Turing Institute. Baskerville provided us access with Nvidia A100 GPUs (40GB and 80GB variants).

The experiments with the TalkLife dataset were ran on *Sanctus*, a Queen Mary University of London maintained server, with a x86_64 processor, 80 CPUs, 384 GB of RAM and 3 Nvidia A30 GPUs.

G Using the model modules

As noted in §4.3, we provide PyTorch modules for each of components of our Sig-Network models to encourage novel integration into other systems. For example, the key building blocks in each of our models are the Signature Window units, SWNU Tseriotou et al. (2023) and SW-Attn, as discussed in §3.4. These can be easily accessed in the toolkit with a few lines of Python code.

For example, in code listings 1 and 2 we can simply load in the SWNU and SW-Attn units and initialise an instance of the module in a few lines. For initialising SWNU in listing 1, we define several arguments: the input channels of our stream, `input_channels=10`, the number of output channels after the convolution-1d layer, `output_channels=5`, whether to take the log-signature or standard signature transformation, `log_signature=False`, the signature depth, `sig_depth=3`, the dimension of the LSTM hidden state(s), `hidden_dim=5`, the pooling strategy to obtain a final stream representation, `pooling="signature"`, to not chronologically reverse the order of the stream, `reverse_path=False`, to use a BiLSTM, `BiLSTM=True`, to use a convolution-1d layer, `augmentation_type="Conv1d"`. The alternative option for `augmentation_type` is to have `augmentation_type="signatory"` which will use the `signatory.Augment` PyTorch module to use a larger convolution neural network (CNN) for which you can specify the hidden dimensions to in the `hidden_dim_aug` argument which is set to `None` in this example. Note that some of these arguments have default values, but we present them all here for more clarity.

```python
from sig_networks.swnu import SWNU

# initialise a SWNU object
swnu = SWNU(
    input_channels=10,
    output_channels=5,
    log_signature=False,
    sig_depth=3,
    hidden_dim=5,
    pooling="signature",
    reverse_path=False,
    BiLSTM=True,
    augmentation_type="Conv1d",
)
```

Listing 1: Example initialisation of Signature Window Network Unit object

The SW-Attn unit, called `SWMHAU` in the library, shares many of the same arguments as expected but since we are using Multihead-Attention (MHA) in place of a (Bi)LSTM, we specify the number of attention heads through the `num_heads` argument and specify how many stacks of these layers through the `num_layers` argument. We can also specify the dropout to use in the MHA layer here too.

```python
from sig_networks.swmhau import SWMHAU

# initialise a SWMHAU object
swmhau = SWMHAU(
    input_channels=10,
    output_channels=5,
    log_signature=False,
    sig_depth=3,
    num_heads=5,
    num_layers=1,
    dropout_rate=0.1,
    pooling="signature",
    reverse_path=False,
    augmentation_type="Conv1d",
)
```

Listing 2: Example initialisation of SW-Attention unit object

Note that there are variants of these PyTorch modules which do not include the convolution 1d or CNN to project down the stream to a lower dimension before taking expanding window signatures, namely `sig_networks.SWLSTM` and `sig_networks.SWMHA`.

Once these objects have been created, they can simply be called to apply a forward pass of the units, see for example listing 3. These units receive as input a three-dimensional tensor of the batched streams and the resulting output is a two-dimensional tensor of batches of the fixed-length feature representations of the streams.

```python
    import torch

    # create a three-dimensional tensor of 100 batched streams, each with history length w and 10 channels
    streams = torch.randn(100, 20, 10)
```

```

    # pass the streams through the SWNU
    # swnu_features and swmhau_features are two-dimensional tensors of shape [batch, signature_channels]
    swnu_features = swnu(streams)
    swmhau_features = swmhau(streams)
```

Listing 3: Example forward pass of SWNU and SWMHAU objects

For full examples on how these PyTorch modules can be fitted into larger PyTorch networks, please refer to the source code for the Sig-Network family models in the library on GitHub[20].

[20]https://github.com/ttseriotou/sig-networks/tree/main/src/sig_networks