Scaling Up Authorship Attribution

Jacob Striebel[1], Abishek Edikala[2], Ethan Irby[2], Alex Rosenfeld[2], J. Blake Gage[2], Daniel Dakota[1,2], Sandra Kübler[1]
[1]Indiana University
[2]Leidos Inc.
{jstrieb,skuebler}@indiana.edu
{abishek.r.edikala,ethan.l.irby,alex.b.rosenfeld, john.gage,daniel.d.dakota}@leidos.com

Abstract

We describe our system for authorship attribution in the IARPA HIATUS program. We describe the model and compute infrastructure developed to satisfy the set of technical constraints imposed by IARPA, including runtime limits as well as other constraints related to the ultimate use case. One use-case constraint concerns the explainability of the features used in the system. For this reason, we integrate features from frame semantic parsing, as they are both interpretable and difficult for adversaries to evade. One trade-off with using such features, however, is that more sophisticated feature representations require more complicated architectures, which limit usefulness in time-sensitive and constrained compute environments. We propose an approach to increase the efficiency of frame semantic parsing through an analysis of parallelization and beam search sizes. Our approach results in a system that is approximately 8.37x faster than the base system with a minimal effect on accuracy.

1 Introduction

Authorship attribution aims to identify the correct author of a document. The IARPA Human Interpretable Attribution of Text using Underlying Structure (HIATUS) program looks to develop novel methods to address several of the current limitations of authorship attribution, with specific consideration given to explainability, higher linguistic features, generalizability, and privacy preservation. In information warfare, operatives adopt local writing patterns in order to infiltrate and influence populations and manipulate legitimate political discourse. While they assume the local patois and customs of the target audiences they are trying to infiltrate, humans cannot completely erase all traces of higher-order authorship characteristics (e.g., socialization, education, culture, and characteristics of their native language). Our system is designed to detect such infiltrators, but should also allow distribution of information without revealing the source. While this approach is tailored towards meeting the HIATUS requirements, these requirements translate into realistic scenarios. For this reason, we assume that our solutions will be usable in real world applications, with additional consideration given to external factors (e.g. costs, time, deployability). Thus, any lessons learned from its development are directly applicable to any related production-level system that uses the same features or focuses on similar tasks.

An overall system should be composed of three interconnected modules: feature generation, authorship attribution, and privacy preservation, with the latter two possessing explainability requirements. An underlying challenge in our applications is that any software must be usable by the Intelligence Community, which entails strict technical restrictions. For this reason, HIATUS systems are required to be fully contained and executable within a pre-defined set of environments as defined by IARPA. This requires a balance between delivering a highly effective solution and maintaining the ability to deploy our solution within the bounds of the technical constraints. Here, we focus on authorship attribution. We evaluate the speed of the overall attribution system, and specifically optimize one of the models for creating features derived from a semantic parser, since the parser requires several passes over a sentence, among other limitations, and thus does not scale sufficiently.

2 Related Work

Authorship analysis refers to a range of tasks that require modeling language use with the goal of grouping texts together that are written by the same author and discriminating among texts that are written by different authors (Bischoff et al., 2020). In order to facilitate authorship analysis, recent work has focused on extracting authorship embedding representations from documents, often leveraging

Large Language Models (LLMs; Rivera Soto et al., 2021; Stubbemann and Stumme, 2022) or other deep learning models (Boenninghoff et al., 2019, 2021; Saedi and Dras, 2021) to aggregate authorship information.

While capturing sophisticated representations, the black-box nature of the models do not make them as inherently usable when explainability is required. This has resulted in work developing explainable authorship embeddings (Patel et al., 2023) where each component corresponds to an interpretable feature.

The two most common authorship analysis subtasks are authorship attribution and authorship verification.

2.1 Authorship Attribution

Authorship attribution is a closed-set classification task where the goal is to decide who in a fixed set of candidate authors is most likely to have written a given anonymous text (Fabien et al., 2020; Ferracane et al., 2017; Kestemont et al., 2019; Saedi and Dras, 2021). Contemporary approaches to authorship attribution frequently mirror approaches used to perform other text classification tasks, such as employing a CNN classifier, with character n-grams as input features (Ferracane et al., 2017) or incorporating pre-trained contextualized embedding model (e.g., a BERT model) with an MLP prediction layer (Fabien et al., 2020).

One real-world application is forensic linguistics, where authorship attribution may be applied within the context of a legal process to help detect a text's most likely author among several suspects (Ainsworth and Juola, 2019; Fobbe, 2020).

2.2 Authorship Verification

Authorship verification, in contrast, is an open-set similarity-based task in which the goal is to compute a measure of authorial similarity between any two texts (Boenninghoff et al., 2021; Stubbemann and Stumme, 2022). Authorship verification is often performed as an embedding task where the input is a document, similar to authorship attribution, but the output is an authorship fingerprint vector, unlike the categorical output in the former. The distance between the authorship fingerprint vectors of any two documents is then used as a measure of authorial similarity. Document embedding models based on extending and fine-tuning Transformers have been used (Rivera Soto et al., 2021; Stubbemann and Stumme, 2022), as well as models that employ other neural architectures (Boenninghoff et al., 2019, 2021; Saedi and Dras, 2021).

3 System Task and Constraints

3.1 Task

While HIATUS consists of three separate tasks, we focus on the first task consisting of feature space generation. The task is to generate document vectors of authors that accurately and distinctly encode individual authors' "fingerprints." The vector representations need to be able to capture enough information about an author to enable the system to perform a successful authorship attribution, regardless of other considerations such as genre, topic, or domain.

3.2 Data

While the system is ultimately ingesting and running on HIATUS specific data, during development we experiment with open-source data sets. This allows for easier in-depth experimentation and comparison with existing approaches. We experiment with a small subset of the Reddit Million User Dataset (Baumgartner et al., 2020; Rivera Soto et al., 2021) to reduce our runtime experiments with the Frame Semantic Transformer (see section 3.3). Reddit is a good proxy as it is noticeably diverse in authors and writing styles, which resembles our ultimate use case. We select a further subset of authors consisting of eight sentences from 50 different authors as our query authors and include an additional 350 authors as our candidate pool. The goal is to rank the correct authors highly over the stack of 400 total authors.

3.3 Features

Contemporary approaches to authorship analysis often use contextualized word embeddings, produced by a Transformer, as input features to the authorship analysis model (e.g., Stubbemann and Stumme, 2022). This type of input feature has yielded good results in many cases, but it has the drawback of lacking interpretability in the sense that feature attributions to individual words will be of limited use to someone trying to understand an authorship prediction. Instead, feature attributions to higher-level language structures, such as syntactic, semantic, and discourse features, will be more useful when the system needs to explain a prediction.

Our system employs several higher-level input features in order to enable a more interpretable system that does not rely exclusively on contextualized word embeddings. While the full system uses syntactic-dependency, semantic, stylometric, rhetorical style, and additional document features, our current work focuses on semantic features using the FrameNet annotation scheme (Ruppenhofer et al., 2016). Previous work has used semantic frames as input features for authorship attribution (Hedegaard and Simonsen, 2011a). We extend this by predicting and incorporating both semantic frame and fame elements using the T5 generative-based Frame Semantic Transformer (FST; Chanin, 2023) into our authorship model.

FrameNet annotation of a sentence consists of two steps: (1) annotation of each frame evoking word in the sentence (that is covered by the FrameNet lexicon) for a semantic frame (similar to word sense identification), and (2) annotation of frame elements per frame evoking word (akin to annotating verb valency, but generalized to all word classes). For example, consider the sentence: "John suffered an injury." In the first step, we identify that "suffered" evokes the `Catastrophe` frame and "injury" evokes the `Medical_conditions` frame. Then in the second step, with respect to the former frame, we identify that the `Patient` frame element is "John" and the `Undesirable_event` frame element is "an injury," and with respect to the latter frame we identify that `Patient` frame element is again "John."

3.4 Evaluation

We evaluate the system for speed and quality, as we are constrained by a time limit and architecture specified by HIATUS. We compare run times of the frame semantic parser along with its performance on frame semantic analysis and authorship attribution.

Frame Semantic Analysis To evaluate the generation of our frame semantic features, we tested the system on the test partition of FrameNet employed in other recent work (e.g., Swayamdipta et al. (2017); Chanin (2023)), consisting of 2,420 sentences collected from different documents. We provide runtime for each configuration, and we also provide trigger identification accuracy, frame identification accuracy, and argument identification F1 score. Trigger identification corresponds to marking each word in the input sentence that evokes a semantic frame; frame identification corresponds to classifying the particular frame that is evoked by each trigger; and argument identification corresponds to delimiting the spans of each frame element.

Authorship Attribution Evaluation We provide two standard metrics: Precision@K (here k=8) and mean reciprocal rank (MRR). Precision@K is calculated by creating the vector representations for selected sentences in the Reddit data, computing cosine pairwise similarities, and ranking them. In addition, we record runtime speeds for the different implemented configurations.

3.5 Technical Constraints

HIATUS imposes several runtime constraints that any implemented system must run within. This means we must replicate the evaluation infrastructure in our own environment to ensure our deployed systems fits within these constraints and runs as expected after submitting for evaluation. Other constraints include no external network connections (e.g., no access to external APIs), thus all binaries and required libraries must be available locally at runtime and additional considerations should be given to cost and speed.

While we focus here on only one of three tasks, Docker images for each of the tasks cannot exceed 50GB and must enable dynamic data flow between the required components / tasks during evaluation within the single chosen instance (e.g., generated features must be accessible for other tasks at runtime). Additionally, each system evaluation must be completed in 12 hours or less. A possible limitation of our system is that the longest document in the validation set is 350 words, which limits our ability to test if the system can successfully be executed on longer documents at evaluation time.

Figure 1 illustrates the official evaluation infrastructure of HIATUS and all sub components that undergo official testing. Although we focus on the evaluation protocol for evaluating authorship embeddings, the same process applies to all task areas. The performance and inference speed are evaluated once the Docker container is loaded into the infrastructure environment (listed as "Performer Model Docker"). If the Docker container completes the entire inference within the 12 hour time limit, the evaluation process is initialized.

The evaluation process is initiated by loading the

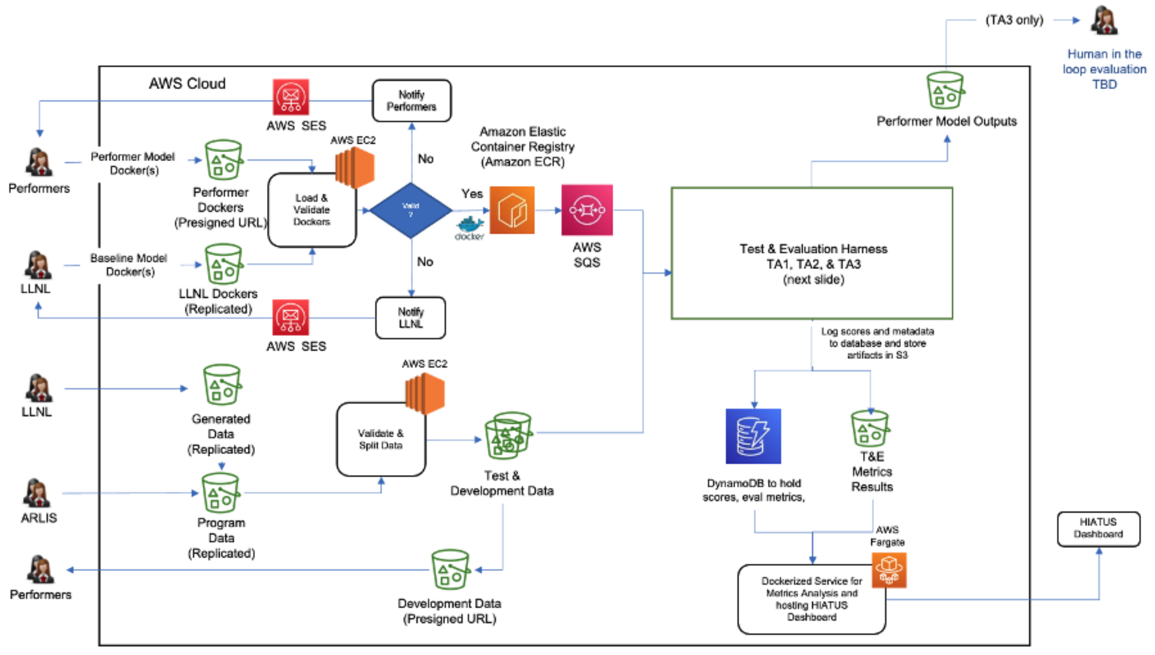
(TA3 only)
Human in the loop evaluation TBD
AWS Cloud
AWS SES
Notify Performers
Performer Model Outputs
Performers
Performer Model Docker(s)
Performer Dockers (Presigned URL)
AWS EC2
Load & Validate Dockers
No
Valid ?
Yes
Amazon Elastic Container Registry (Amazon ECR)
docker
AWS SQS
Test & Evaluation Harness TA1, TA2, & TA3 (next slide)
LLNL
Baseline Model Docker(s)
LLNL Dockers (Replicated)
No
AWS SES
Notify LLNL
Log scores and metadata to database and store artifacts in S3
LLNL
Generated Data (Replicated)
AWS EC2
Validate & Split Data
Test & Development Data
DynamoDB to hold scores, eval metrics,
T&E Metrics Results
ARLIS
Program Data (Replicated)
AWS Fargate
HIATUS Dashboard
Performers
Development Data (Presigned URL)
Dockerized Service for Metrics Analysis and hosting HIATUS Dashboard

Figure 1: HIATUS evaluation infrastructure. Image reproduced from the Evaluation Plan with permission.

Docker container and sending it to the the Amazon Elastic Container Registry (ECR). Amazon ECR allows for ease of storing and deploying the individual Docker containers within the architecture. Once stored in ECR, an AWS Simple Queue Service (SQS) notifies the official metrics harness to begin the process of generating metrics. Here, test documents are passed to the Docker container. When the system completes inference, official metrics are logged and saved to an AWS S3 storage instance, which are then accessible to be displayed on the official metric dashboard.

In the case where the Docker container does not finish inference within the allotted time, no metrics are produced or logged, resulting in a failed system run. This is one reason why speed optimization is necessary after a system is built to ensure a complete successful run within the time constraint.

4 Implemented Architecture

4.1 Base Architecture

The designed infrastructure allows for continuous development and integration to support model development, storing training and testing data, and strategic GPU and memory optimization for training LLMs.

The data stream first ingests multi-line JSON files, where each line in a file is a single document written by an author in the corpus. Subsequently, the feature extraction modules are run on each document to generate a vector that includes psychometric, dependency, rhetorical style, and additional document features used to create document embedding. Each embedding is saved to the AWS Elastic Block Storage (EBS) for later retrieval on and are accessible for any subsequent downstream applications.

4.2 Architecture Modifications

When testing the complete system in an environment with the limitations specified by HIATUS, it became clear that there were two parts of the architecture that needed to be modified: 1) generating unique authorship embeddings came with a large storage price as features are placed directly into the EBS volume on the EC2 instance where the model performs inference and generates new features, 2) the frame semantic parser requires several passes over each sentence, which poses a problem in terms of runtime efficiency.

To alleviate large storage requirements, an AWS S3 cloud storage is secured and mounted to the EC2 instance in which the system is run end-to-end. The additional storage space provided by an S3 "bucket" alleviates the need to store large authorship vectors on an EBS volume, where reaching the storage limit would prevent the creation of future embeddings and place a limit on the development environment.

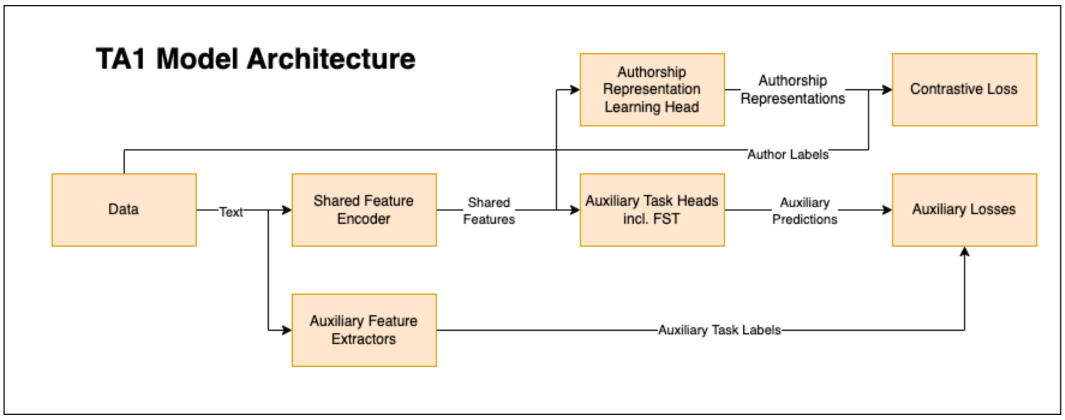

Figure 2: TA1 Author Embedding Architecture

The default runtime efficiency of the semantic parser in our environment is only five sentences per second. To increase computational efficiency and thus scalability, we implemented multiprocessing support, enabling the parser to run in a distributed fashion across all available GPUs while still ensuring successful inter-process communication and synchronization among processes. As part of the infrastructure development, multiple implementations were tested to support the distributed processing capability as the evaluation system came with its own unique set of issues (see section 3.5).

4.3 Author Embedding Architecture

Our authorship encoder focuses on synergizing neural and theory-driven features to learn a common representations of authorship that can be used for authorship attribution and explainability. Learning authorship representation consists of encoding only the distinct features of an author's style. Traditionally, hand-crafted, theory-driven features were used. While these approaches are data efficient, they can be limited in definition. Neural models, on the other hand, do not require explicit definition but rely on the training data to learn relevant features.

Training consists of a multi-task learning (Caruana, 1997) architecture in which we train a transformer encoder for authorship attribution and include several auxiliary tasks (e.g., frame semantic features (Hedegaard and Simonsen, 2011b)), selected based on their theoretical importance to authorship attribution (see Figure 2). During inference only the encoder's output is used. This setup enables the authorship attribution embeddings to learn more nuanced, higher linguistic information about the author from the auxiliary tasks through information sharing, yielding more distinguishable embeddings.

To facilitate rapid experimentation, we have created a feature extractor module that takes in corpora and extracts features that are used as labels for the auxiliary tasks. New auxiliary tasks can be added and experimented with easily if features are extracted in real-time during training, as opposed to having to prepare training data beforehand. In practice, frame semantic feature extraction (and other feature extractors) reside inside of this module.

Furthermore, auxiliary tasks provide transparency during training and explainability during inference. During training, the loss the auxiliary tasks can be monitored to see which theory-backed features are being learned over time as the authorship encoder learns. For inference, the human-understandable auxiliary features can be predicted to associate them to neural features that are used for attribution. These features can then be used to highlight tokens with a framework like SHAP (Lundberg and Lee, 2017) so that forensic analysts can quickly verify the attribution. Our system therefore leverages black box attributions and provides human-understandable explanations that can be used for verification via highlighting various aspects of authorship within text.

4.4 Parallelizing Frame Semantic Features

Given that the solution must fit within the specified time constraints (otherwise no results will be recorded), initial time estimates using the default parser suggested that including it in the pipeline would result in the entire system taking 8x longer than the maximal time-frame, making it impractical to include in the feature generation. We chose to thus implement a multiprocessing variation and

	Runtime (secs)	Rate (sent/sec)	Trigger ID	Frame ID	Argument ID
No MP, Beam=5	459.7	5.26	0.7127	0.8643	0.7324
MP, Beam=5	184.7	13.10	0.7127	0.8643	0.7324
MP, Beam=3	153.6	15.76	0.7143	0.8631	0.7194
MP, Beam=1	72.3	33.47	0.4983	0.9036	0.7414

Table 1: Section: Evaluation results on FrameNet parsing task.

gained additional speed by reducing its beam size.

The specific architecture that we employed to achieve parallelization uses a controller–worker pattern. Invoking our application creates the controller process, which in turn creates a separate worker process for each available GPU. We tested two versions of this architecture.

Sentence groups subbatches read in batches by the controller process, and all of the sentences in a batch of documents are divided into equally sized groups, with the number of groups equal to the number of worker processes. The controller process uses inter-process communication to submit a subbatch to each worker process, and then the controller waits to receive the results from each worker. Each worker process runs its own instance of the parser, and each worker has exclusive use of one of the GPUs in our AWS compute environment. Once the controller process receives the subbatch results from each worker process, the parsed sentences in each subbatch are collated back into their original documents in the higher-level batch, and then each processed document in the batch is written as output by the controller in the same order that it was read as input. This process of batchwise processing is repeated continuously until all documents in the input file have been parsed.

Equal group subbatches reads all documents at once by the controller and then splits them into equally sized groups, with the number of groups equal to the number of GPUs. The controller then starts worker processes for each segment of the dataset to extract frame semantic features. After the worker processes finish running, the controller concatenates the outputs to return the final output. This method circumvents the need for inter-process communication to process subbatches.

While initial experiments showed both approaches yielded similar results, we ultimately chose the equal group subbatches implementation for our system as given that the entire datasets fits in memory, it provides a simpler, working alternative.

	Runtime (sec.)	P@8	MRR
No MP, Beam=5	1456.9	0.231	0.117
MP, Beam=5	462.4	0.231	0.112
MP, Beam=3	368.4	0.255	0.128
MP, Beam=1	174.6	0.240	0.121

Table 2: Results of the authorship attribution system. P@8 vs Beamsize (or MP)

5 Results

5.1 Performance of the Frame Semantic Parser

The evaluation of the frame semantic parser focuses on two points, speed and and beam size decoding. The results on the test partition across several configurations are presented in Table 1. Enabling multiprocessing using four GPUs, as opposed to the baseline approach using a single GPU, produced an increase in parsing rate from 5.26 sentences per second to 13.10, a 2.5x increase in speed. While keeping multiprocessing enabled and decreasing the decoding beam size from five to three, the parsing rate further improves from 13.10 sentences per second to 15.76. Decreasing the beam size from five to three produced negligible reduction in parsing accuracy. Further decreasing the beam size from three to one led to a further increase in parsing rate from 15.76 sentences per second to 33.47. However, this final decrease in beam size led to a substantial degradation in trigger identification. Therefore the best performing setup is to enable multiprocessing with the decoding beam size set to three, which is still almost a 3x speed increase of the base parser.

5.2 Performance on Authorship Attribution

Authorship attribution results are presented in Table 2. We report Precision@8 (P@8) and mean reciprocal rank (MRR) on the subset of the Reddit dataset (see section 3.2). While there are only minimal differences in performance across the various configurations, multiprocessing and beam search reduction yields a system that only takes 12% of runtime of the original implementation. Interestingly, while a beamsize of three still yields the best results, the performance degradation when using only a beamsize of one is not nearly as substantial on the authorship attribution performance when compared to the trigger identification results in Table 1, while being almost 50% faster. This suggests this feature is either not highly relevant or its performance loss is mitigated by the other incorporated frame features.

6 Conclusion and Future Work

We detailed our system for authorship attribution within the HIATUS environment, which has an infrastructure that poses overhead limits to the compute and memory configurations of the specific task of creating authorship embeddings. Given these constraints, we highlighted our computational efficiency modifications to the frame semantic parser in order to be able to integrate frame semantic linguistic features into our authorship attribution model. Our architecture generates authorship embeddings that are successfully evaluated on the Test & Evaluation Harness, while resulting in negligible performance reduction. The integrated modifications are directly applicable to any other resource constrained production system focused on authorship attribution or using similar features. Future modifications will include more robust parallelization of the feature generation ability in addition to an optimization step that includes automating the Docker image creation for the task and submission to the HIATUS evaluation infrastructure.

Ethical statement

While the goal of our authorship attribution system is to identify malicious actors to aid the Intelligence Community, we recognize that we cannot guarantee that such a system will never produce false positives. Thus we strongly encourage authorship attribution be used as merely a point of support in combination with additional data sources, tools, and metrics; and no single strategy be relied upon as a sole source of intelligence for any subsequent decision or action taken.

Acknowledgments

The authors acknowledge the Indiana University Pervasive Technology Institute for providing supercomputing and storage resources that have contributed to the research results reported within this paper.

This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute.

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract #2022-22072200002. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Janet Ainsworth and Patrick Juola. 2019. Who wrote this: Modern forensic authorship analysis as a model for valid forensic science. Washington University Law Review, 96:1159–1188.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. volume 14, pages 830–839.

Sebastian Bischoff, Niklas Deckers, Marcel Schliebs, Ben Thies, Matthias Hagen, Efstathios Stamatatos, Benno Stein, and Martin Potthast. 2020. The importance of suppressing domain style in authorship analysis. CoRR, abs/2005.14714.

Benedikt Boenninghoff, Steffen Hessler, Dorothea Kolossa, and Robert M. Nickel. 2019. Explainable authorship verification in social media via attention-based similarity learning. In 2019 IEEE International Conference on Big Data (Big Data), pages 36–45.

Benedikt Boenninghoff, Dorothea Kolossa, and Robert M. Nickel. 2021. Self-calibrating neural-probabilistic model for authorship verification under covariate shift. In Experimental IR Meets Multilinguality, Multimodality, and Interaction: 12th International Conference of the CLEF Association, CLEF 2021, Virtual Event, September 21–24, 2021, Proceedings, page 145–158, Berlin, Heidelberg. Springer-Verlag.

Rich Caruana. 1997. Multitask learning. Machine Learning, 28(1):41–75.

David Chanin. 2023. Open-source frame semantic parsing. https://arxiv.org/abs/2303.12788.

Maël Fabien, Esau Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. 2020. BertAA : BERT fine-tuning for authorship attribution. In Proceedings of the 17th International Conference on Natural Language Processing (ICON), pages 127–137, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLPAI).

Elisa Ferracane, Su Wang, and Raymond Mooney. 2017. Leveraging discourse information effectively for authorship attribution. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 584–593, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Eilika Fobbe. 2020. Text-linguistic analysis in forensic authorship attribution. JLL, 9:93.

Steffen Hedegaard and Jakob Grue Simonsen. 2011a. Lost in translation: Authorship attribution using frame semantics. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 65–70, Portland, Oregon, USA. Association for Computational Linguistics.

Steffen Hedegaard and Jakob Grue Simonsen. 2011b. Lost in translation: Authorship attribution using frame semantics. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 65–70.

Mike Kestemont, Efstathios Stamatatos, Enrique Manjavacas, Walter Daelemans, Martin Potthast, and Benno Stein. 2019. Overview of the Cross-domain Authorship Attribution Task at PAN 2019. In CLEF 2019 Labs and Workshops, Notebook Papers. CEUR-WS.org.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 4765–4774. Curran Associates, Inc.

Ajay Patel, Delip Rao, Ansh Kothary, Kathleen McKeown, and Chris Callison-Burch. 2023. Learning interpretable style embeddings via prompting LLMs. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 15270–15290, Singapore. Association for Computational Linguistics.

Rafael A. Rivera Soto, Olivia Elizabeth Miano, Juanita Ordonez, Barry Y. Chen, Aleem Khan, Marcus Bishop, and Nicholas Andrews. 2021. Learning universal authorship representations. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 913–919, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. FrameNet II: Extended Theory and Practice.

Chakaveh Saedi and Mark Dras. 2021. Siamese networks for large-scale author identification. Computer Speech & Language, 70:101241.

Maximilian Stubbemann and Gerd Stumme. 2022. Lg4av: Combining language models and graph neural networks for author verification. In Advances in Intelligent Data Analysis XX: 20th International Symposium on Intelligent Data Analysis, IDA 2022, Rennes, France, April 20–22, 2022, Proceedings, page 315–326, Berlin, Heidelberg. Springer-Verlag.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. https://arxiv.org/abs/1706.09528.