Author Classification using Traditional NLP and Modern Deep Learning Methods

Team Bayesian Banshee

- Dean Weiss, Nikita Parulekar, Hunter Worssam

Final Presentation

Contents

- Abstract
- Dataset Creation & EDA
- Traditional NLP Preprocessing
- Traditional Classifiers
- Transformers
- Results comparison

Abstract

Abstract

We explored author classification using the Project Gutenberg corpus by training models on full-length books from 80 unique authors.

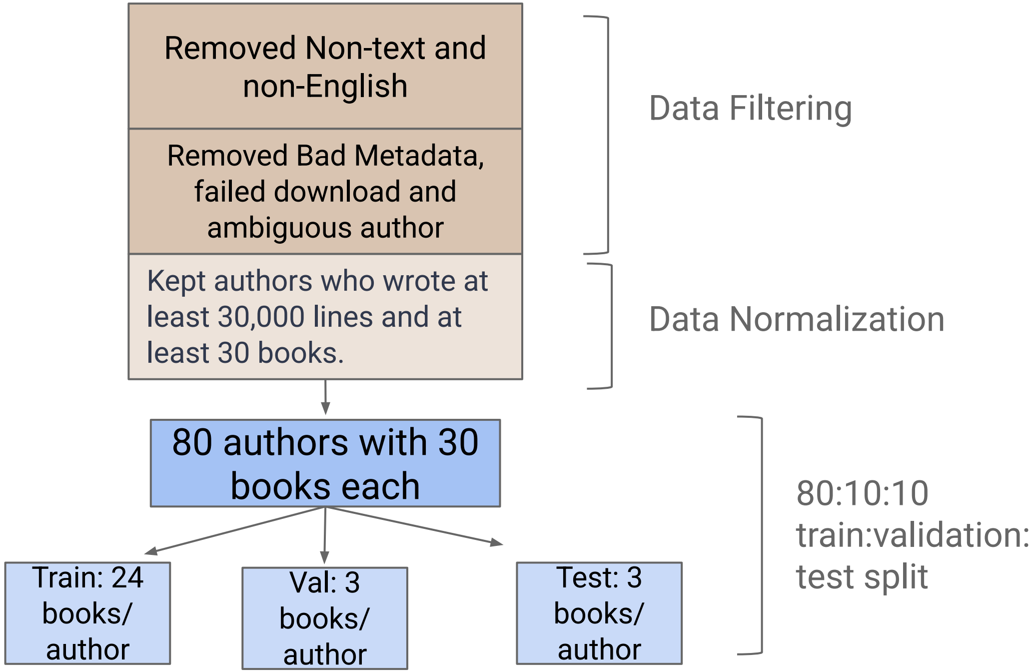
Our analysis compared traditional algorithms (Naive Bayes, kNN, Random Forest, SVM, and MLP) with a more modern Transformer-based approach.

Dimensionality reduction & additional pre-processing was applied where appropriate, and models were optimized through hyperparameter tuning.

Results showed that Support-Vector Machine and Multilayer Perceptron models achieved the highest accuracy, while the Transformer, Naive Bayes, Random Forest and kNN also yielded impressive results relative to literature.

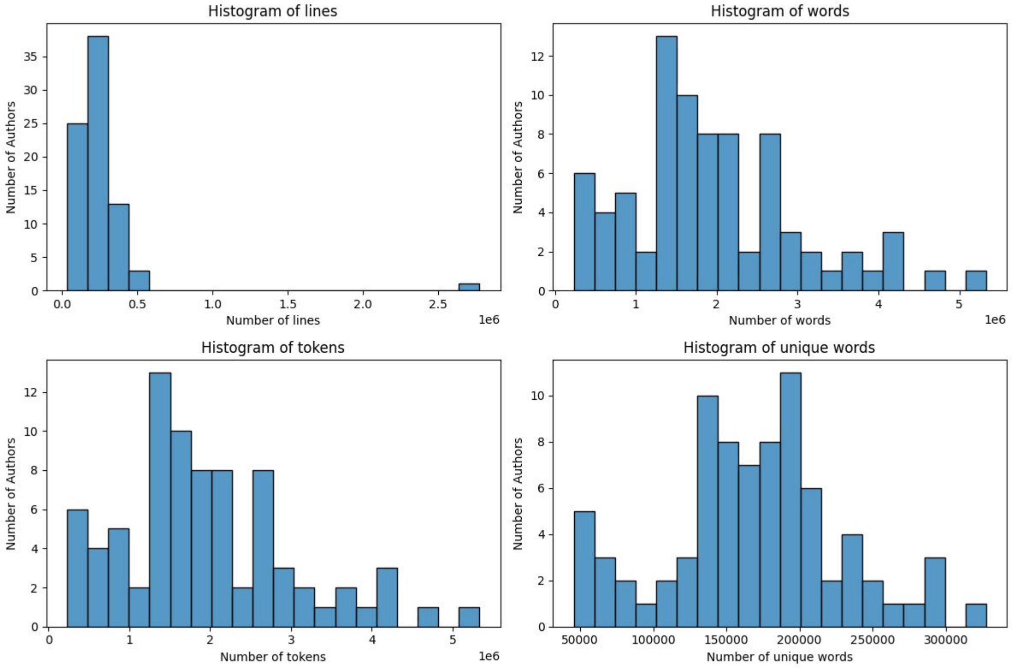
Dataset Creation and EDA

Dataset Creation Starting from All 75k PG Texts



Dataset EDA

- Wide range of documents across ‘authors’ from Mark Twain to the National Park service guides
- 81% of authors have books with > 1 million words



Traditional NLP Data Pre-Processing

Stemming and Lemmatization: root word creation

Text normalization processes used to **reduce a word to its base form**

Stemming	Lemmatizing
• Uses simple rules • Usually just strips suffix • Stemmed version may not be real word	• Uses a dictionary, the part of speech, and grammatical rules to know what a word means • Outputs dictionary form of word (lemma) • Contextual understanding
Running → Runn Better → Bett Happiness → Happi Played → Play	Running → Run Better → Good Happiness → Happy Played → Play

TF–IDF: how important a word is to a document

- **Term Frequency (TF):** how often a word appears in a document.
- **Inverse Document Frequency (IDF):** how rare a word is across all documents.
- The product of TF and IDF **highlights important, distinguishing words** while down-weighting common terms.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

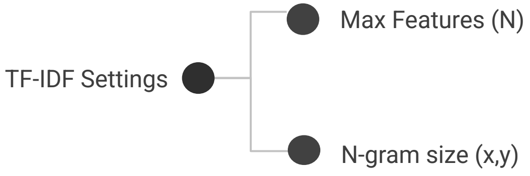
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Why Use TF-IDF?

- Converts raw text into numeric feature vectors usable by machine learning models.
- Scores & keeps top N word sequences of size 1, 2, … y; set via the n-gram setting.
- Applied TF-IDF vectorization to cleaned text



Stylometric Features Added to Enhance TF-IDF Feature Matrix

For Naive Bayes, Random Forest and kNN, **12 stylometric features & 6 readability metrics** were generated from each text, including:

Stylometric Features

- **avg_word_length** – average number of characters per word
- **avg_sentence_length** – average number of words per sentence
- **type_token_ratio** – lexical diversity (unique words / total words)
- **punctuation frequencies** – relative use of `, . ; : ! ? -`
- **uppercase_ratio** – proportion of capital letters
- **digit_ratio** – proportion of numeric characters

Readability Metrics

- **Flesch Reading Ease**
- **Flesch-Kincaid Grade Level**
- **Gunning Fog Index**
- **SMOG Index**
- **Coleman-Liau Index**
- **Automated Readability Index**

1. Texts **cleaned and tokenized** using `nltk toolkit`.
2. Features calculated **per document** using custom Python functions and `textstat` (18 new features per document).
3. Stylometric features were both kept and removed during model training to evaluate their impact on accuracy.
4. **For Naive Bayes and kNN (magnitude and distance based models),** these **features** were **normalized;** but this was **not required for Random Forest model.**

Classical ML methods

Naive Bayes

Bayes' Theorem gives us the posterior probability of a class (e.g., an author) given features X_1, X_2, ..., X_n:

$$P(C \mid X_1, X_2, ..., X_n) = \frac{P(C) \cdot P(X_1, X_2, ..., X_n \mid C)}{P(X_1, X_2, ..., X_n)}$$

This is hard to compute this directly because the joint probability portion in the numerator is complex — features may be dependent on each other.

For **Naive Bayes**, we assume all features are **conditionally independent given the class**:

$$P(X_1, X_2, ..., X_n \mid C) \approx \prod_{i=1}^{n} P(X_i \mid C)$$

This simplifies computation and lets us estimate probabilities from frequencies (which we calculated directly through TF-IDF).

Parameter	Optimal Setting
TF-IDF Max Features	15,000
TF-IDF n-gram Size	(1, 3)
Model Variant	Multinomial
Alpha	0.1

Dataset	F1 Score
Training	0.9792
Validation	0.9376
Test	0.9520

Random Forest

Random Forest is an **ensemble method** that builds multiple decision trees and averages their predictions.

Random Forest is an attractive model for author classification:

- Robust to multicollinearity & noise.
- Handles high dimensional, sparse data.
- Ensemble nature reduces overfitting risk.

Parameter	Optimal Setting
# of Trees	300
min_samples_split	5
min_samples_leaf	2
Max Depth	None

Dataset	F1 Score
Training	1.0000
Validation	0.9455
Test	0.9263

TF-IDF settings of 15,000 features of size (1,3) were used for Naive Bayes, Random Forest, and kNN to maintain comparability.

Only model-specific hyperparameters were folded into the tuning stage.

kNN

kNN is a **non-parametric, distance-based** learning algorithm that classifies a sample based on the majority label among its *k* nearest neighbors in the feature space.

A downside to the distance-based nature of kNN is that it is increasingly prone to the **curse of dimensionality**.

To account for this, we use **singular value decomposition (SVD)** to trim our TF-IDF feature matrix from 15,000+ features down to 200-1000.

Stylometric features were particularly valuable to the kNN due to their **low-dimensional, dense signal** for writing style.

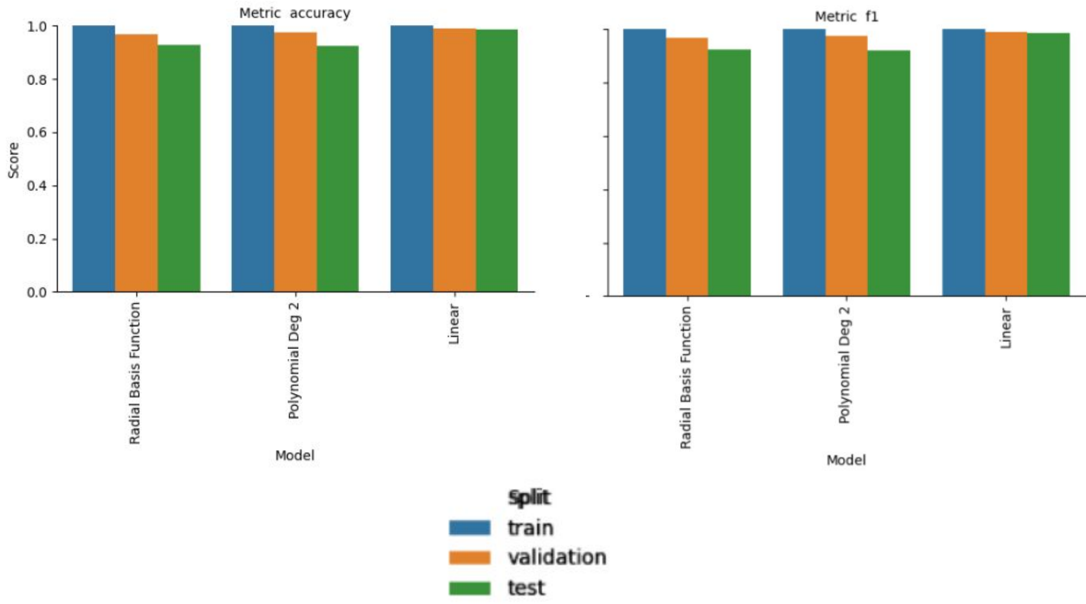
- When SVD Components is set to 18, all 18 stylometric features are kept above TF-IDF values.

Parameter	Optimal Setting
SVD Components	300
n-neighbors	3
Distance Metric	Cosine

Dataset	F1 Score
Training	1.0000
Validation	0.9438
Test	0.9344

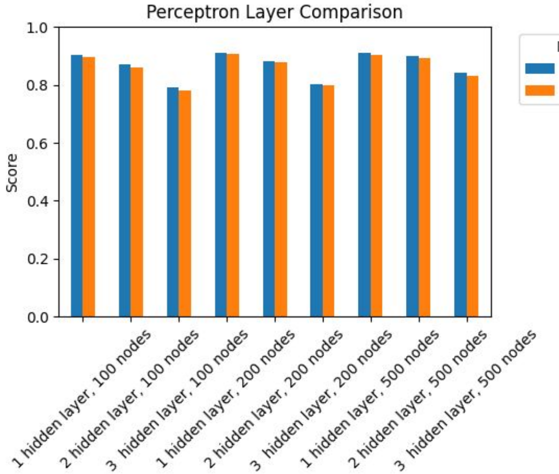
Support Vector Machine – Choosing Kernel

- Linear kernel outperformed radial basis function and polynomial (deg 2) kernel on validation (and test) set
- Features already capture non-linear complexity between authors’ styles and the classes are linearly separable
- Degradation of validation and test set performance for RBF and polynomial kernel -> signs of overfitting relative to linear kernel

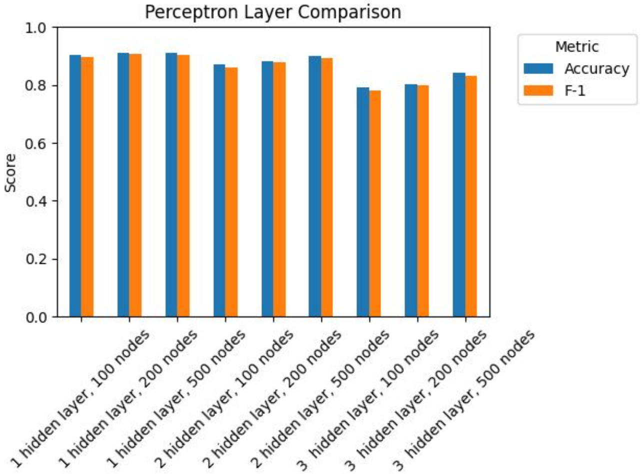


Multi-Layer Perceptron – simpler the better

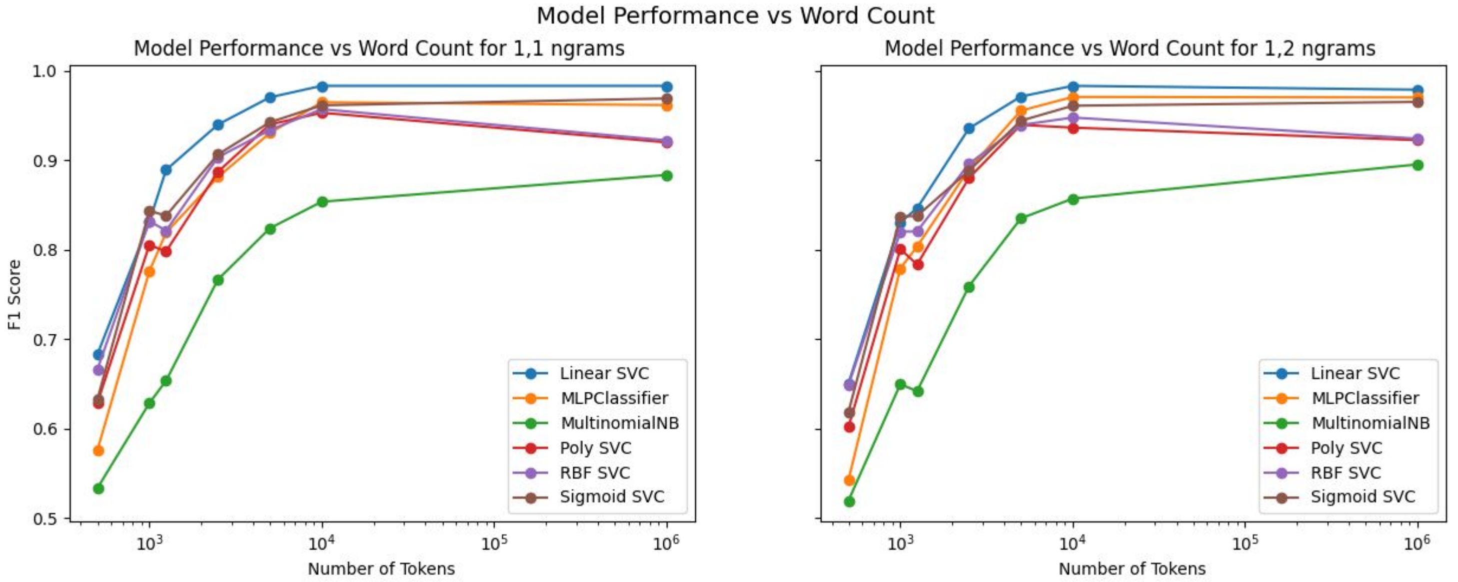
More hidden layers = worse performance -> overfitting



Leaving number of hidden layers constant, increasing node size little, if any benefit

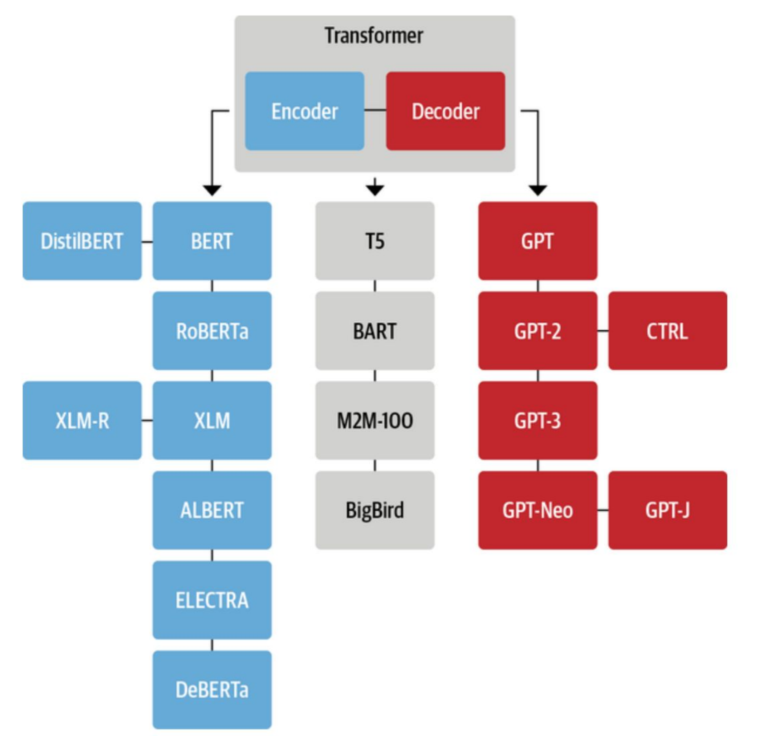


Impact of Word Count and N-Grams



All classifiers benefit from increased word count, little benefit from bi-grams, though have diminishing returns

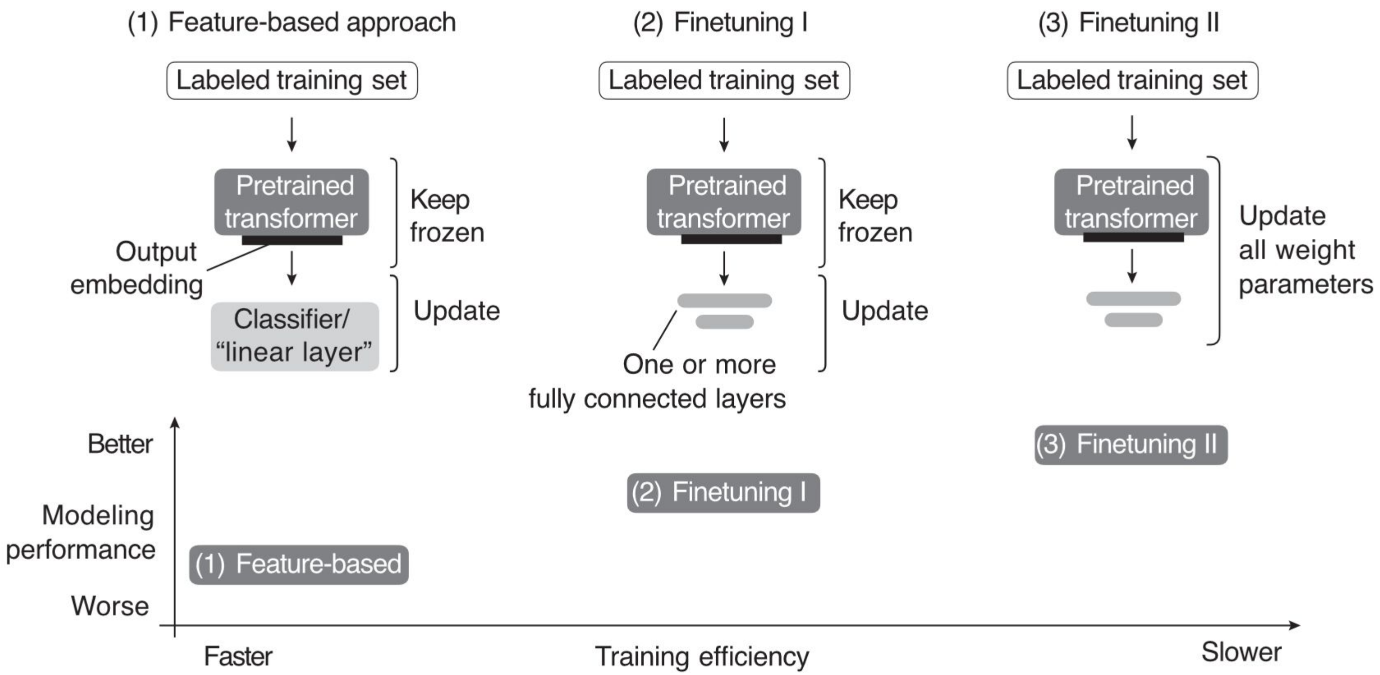
Transformers



Fine-Tuned transformers have more practical compute and data requirements

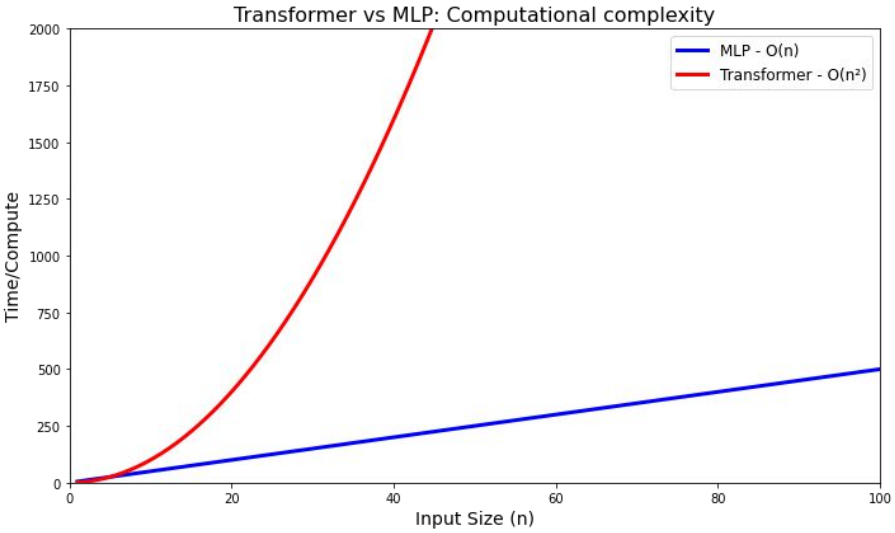
Aspect	Train From Scratch	Fine-Tuned Transformer
Data Requirement	Requires large datasets (100,000s+ examples)	Works well with small datasets (100s–1000s of examples)
Compute & Time	High compute, weeks to months	Low compute, hours to days
Model Customization	Full architectural control	Limited to existing architectures
Performance Ceiling	Potentially higher (with enough data)	Lower, capped by pre-training limitations

Finetuning II approach was chosen for best modelling performance

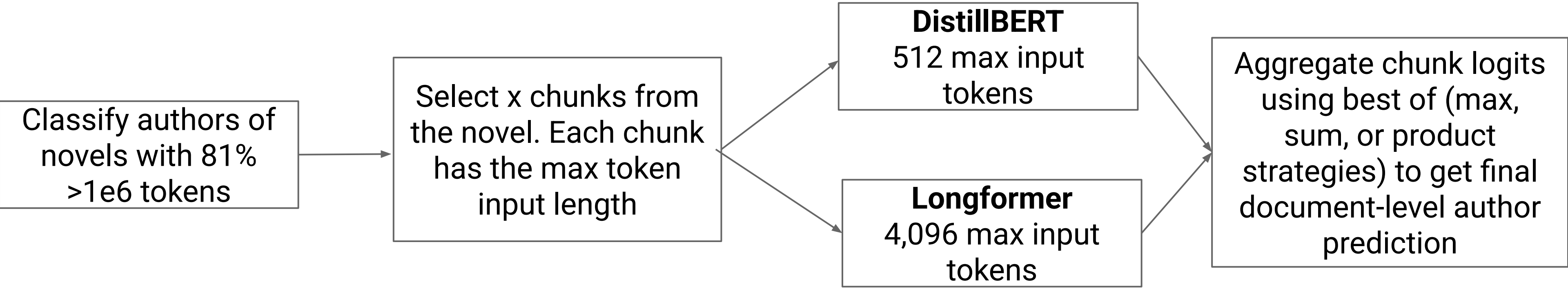


Transformers have limits on input length

- Due to token interaction in the attention mechanism, time and compute requirements scale quadratically with sequence length
- 81% of our documents have **>1e6 words** while most **regular** pretrained encoder **transformers** have **token input limits 512-1000**
- **Sparse attention** transformer models such as Longformer can handle longer **token lengths of 4096**
- Transformers cannot handle the entire document

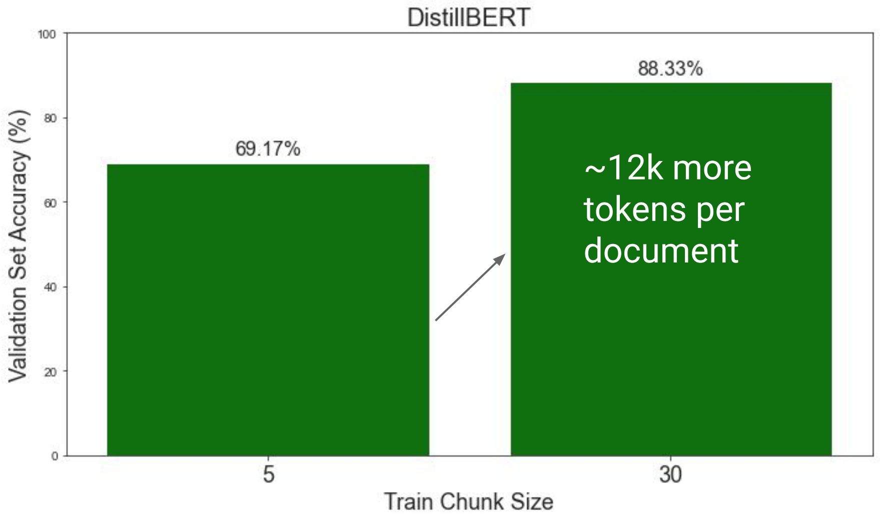


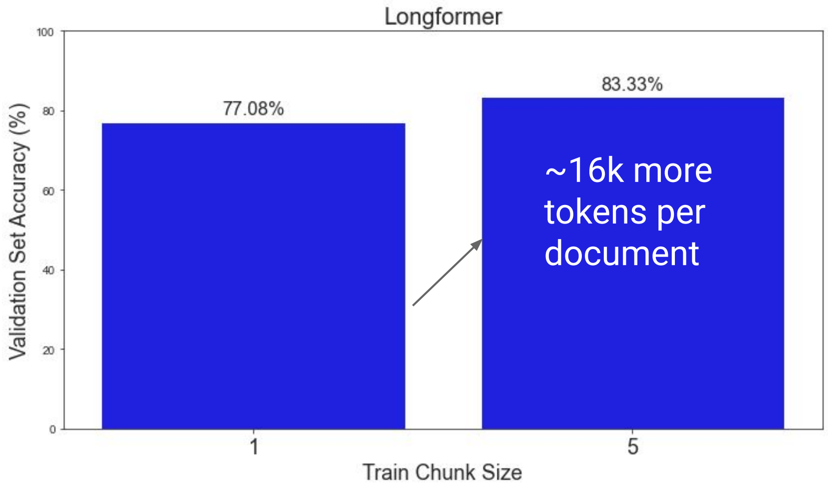
Hierarchical novel-level approach improved accuracy by 12%*



*accuracy improvement quoted for DistillBERT using 5 samples/document

DistillBERT outperformed Longformer with increased document coverage





DistillBERT Hyperparameter tuning

- Model benefits from aggressive updates. Author classification requires more adaptation from the pretrained model
- Significantly stronger regularization, common in author attribution tasks to prevent model from latching onto specific words.
- Also saw this in SVM preferring linear kernel over more complex ones.

Parameter	Comparison of optimal hyperparameter vs default
learning_rate	**~40% higher**
weight_decay	**8.6x higher**
dropout	**3x higher**
attention_dropout	**2x higher**
batch_size	**2-4x larger**

Comparison Across Models and Other Studies

Literature Review of text classification

- A study on **18-authors** of late 19th century novels yielded an **accuracy of 0.80 and F-1 of 0.82** using a **GAN-BERT** model[1].

- The ‘goldilocks hypothesis’ suggests that dataset size and diversity of the dataset determine which methods outperform[5]

- A text classification study showed **Linear SVM outperformed vs fine tuned transformer** for BBC & 20 NewsGroup datasets[2].

- A study using “all the news” dataset compared DistillBERT to ,RF,MLP. Results showed **DistillBERT underperformed vs RF and MLP** in **Article 3** Dataset, **Article 1** dataset (10 authors) but **outperformed** in **Article 2** dataset[3]

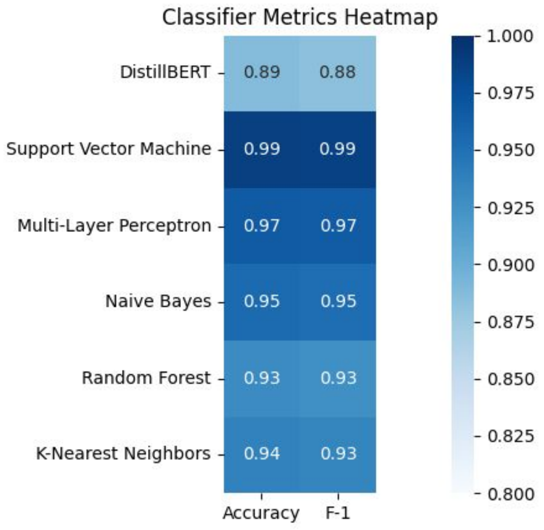
- There are however other studies that see fine tuned transformers outperforming traditional NLP!

Summary Metrics on Test Dataset

Model Name	Accuracy	F-1	Precision	Recall
DistillBERT	0.8875	0.8842	0.9181	0.8875
Support Vector Machine	0.9875	0.9871	0.9906	0.9875
Multi-Layer Perceptron	0.9667	0.9662	0.9763	0.9667
Naive Bayes	0.9542	0.9520	0.9700	0.9542
Random Forest	0.9292	0.9263	0.9442	0.9292
k -Nearest Neighbors	0.9375	0.9344	0.9565	0.9375

Results Discussion

- For this problem, lower cost NLP methods outperform fine tuned transformers
    - No Free Lunch Theorem
- SVM followed by MLP were the top 2 models
- Regularization is key
    - Linear SVM outperforms more complex kernels
    - 1-layer MLP outperformed MLP with multiple hidden layers significantly
    - Possibly why fine tuned transformers underperformed
- Caveat: possible that in wider parameter space (hyperparameters, epoch, test set, etc) different models “win”



Limitations & Future Directions

- Ensemble methods that combine TF-IDF features ability to capture global style, with fine tuned transformers that capture more nuanced patterns.
- Integrate document level hierarchical knowledge within training process for transformers.
- Try the other fine tuning approaches for transformers
- More compute for transformers to try even larger word coverage and a larger hyperparameter space
- Developing more efficient transformers for long documents is an active area of research.
- Use of Latent Dirichlet Allocation (LDA) along with TF-IDF to better differentiate writing styles and themes within texts.

Summary

Abstract

SPGC is a standardized dataset created from Project Gutenberg.

Dataset Creation & EDA

Text was cleaned and trimmed to 80 authors, each containing 30 books.

Traditional NLP Preprocessing

Cleaned text was tokenized and weighted via TF-IDF or transformer-specific tokenizer.

Traditional Classifiers

Baseline metrics generated using Naive Bayes, Random Forest, k-Nearest Neighbors & SVM.

Transformers

DistillBERT outperformed Longformer

Results comparison

SVM accuracy topped the transformer approach, while traditional classifiers also performed impressively well.

References

1. Ugolini, Laura and Mitkov, Ruslan (2023) Authorship attribution of late 19th century novels using GAN-BERT. In: 61st Annual Meeting of the Association for Computational Linguistics - Student Research Workshop, 10 July 2023 - 12 July 2023, Toronto, Canada

2. Wahba, Yasmen, et al. *A Comparison of SVM against Pre-Trained Language Models (PLMs) for Text Classification Tasks*. arXiv:2211.02563, arXiv, 4 Nov. 2022. *arXiv.org*, https://doi.org/10.48550/arXiv.2211.02563.

3. Abbasi, Ahmed, et al. “Authorship Identification Using Ensemble Learning.” *Scientific Reports*, vol. 12, no. 1, June 2022, p. 9537. *DOI.org (Crossref)*, https://doi.org/10.1038/s41598-022-13690-4.

4. Rennie, J. D. M., Shih, L., Teevan, J., & Karger, D. R. (2003). *Tackling the poor assumptions of naive Bayes text classifiers*. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington, DC. Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

5. Snyder, Scott H., et al. “The Goldilocks Paradigm: Comparing Classical Machine Learning, Large Language Models, and Few-Shot Learning for Drug Discovery Applications.” Communications Chemistry, vol. 7, no. 1, June 2024, pp. 1–11. www.nature.com, https://doi.org/10.1038/s42004-024-01220-4.

6. Daniel Jurafsky and James Martin. Speech and Language Processing. 2008.

Sources

[1]. A standardized Project Gutenberg corpus for statistical analysis of natural language and quantitative linguistics.
[2]. NLTK: The Natural Language Toolkit
[3]. A prototype gutenberg-hathitrust sentence-level parallel corpus for OCR error analysis: pilot investigations

Our code is available at:
https://github.com/DeanKW/gutenberg_corpus_analysis

Our fork of the Project Gutenberg Repository:
https://github.com/DeanKW/gutenberg

The original repository:
https://github.com/pgcorpus/gutenberg