

Challenges in Modeling Historical Language Style for Diachronic Style Recognition

Adrian Niedziółka-Domański

0009-0003-4797-7484

Maria Curie-Skłodowska University

Plac Marii Curie-Skłodowskiej 5

20-031 Lublin, Poland

Email: niedziolkadrian@gmail.com

Jarosław Bylina

0000-0002-0319-2525

Maria Curie-Skłodowska University

Plac Marii Curie-Skłodowskiej 5

20-031 Lublin, Poland

Email: jaroslaw.bylina@mail.umcs.pl

Abstract—This paper addresses the issue of modeling and recognizing historical linguistic styles. Particular attention is paid to the challenges associated with processing diachronic data, such as stylistic differences between eras, limited availability of historical data, and issues arising from the digitization of texts. As part of this work, a process was developed for constructing and cleaning a corpus consisting of texts in Early Modern English and contemporary English. Next, a SentencePiece BPE tokenizer adapted to historical language variants was developed. A discriminator model based on a Transformer Encoder with a FiLM conditioning mechanism was also proposed, designed to determine whether a text belongs to a specific linguistic period. The experiments conducted demonstrated high classification accuracy and a negligible impact of increasing the model size on the final prediction quality.

I. INTRODUCTION

THE study of language from a diachronic perspective, i.e., in terms of changes over time, is a key area of both historical linguistics and natural language processing (NLP). As language models evolve, there is growing interest in how these systems handle data from different eras and how they recognize and model variations in linguistic style. In this paper, we focus on the problem of modeling linguistic style using modern machine learning methods and identifying historical style.

A. Characteristics of Historical Language Variants

Historical variants of language differ from their modern forms in many ways: orthographically, morphologically, syntactically, and lexically. These changes result from both the naturally occurring evolution of language itself and from social, cultural, and technological factors [1], [2].

Historical texts often feature non-standard or uncoded spelling, a lack of uniform spelling rules, and greater variation in grammatical forms. Additionally, language from different periods may exhibit significant differences in vocabulary, syntactic structures, and stylistic conventions.

B. NLP Challenges for Diachronic Data

Processing diachronic data in NLP presents a number of challenges. One of the main issues is the lack of linguistic

consistency across corpora from different eras. These differences include vocabulary, syntax, punctuation, and stylistic conventions characteristic of a given historical period.

Another challenge is the limited availability of historical data and its inconsistent quality. Many historical texts come from digitized archives, which introduces OCR (Optical Character Recognition) errors, missing characters, non-standard formatting, and scanning artifacts. Additionally, historical corpora are typically much smaller than the modern datasets used to train language models.

Another significant challenge is the lack of large parallel datasets containing texts with the same meaning written in different historical variants of the language. Unlike traditional translation tasks, in the case of historical style transfer, there are rarely direct sentence equivalents across different historical periods. This complicates the training of supervised models and the objective evaluation.

Additionally, standard NLP models are typically optimized for modern linguistic data, which means they perform poorly with archaic language forms and historical spelling. The problem is further complicated by the fact that differences between eras are often not limited to vocabulary alone but also involve more subtle stylistic and syntactic features that are harder to model.

C. Aim of the study

The aim of this work was to investigate the problem of representing and recognizing historical style using models based on the Transformer architecture [3]. Particular attention was paid to:

- preparation of a diachronic corpus,
- challenges of tokenizing historical language variants,
- representation of linguistic style,
- use of stylistic conditioning, and
- analysis of the possibilities for automatically recognizing the historical period of a text.

The study developed:

- data processing pipeline for historical data,
- tokenizer adapted to historical texts, and
- style discriminator based on the Transformer Encoder.

The main contribution of this work is the development of a unified framework for diachronic style recognition that combines corpus construction, historical-aware tokenization, and Transformer-based style discrimination. Although the individual components are based on established methods, their integration addresses the specific challenges of historical language processing, including orthographic variability, OCR artifacts, and stylistic differences between language periods. Furthermore, the proposed discriminator is designed as a reusable component of a future adversarial framework for diachronic text style transfer.

The remainder of this paper is organized as follows. Section II describes the construction and preprocessing of the corpus. Section III presents the tokenizer adapted to historical texts. Section IV introduces the proposed discriminator architecture. Section V reports the experimental setup and results. Finally, Sections VI and VII present the conclusions and directions for future work.

II. DATASET CONSTRUCTION AND PREPROCESSING

This section describes the preparation of the diachronic corpus used in the experiments. It covers the data collection process, preprocessing and OCR correction procedures, sentence extraction, and the construction of the final dataset used for training and evaluation.

A. Data Collection

The data used in this study was collected from publicly available online sources containing both historical and modern texts. The main objective was to create a corpus that would enable the analysis of stylistic differences between different historical periods, in this experiments mainly from Early Modern English (EMO) and modern English of the XXI century.

For each document, basic metadata was recorded, including:

- file identifier,
- text title,
- approximate year of creation,
- language period, and
- source of the text.

The metadata was saved in a standardized text format, which enabled automatic processing of the corpus using various data-processing scripts.

Example entry describing the document is shown in Listing 1.

Listing 1 Example of the entry.

```
3007
The shoemakers' holiday
1599
Early Modern English
https://llds.ling-phil.ox.ac.uk/llds/xmlui/handle/20.500.14106/3007
```

The historical texts were sourced mainly from digital archives of English literature and repositories of historical

texts. The modern data consisted mostly of newspaper articles and informational texts from the XXI century.

Due to the nature of the historical data, the corpus contained texts of varying editorial quality, which required an additional data cleaning step.

B. Data Cleaning and OCR Correction

One of the most important stages in preparing the corpus was cleaning the historical data. Many of the texts came from digitized archives and scans of historical documents, which resulted in:

- OCR errors,
- missing characters,
- damaged text segments,
- non-standard symbols, and
- scanning artifacts.

For example, the original OCR output was:

Next to C••do in order by and by

After analyzing the source and manual correction, it was updated to:

Next to Cupydo in order by and by

In cases where it was impossible to correctly reconstruct the text or there were too many OCR errors, the sentences were removed from the corpus.

For modern data, the cleaning process was more restricted. Given the wide availability of modern texts, it was decided to remove fragments containing OCR errors or other damages rather than attempt to reconstruct them manually.

Special attention was paid to preserving historical characters and the original spelling of the texts. Spelling was not standardized to modern linguistic forms, as this could lead to the loss of important stylistic information characteristic for a given historical period.

C. Sentence Extraction and Dataset Construction

After cleaning the data, all text was split into sentences. The segmentation was performed based on basic punctuation marks, such as:

- .
- !
- ?

The sentences were then tokenized using a pre-trained tokenizer (more on this in section III). The maximum sequence length was set to 128 tokens. For sentences exceeding the length limit, an additional splitting step was performed. Due to the irregular syntax of historical texts, splitting was not performed automatically based solely on the number of tokens. Instead, a semi-automatic approach was used, allowing for interactive selection of the sentence split point, which helped minimize the loss of syntactic consistency.

After preparing the sentences, a common dataset was created containing:

- text, and
- label indicating the time period.

The modern and historical data were then balanced in terms of character count and text volume. Due to the significantly greater availability of modern data, some of the XXI century texts were removed to mitigate the problem of class imbalance during model training.

The final corpus consisted of 4 historical and 32 modern documents, yielding 31841 sentences after preprocessing. In total, the corpus contained approximately 600000 words. Table I summarizes the corpus statistics.

TABLE I
CORPUS STATISTICS

Split	Documents	Sentences	Words
EMO	4	16250	295394
XXI	32	15591	302259
Total	36	31841	597653

The final dataset was divided into:

- training set, and
- test set.

The test set constituted 20% of the data. To preserve the distribution of historical and modern texts, stratified sampling was applied using the `stratify` parameter based on the period label.

```
train_df, test_df = train_test_split(
    data,
    test_size=0.2,
    stratify=data["period"]
)
```

An example of the data structure is shown in Listing 2 (Tab-Separated Values – TSV file).

Listing 2 Example entries in a dataset.

```
text    period
Farewell good fellows of the Gentle trade,
↪    emo
Apurva Asrani retweeted Krish's interview...
↪    xxi
```

This corpus served as the basis for training the style discriminator, as well as for further experiments.

III. TOKENIZATION OF HISTORICAL TEXTS

The tokenization of historical texts presents a significant challenge in natural language processing tasks due to the highly non-standard nature of the text and its significant orthographic variability.

The analyzed data included, among other things:

- archaic characters, such as *f* (long *s*) or *p*,
- diacritical marks and their non-standard combinations (e.g., ligatures such as *ö*),
- lack of standardization in historical spelling, and
- high orthographic variability of the same lexeme.

As a result, classical tokenization approaches based on the modern Latin alphabet were often unable to correctly segment the text or led to the fragmentation of semantically consistent units.

A. Experiments with the Byte-Level BPE Tokenizer

In the first phase of the experiments, a Byte-Level Byte Pair Encoding (ByteLevelBPE) tokenizer from the Hugging Face tokenizers library [4] was used.

The model was trained with a vocabulary size of 32,000 tokens.

This approach is based on representing characters in byte space, which theoretically allows for the handling of any Unicode characters. In practice, however, significant limitations were observed in the context of historical data.

The most significant issues included:

- suboptimal segmentation of special and archaic characters,
- breaking rare characters into unintuitive sequences of sub-tokens,
- difficulties in maintaining consistency in the representation of characters such as *f*, *p*, and complex diacritics, and
- degradation of tokenization quality for non-standard spelling variants.

As a result, ByteLevelBPE did not provide a stable and semantically consistent representation of historical texts, particularly in the case of rare or archaic symbols.

B. SentencePiece Tokenizer Based on BPE

In the next stage, SentencePiece tokenizer [5] in BPE mode was used, which proved to be significantly more stable when applied to historical data.

The following configuration was used:

- dictionary size: 32,000 tokens,
- character coverage: 1.0,
- model mode: BPE,
- normalization: identity, and
- special tokens: `<mask>`, `<pad>`, `<sos>`, `<eos>`, `<unk>`.

A key decision was to use *identity* normalization (i.e., no text normalization), which allowed the original spelling of the texts to be preserved, including all special characters, archaic letters, and non-standard diacritics.

Additionally, setting `character_coverage = 1.0` ensured that the tokenizer accounts for all characters present in the corpus, eliminating the problem of unsupported symbols.

The tokenizer was trained using pre-cleaned text files that were merged into a single input corpus using a dedicated script (which parses the `info.txt` file containing metadata about the sources, as shown in Listing 1). The training setting is presented in Algorithm 1.

Using SentencePiece in BPE mode with normalization disabled and 100% character coverage proved to be the most suitable solution for the historical data. Unlike ByteLevelBPE, this tokenizer provided a stable representation of rare characters and complete control over how the text was segmented, which was crucial for further experiments.

IV. DISCRIMINATOR ARCHITECTURE

The discriminator in the proposed system is designed as a language style classification model. Its task is to determine

Algorithm 1 Training SentencePiece tokenizer.

```

spm.SentencePieceTrainer.Train(
    input=raw_files,
    model_prefix=str(save_dir / "tokenizer"),
    vocab_size=32000,
    character_coverage=1.0,
    model_type="bpe",
    normalization_rule_name="identity",
    pad_id=0,
    unk_id=1,
    bos_id=2,
    eos_id=3,
    user_defined_symbols("<mask>")
)

```

whether a given text sequence belongs to a specific linguistic period, that is, in the case of these experiments, Early Modern English (EMO) or modern English of the XXI century.

Unlike traditional approaches based solely on lexical features, the proposed architecture utilizes deep contextual representations based on the Transformer Encoder and style conditioning mechanisms. As a result, the model is capable of capturing both local and global characteristics specific to a given linguistic period.

A. Representation of style using period encoding

Style information is encoded using period encoding, which is a simple representation of a condition in which each example is assigned to one of two style classes (EMO/XXI). This class is then mapped to an embedding space, creating a vector that represents the linguistic style.

The style embedding is trained alongside the other model parameters and serves as a conditioning factor for the entire architecture.

B. FiLM conditioning

The key conditioning mechanism in the model is FiLM (Feature-wise Linear Modulation) [6], which allows for dynamic modification of hidden representations depending on the linguistic style. The FiLM mechanism uses two parameters:

- γ (scaling), and
- β (shift),

which are generated based on the style embedding.

Each hidden representation x of a token is transformed according to the following relationship:

$$x' = \gamma(c) * x + \beta(c),$$

where c denotes the style embedding.

This operation allows the model to adapt its internal representations depending on the selected linguistic style.

C. Style Discriminator

The discriminator was implemented as an encoder-only Transformer inspired by the BERT(Bidirectional Encoder Representations from Transformers) architecture [7].

The proposed architecture combines an encoder-only Transformer with FiLM for conditioning on linguistic period information. This design is motivated by the need to jointly model semantic content and stylistic information in a controlled manner.

Transformer encoders are well-suited for capturing contextual dependencies in text, making them a natural choice for modeling diachronic language variation. However, standard Transformer architectures do not provide an explicit mechanism for separating content representation from external conditioning variables such as linguistic period.

To address this limitation, FiLM is employed as a conditioning mechanism that modulates intermediate hidden representations using period-dependent scaling and shifting parameters.

The model processes the entire input sequence in parallel, using a self-attention mechanism to model contextual dependencies between tokens. The architecture includes:

- token embedding layer,
- several Transformer encoder layers,
- FiLM layer introducing style conditioning,
- representation aggregation mechanism (mean pooling), and
- classification layer.

The overall architecture is illustrated in Figure 1.

After the sequence is processed by the main network, the token representations are aggregated using mean pooling, which yields a global representation of the sentence. This representation is then fed to the classification layer, which predicts whether a given sequence belongs to a specific language style or not.

D. Dataset for style recognition

To train the discriminator, a specially constructed dataset was prepared to test the model's ability to distinguish between linguistic styles.

The discriminator was trained using the corpus derived from the one described in Section II, which contains sentences assigned to two linguistic periods: Early Modern English and modern English of the XXI century.

Based on this dataset, a version adapted for the task of verifying stylistic consistency was created.

In the original corpus, each example contained text and a correct label for the linguistic period. Each example was initially treated as correct (label = 1), meaning the text matched the assigned period. Negative examples were then generated by intentionally disrupting this consistency. For each period, 50% of the samples were randomly selected, and the assigned linguistic period was switched to the opposite (EMO $\leftrightarrow$ XXI) while leaving the text unchanged.

As a result, a dataset was created in which:

- $label = 1$ indicates that the text matches the assigned period, and
- $label = 0$ indicates a mismatch between the text and the assigned style.

A dataset structured in this way allows the model to learn the task of judging the consistency between the text and the

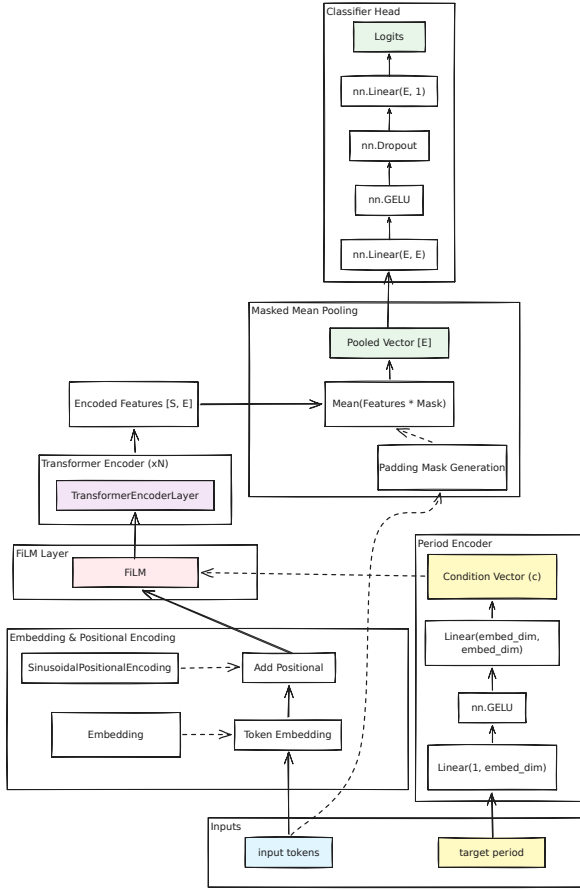


Fig. 1. Discriminator architecture.

assigned stylistic period. An example of the data structure is shown in Listing 3.

Listing 3 Example entries from the style recognition dataset.

text	period	label
Farewell good fellows of the Gentle trade...		
↪ emo	1	
Apurva Asrani retweeted Krish's interview...		
↪ emo	0	
New York's American Museum of Natural History		
↪ has an...	xxi	1

V. EXPERIMENTS

The aim of the experiments was to evaluate the effectiveness of the proposed discriminator in verifying linguistic style consistency and its ability to distinguish between texts from different linguistic periods.

A. Baseline model

As a reference for evaluating the proposed architecture, a classical machine learning baseline was implemented

using Term Frequency–Inverse Document Frequency (TF-IDF) [8] features and a Linear Support Vector Classifier (LinearSVC) [9].

The baseline was trained and evaluated using the same train/test split as the proposed Transformer-based discriminator.

Since the task consists of verifying whether a given sentence is consistent with an assigned linguistic period, the period information is explicitly included in the input representation. In the baseline setting, this is achieved by prepending a special token in the form [PERIOD=EMO] or [PERIOD=XXI] to each sentence prior to feature extraction. This ensures that both the textual content and the assigned period are available to the classifier while preserving a consistent problem formulation across all models.

The input text is transformed into numerical representations using TF-IDF with unigram and bigram features. The resulting sparse feature vectors are classified using a Linear Support Vector Classifier, which learns a linear decision boundary in a high-dimensional feature space. Due to its robustness and strong performance on sparse textual representations, LinearSVC is a standard and widely used baseline in text classification tasks.

B. Experiment Settings

The experiments were conducted using a model based on an encoder-only Transformer architecture with style conditioning via the FiLM mechanism described in Section IV.

Two model size variants were tested:

- Larger Model:
 - 4 Transformer encoder layers
 - 256 embedding dimensions
- Smaller Model:
 - 3 Transformer encoder layers
 - 128 embedding dimensions

The model was trained using the Binary Cross Entropy (BCE) loss function, which is suitable for the binary classification problem of determining whether a text matches a given linguistic style.

For parameter optimization, the Adam algorithm was used, which allows for stable and fast convergence in Transformer-based models. Training was performed with a fixed batch size of 32, selected experimentally as a compromise between training stability and memory constraints.

C. Results

This section presents the results of the proposed language style discriminator for both architectural variants.

To provide a reference point for evaluating the proposed Transformer-based discriminator, a classical TF-IDF + LinearSVC baseline was implemented. The baseline achieves an accuracy of 0.6747 on the test set. Comparison between models is shown in Table II.

The figures show the loss function and accuracy curves during training and testing.

Figure 2 presents the loss function, and Figure 3 shows accuracy during training. Although the smaller model achieved a slightly lower loss value during training, their accuracies are similar. Both models show stable convergence, as can be seen in these graphs.

In the initial phase of training, a rapid decrease in the loss value and a simultaneous increase in accuracy are observed, indicating effective learning of language style representations.

After a few epochs, the learning process stabilizes, and further changes in the loss function values are minor, suggesting that a state close to convergence has been reached.

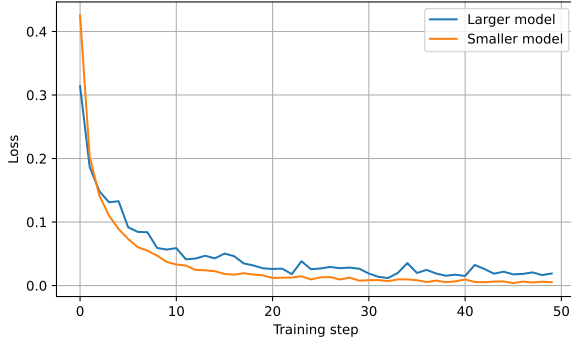


Fig. 2. Train loss.

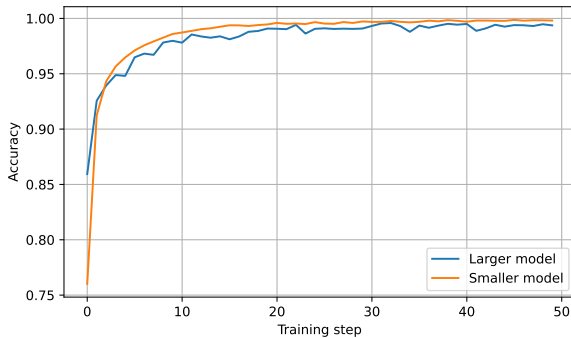


Fig. 3. Train accuracy.

The results on the test set, shown in Fig. 4 and Fig. 5, indicate that both models generalize well to unseen data.

The results indicate that both models generalize similarly in terms of accuracy. Differences in loss suggest differences in prediction calibration, i.e., the larger model produces more confident (or better-calibrated) predictions, even though this does not translate into improved classification accuracy. This means that increasing the model's complexity does not lead to a significant improvement in quality in terms of accuracy, and the potential gains relate only to the probability distribution.

Table II compares the two model variants. The larger model achieves a slightly better accuracy score (0.9708 vs. 0.9697) and a lower BCE loss value (0.0962 vs. 0.1366). However, these differences are marginal.

The smaller model, despite its simplified architecture, achieves comparable performance. The results suggest that the

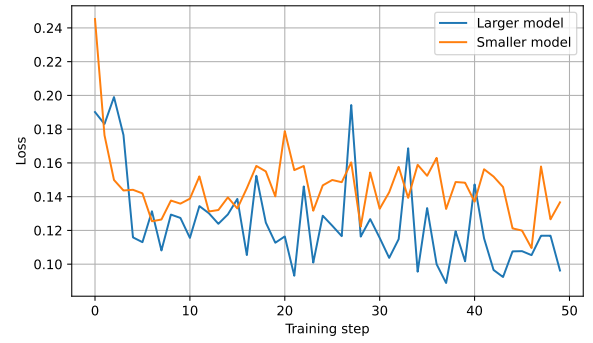


Fig. 4. Test loss.

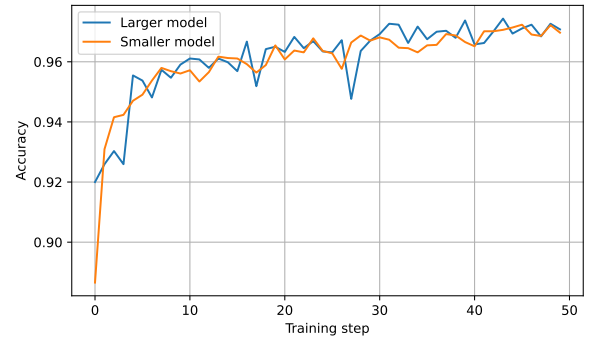


Fig. 5. Test accuracy.

task has reached a saturation point where further increases in model complexity do not translate into significant improvements in prediction quality.

The substantial performance gap between the TF-IDF baseline and the Transformer-based models demonstrates the advantage of contextual representation learning for the proposed task. While the baseline relies on sparse lexical features, the Transformer architecture is able to capture deeper semantic and stylistic patterns, resulting in a significant improvement in classification accuracy.

TABLE II
COMPARISON OF MODELS

Model	Layers	Embed dim	Test Accuracy	Test Loss
Baseline	-	-	0.6747	-
Larger	4	256	0.9708	0.0962
Smaller	3	128	0.9697	0.1366

VI. CONCLUSION

This paper proposed a linguistic style discriminator model based on the Transformer Encoder architecture with the FiLM conditioning mechanism. The model was designed to classify texts according to their affiliation with two linguistic periods: Early Modern English and modern English of the XXI century.

The experiments conducted showed that both the larger and smaller variants of the model achieve very similar classification performance. Differences in accuracy are marginal, while

the larger model achieves a lower loss function value, which may indicate better prediction calibration.

The results suggest that the problem of historical style recognition can be effectively solved using relatively small models, and that increasing the complexity of the architecture does not lead to significant improvements in classification accuracy. This indicates the presence of a saturation effect, in which further increases in model capacity do not translate into noticeable improvements in performance. At the same time, the results suggest that the preparation of diachronic data, including corpus construction and the selection of tokenization and representation, is a crucial element of the entire pipeline.

VII. FUTURE WORK

Future research plans to use the proposed discriminator as a component of a larger generative architecture for language style transfer.

In particular, the model will be integrated into a framework based on an adversarial approach, inspired by the CycleGAN architecture, where the discriminator will evaluate the consistency of the generated text with the target linguistic style [10]. This will allow for a more controlled and structurally consistent transformation of text across different linguistic periods.

Additionally, the research can be expanded to include:

- more diverse historical periods,
- larger and more balanced corpora,
- analysis of the impact of tokenization on the quality of style transfer.

GENERATIVE AI DISCLOSURE STATEMENT

During the preparation of this work the author(s) used ChatGPT in order to improve the language and clarity of the manuscript. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

REFERENCES

- [1] R. W. Murray, "Historical linguistics: the study of language change," *Contemporary linguistics. An introduction*, pp. 313–371, 1996. [Online]. Available: <http://www.pearsoncanada.ca/media/highered-showcase/multi-product-showcase/ogrady-ch08.pdf>

- [2] S. Kirby, M. Dowman, and T. L. Griffiths, "Innateness and culture in the evolution of language," *Proceedings of the National Academy of Sciences*, vol. 104, no. 12, pp. 5241–5245, 2007. doi: 10.1073/pnas.0608222104. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.0608222104>. [Online]. Available: <https://www.pnas.org/doi/abs/10.1073/pnas.0608222104>
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. doi: 10.48550/arXiv.1706.03762. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
- [4] H. Face, "Tokenizers: Fast state-of-the-art tokenizers optimized for research and production," 2020. [Online]. Available: <https://github.com/huggingface/tokenizers>
- [5] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," arXiv, Aug. 2018. doi: 10.48550/arXiv.1808.06226. ArXiv:1808.06226 [cs.CL]. [Online]. Available: <http://arxiv.org/abs/1808.06226>
- [6] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. doi: 10.1609/aaai.v32i1.11671. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/11671>
- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019. doi: 10.18653/v1/N19-1423. pp. 4171–4186. [Online]. Available: <https://aclanthology.org/N19-1423/>
- [8] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, Jan. 1988. doi: 10.1016/0306-4573(88)90021-0. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0306457388900210>
- [9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995. doi: 10.1007/BF00994018. [Online]. Available: <https://doi.org/10.1007/BF00994018>
- [10] A. Niedziółka-Domański and J. Bylina, "Adapting CycleGAN architecture for Unpaired Diachronic Text Style Transfer," Kraków, Poland, Oct. 2025. doi: 10.15439/2025F4661. pp. 81–86. [Online]. Available: https://annals-csis.org/Volume_44/drp/4661.html