ASSIGNMENT : PREDICTING METER

Jan Grielens
UAntwerpen
Master Digital Text Analysis
Machine Learning
2023-2024

ABSTRACT

This paper investigates the application of machine learning methods, comparing neural networks and traditional approaches, to predict the meter in poetry. Analyzing rhythm patterns in verses, it explores both rule-based and neural network models for classifying poetic meters. The study demonstrates the feasibility of using rhythmic encoded patterns for meter prediction and suggests potential for future research in direct meter detection from spoken verses

1 Introduction

In language, music, and poetry, rhythmic features play a crucial role in understanding. This area has been under investigation since Logan's work in 1988 [1]. Historically, poets adhered to traditional and strict formats. However, in today's postmodern era, there is a trend towards exploring novel ways to deviate from these conventions. The learning, deciphering, and identification of these patterns, including meter and foot, remain integral to the study and practice of poetry. Recent developments in natural language processing (NLP) have led to research exploring new NLP approaches. The transition from spoken sound to written word often results in the loss of rhythmic information. A sentence can be viewed as a combination of phonetic syllables, stressed or unstressed. A 'foot' comprises two or three parts, stressed (s) or unstressed (w), leading to 12 possible combinations, known as 'feet', which include 4 bigram and 8 trigram combinations. Traditionally, English poetry utilizes seven of these combinations.

Table 1: Seven Poetic Feet and Their Stresses

Category	Poetic Feet and Stresses						
Names	Iamb	Trochee	Dactyl	Anapest	Pyrrhic	Amphibrach	Spondee
Stresses	ws	sw	sww	wws	ww	wsw	ss

Meters then are described as a repetition, from one to eight, of a specific feet. If the foot is repeated once, then the verse is monometer, twice then it is a dimeter verse, until the octameter, eight repetitions of the same foot.

We will investigate both neural network approach and traditional machine learning approaches to their feasibility in this context.

2 Related research

Until recently, computational approaches to verse analysis were predominantly rule-based, relying on phonetic dictionaries and a set of rules for phonological parsing and generating metrical patterns. Prosodic[1], based on the CMU pronunciation dictionary, also utilizes the Text-to-Speech engine 'espeak' for voicing unknown words to identify

[1]`https://github.com/quadrismegistus/prosodic`

their phonetic signatures. Extensions of Prosodic, such as Pronouncing[2], Erato[3], and Poesy[4], also employ the CMU pronunciation dictionary. In contrast, Rantanplan [2], initially developed for Spanish, utilizes SpaCy. ZeuScansion [3] employs finite-state technology for metrical scansion, while Metricalizer is a rule-based tool for metrical annotation in German.[5]

Agirrezabal [4] employed LSTM methods to predict the metrical patterns in lines of verse, achieving a per-line accuracy of 61.39% for English. Yousef et al.[5] adopted a distinct approach by using featureless, character-based inputs in different RNN networks, reporting an overall accuracy of 82.31% for English. The methodology of Transformers was also explored by de la Rosa et al. [6], which provided a baseline for both rule-based and neural network approaches [7].

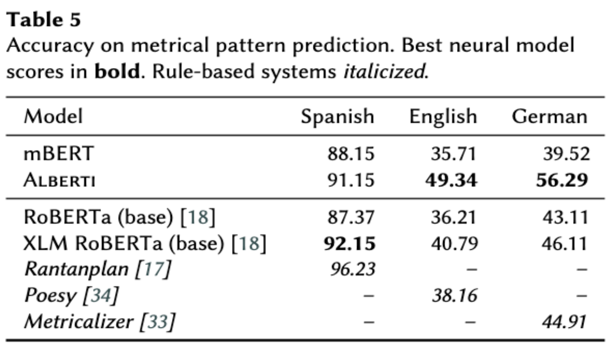


Table 5
Accuracy on metrical pattern prediction. Best neural model scores in **bold**. Rule-based systems *italicized*.

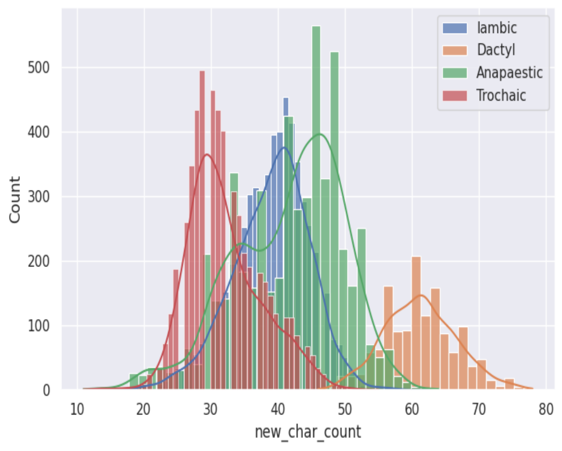
Model	Spanish	English	German
mBERT	88.15	35.71	39.52
Alberti	91.15	**49.34**	**56.29**
RoBERTa (base) [18]	87.37	36.21	43.11
XLM RoBERTa (base) [18]	**92.15**	40.79	46.11
Rantanplan [17]	*96.23*	–	–
Poesy [34]	–	*38.16*	–
Metricalizer [33]	–	–	*44.91*

Figure 1: Baseline for Accuracy of rule-based and alternative methods

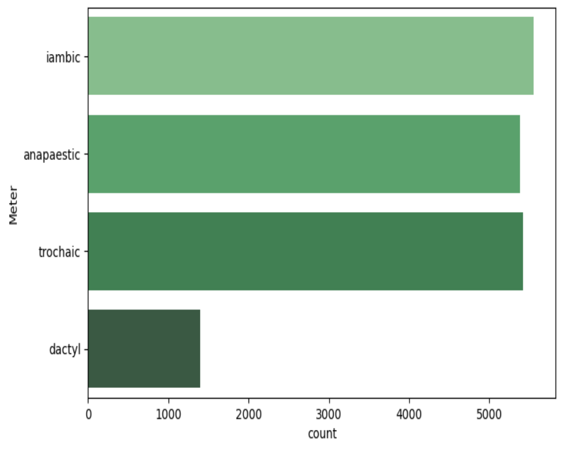
3 Experimental design

3.1 Data

The English dataset comprises 199,002 verses. Each verse is categorized into one of four meters: Iambic, Trochee, Dactyl, and Anapaestic. The dataset is predominantly Iambic, with 186,809 Iambic verses, followed by 5,418 Trochee, 5,378 Anapaestic, and 1,397 Dactyl verses \cite{PCD2018}. For our analysis, we utilized the Down-sampled English PCD, which includes 17,743 verses.



(a) Char count per Meter

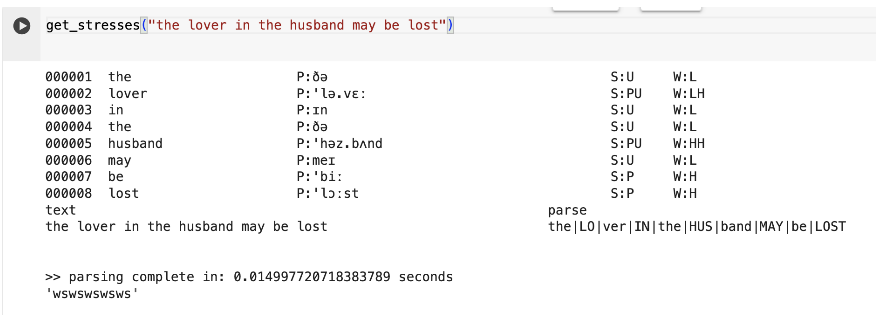


(b) Distribution per Meter

[2]https://github.com/aparrish/pronouncingpy
[3]https://github.com/manexagirrezabal/erato
[4]https://github.com/quadrismegistus/poesy
[5]https://www.metricalizer.de/en/metrikanalyse/

3.2 Tokenizing data

We experimented with various code bases (Prosodic, Pronouncing, Scansion, and Erato) to translate verses into stress patterns (strings). Owing to its ease of use and comprehensive documentation, we selected Prosodic for this task. Each verse in the dataset was parsed into a string of 'w's and 's's, representing the stress patterns. Subsequently, the dataset underwent further preprocessing and was saved to our local system. The dataset was then further preprocessed and



```
get_stresses("the lover in the husband may be lost")

000001  the                  P:ðə                                   S:U     W:L
000002  lover                P:'lə.vɛː                              S:PU    W:LH
000003  in                   P:ɪn                                   S:U     W:L
000004  the                  P:ðə                                   S:U     W:L
000005  husband              P:'həz.bʌnd                            S:PU    W:HH
000006  may                  P:meɪ                                  S:U     W:L
000007  be                   P:'biː                                 S:P     W:H
000008  lost                 P:'lɔːst                               S:P     W:H
text                                                        parse
the lover in the husband may be lost                        the|LO|ver|IN|the|HUS|band|MAY|be|LOST


>> parsing complete in: 0.014997720718383789 seconds
'wswswswsws'
```

Figure 3: Calculated stress pattern by Prosodic

saved to our local system.

3.3 Methods and experiments

We aimed to evaluate the feasibility of two methods for predicting one of four meters from a string, such as 'wswswswsw'. These methods align with the bigram-trigram approach traditionally used in English meter analysis. The preprocessed data was divided into a 60/20/20 split for training, validation, and testing.

The first model is XGBoost, a tree-based algorithm. It employs a level-wise strategy, scanning across gradient values and using these sums to evaluate the quality of splits at every potential point in the training set. For our classification task, we used `mlogloss`, a metric designed for multiclass classification problems and comparable to the categorical crossentropy loss. We adapted the dataset to meet the specific requirements of the XGBClassifier and applied a character-based CountVectorizer. A grid search was conducted with parameters: 'ngram range': [(1, 1), (2, 2), (1, 2), (**2, 3**)], 'depth': [5, **7**], 'estimators': [100, **200**], and 'learning rate': [0.01, **0.1**], with the best parameters in bold. We produced a confusion matrix, a classification-precision report, and calculated the Mean Squared Error (MSE) and Mean Absolute Error (MAE). The tree structure utilized 10 features ['ss', 'ssw', '**sw**', 'sws', '**sww**', '**ws**', '**wss**', 'wsw', 'ww', 'wws'], with the bolded features corresponding to our four meters.

The second model is a convolution model applied to text by Kim. [8]

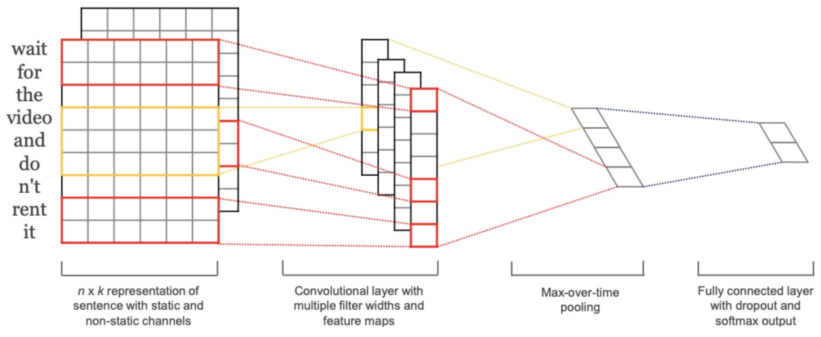

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 4: Convolution Neural Network for Sentence Classification

We adjusted this approach to one string of 'wswssswsws', padded to the maximum length because the convolution needs fixed length vectors. Three convolution layers were used with respective 1, 2 and 3 size kernels. Then a GlobalMaxPooling layer and a Dense layer with SoftMax. The `CategoricalCrossentropy` was used as a loss function applicable for our multiclass classification problems. This function heavily penalizes discrepancies between true labels (one-hot encoded) and predicted probabilities, applying milder penalties for correct predictions. For optimization, we chose Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and momentum of 0.9. The training was conducted over 10 epochs.

3.4 System

Jupyter Notebook in Google Colab with a V100 runtime. Coding was done with the assistance of ChatGPT 4.0, used for debugging, explaining code and code generation.

4 Results

Our main focus was determining the feasibility of our approach. The XGBClassifier performed with 76.00 accuracy better than the 1DConvolution (50.09) and second best in comparison with the rest. The XGBClassifier model has f1-score 5bbetween 0.72 and 0.78 with 0.98 for the Dactyl meter (sww). The 1DConvolution 6b has a low f1-score (0.22) for the Trochaic meter (sw).

Comparison with Baseline	
Model	**Accuracy %**
Youssef: RNN	82.31
Model 1: XGBClass	**76.00**
Agirrezabal: LSTM	61.39
Model 2: conv1D	**50.09**
ALBERTI: Transformer	49.34
Poesy: Rule-based	38.16

Table 2: Baseline comparison ranked

5 Conclusion

The current results show that on it is feasible to predict meter of verses on the basis of a rhythmic encoded pattern. We used a hybrid approach using existing methods. The 40-80 % accuracy indicates that there is room for more research. A interesting avenue for research could be to go from spoken verses directly to meter detection.

References

[1] Harry M. Logan. Computer analysis of sound and meter in poetry. *College Literature*, 15(1):19–24, 1988.

[2] Javier de la Rosa, Álvaro Pérez, Laura Hernández, Salvador Ros, and Elena González-Blanco. Rantanplan, fast and accurate syllabification and scansion of spanish poetry. *Proces. del Leng. Natural*, 65:83–90, 2020.

[3] Manex Agirrezabal, Bertol Arrieta, Aitzol Astigarraga, and Mans Hulden. Zeuscansion: a tool for scansion of english poetry. 2013.

[4] Manex Agirrezabal, Iñaki Alegria, and Mans Hulden. A comparison of feature-based and neural scansion of poetry. *CoRR*, abs/1711.00938, 2017.

[5] Waleed A. Yousef, Omar M. Ibrahime, Taha M. Madbouly, and Moustafa A. Mahmoud. Learning meters of arabic and english poems with recurrent neural networks: a step forward for language understanding and synthesis. *arXiv preprint arXiv:1905.05700*, 2019.

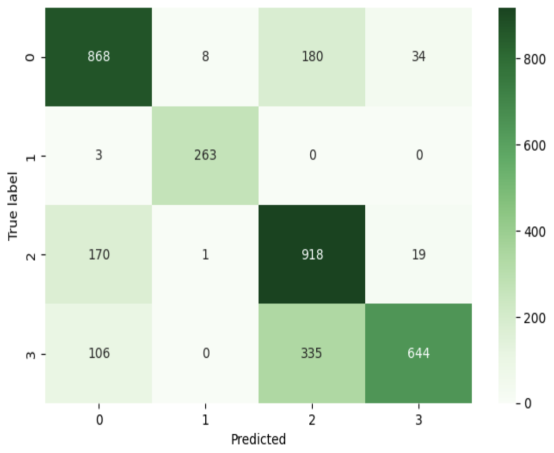
[6] Javier De la Rosa, Álvaro Pérez, Mirella De Sisto, Laura Hernández Lorenzo, Aitor Diaz, Salvador Ros, and Elena González-Blanco. Transformers analyzing poetry. multilingual metrical pattern prediction with transfomer-based language models. *Neural Computing and Applications*, 2021.

[7] Javier de la Rosa, Álvaro Pérez, Salvador Ros, and Elena González-Blanco. Alberti, a multilingual domain specific language model for poetry analysis. 2023.

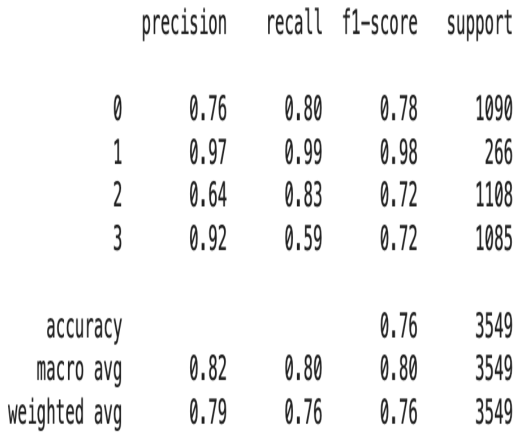
[8] Yoon Kim. Convolutional neural networks for sentence classification. 2014.

*Addendum

Mode 1: XGBClassifier



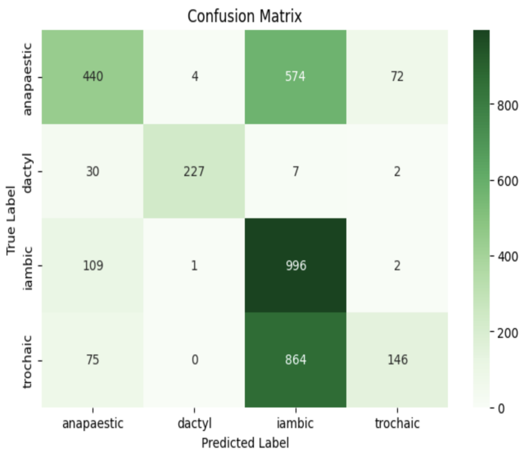
(a) Confusionmatrix



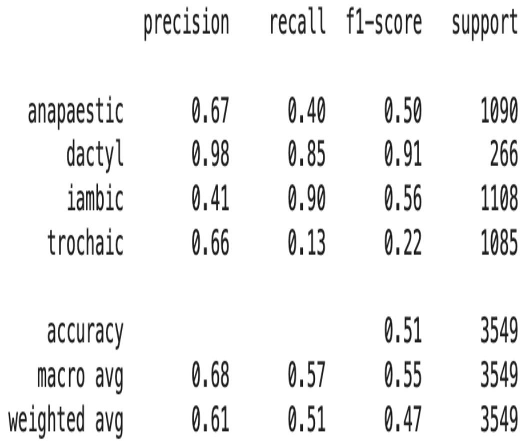
```
              precision    recall  f1-score   support

           0       0.76      0.80      0.78      1090
           1       0.97      0.99      0.98       266
           2       0.64      0.83      0.72      1108
           3       0.92      0.59      0.72      1085

    accuracy                           0.76      3549
   macro avg       0.82      0.80      0.80      3549
weighted avg       0.79      0.76      0.76      3549
```

(b) Classification report

Model 2: 1DConvolution



(a) Confusionmatrix



```
              precision    recall  f1-score   support

  anapaestic       0.67      0.40      0.50      1090
      dactyl       0.98      0.85      0.91       266
      iambic       0.41      0.90      0.56      1108
    trochaic       0.66      0.13      0.22      1085

    accuracy                           0.51      3549
   macro avg       0.68      0.57      0.55      3549
weighted avg       0.61      0.51      0.47      3549
```

(b) Classification report