Contents

[Abstract 2](#_Toc184345601)

[Introduction 2](#_Toc184345602)

[Literature Review 2](#_Toc184345603)

[Methodology 4](#_Toc184345604)

[Data 4](#_Toc184345605)

[Data Preprocessing 6](#_Toc184345606)

[Loading the dataset 6](#_Toc184345607)

[Extract target variable 6](#_Toc184345608)

[Data cleaning 6](#_Toc184345609)

[LSTM 6](#_Toc184345610)

[Text preprocessing 6](#_Toc184345611)

[Cleaning and Tokenizing text 6](#_Toc184345612)

[Feature Extraction 7](#_Toc184345613)

[Feature Matrix 7](#_Toc184345614)

[Model Architecture 7](#_Toc184345615)

[BERT 7](#_Toc184345616)

[Preprocessing 7](#_Toc184345617)

[Tokenization 8](#_Toc184345618)

[Model Architecture 8](#_Toc184345619)

[Model Training 8](#_Toc184345620)

[Results and Discussion 8](#_Toc184345621)

[LSTM Model Performance 9](#_Toc184345622)

[BERT Model Performance 9](#_Toc184345623)

[Comparison of Models 9](#_Toc184345624)

[References 11](#_Toc184345625)

Intelligent Prototype Development Evaluative Report

Topic: NLP for automated essay grading

# Abstract

Automated Essay Scoring (AES) is a service or software that predictively grades essays on the basis of a pre-trained computational model. Over the years, AES has captured increasing attention in the educational sector as it substantially lowers the manual labour of scoring or grading essays as close to humans’ decisions. From observation, it has ben discovered that Natural Language Processing (NLP) built off machine learning is especially satisfactory for AES and text classification. This report will evaluate two deep learning models which are the LSTM with Word2Vec embeddings and the BERT (Bidirectional Encoder Representations from Transformers) transformer model. These models will be criticised on their performance, effectiveness and accurate prediction of essays using Quadratic Kappa (QWK), Mean Squared Error (MSE), Mean Absolute Error (MAE) and R-squared (R²) evaluation metrics.

# Introduction

Recent progress in natural language processing (NLP) coupled with the advancements in machine learning (ML) algorithms have paved the way to new overall approaches in the sector of education and notably in the measurement of student performance. Automated Essay Scoring (AES) is an independent computer software or distributed services that assesses a written prose (Steffen Brandt, 2021).

Project Essay Grader (PEG) developed by Ellis Page in the early 1960s is considered as the first AES (Page, 1966). It puts emphasis on evaluating essays according to its writing style by employing “trins” and “proxes”. It follows the assumption that intrinsic characteristics exist in a person’s writing style which is referred to as trins. These trins can be measured or interconnected with observable elements illustrated as proxes. For example, fluency of an essay(trin) can be correlated to the amount of vocabulary (proxe). This system uses a sample set of 100 to 400 and facilitates statistical regression analysis for essay scores estimation. Up until 2020, PEG has developed trins exceeding 500 to be used on essays. On the other hand, Pearson Knowledge Technologies (PKT) introduced Intelligent Essay Assessor (IEA) (Foltz et al., 1999; Pearson, 2010). This model was developed to assess the quality of essays using a computational distribution model called the Latent Semantic Analysis (LSA) which examine the semantics similarity of texts (Landauer et al., 1998). LSA operates on domain -specific corpus where the essays are represented through the multidimensional semantic space of the meaning of their contained words and the similarity is obtained by comparison with other essays semantic representation. This model is particularly distinguished from other AESs due to scores being derived being in close proximity with human graders other than derivation of scores by correlation of essay features. IntelliMetric was presented by Vintage Learning in 1998 as a proprietary AES (Vantage Learning, 2020). It was considered as the first AES system that leveraged artificial intelligence (AI), and machine learning (ML) simulate the grading process (Dikli, 2006; Hussein et al., 2019). IntelliMetric uses over 400 elements (e.g. semantics, syntactics and discourses) characterised into five groups namely focus and unity(coherence), organization, development and elaboration, sentence structure, mechanics and conventions for the scoring system. IntelliMetric is said to have various active automated scoring systems, each utilizing a different mathematical model (e.g., Linear Analysis, Bayesian and LSA) for essay scoring. Lastly, E-rater is developed and used by the Educational Testing Service (ETS) since 1999(Attali & Burstein, 2006). This model depends on patented NLP techniques for the extraction of grammatical elements like grammatical errors and words usage errors for assessing style and content of an essay (Lim et al., 2021).

The aim of automated essay scoring is to accurately generate scores using machine learning automatically instead of manual human or with reduced involvement of human rates scoring vanquishing the time, cost and reliability concerns (Lim et al., 2021). With the of expanding high volumes of student submissions, manual grading is inefficient and immensely error prone. In automating the process, educators can focus on student feedback and curriculum improvement. AES is not aimed to substitute human assessors but to be employed for assistance by teachers or as an additional rater for cross-examination.

LSTM and BERT embody two robust approaches in NLP, each offering unique strengths in understanding textual data. I will explore LSTM with Word2Vec embeddings for sequential essay understanding as well as BERT for capitalizing on contextual embeddings and transformer-based architectures. The objective of this study is to develop an automated essay scoring model using LSTM with word2vec embeddings, implement and fine-tune a BERT model for essay scoring and lastly comparison of the models' prediction performance to establish the optimal and ideal approach for AES.

# Literature Review

Recurrent Neural Nets (RNN) are a fairly old technique in regard to neural network frameworks. Two of the predominant types of RNNS are the gated recurrent units (GRU) and the long short-term memory (LSTM). These models hold substantial responsible for cutting-edge results in speech recognition while also dominating the field of NLP until the emergence of transformer-based models (Steffen Brandt, 2021).

The LSTM permits for the incorporation of longer text sequences to be modelled by integrating a memory cell that allows for relatively smooth passing of information in a sequence from one point to another. Therefore, the LSTM enables the training of language models with contextual embeddings, where a tokens position in a sequence would make a difference for its meaning. This sequential structure however come at a drawback as training is generally very slow. In addition, the LSTM are negatively affected by the “vanishing gradient problem” which outlines an optimization problem that conventionally arises in large neural nets consequently, for this reason, making training of LSTMs impractical for long sequences (Steffen Brandt, 2021).

Principally, transformer models were initiated in the context of language translation and are based on the attention mechanism which was introduced in 2014. The attention mechanism allows the model to directly search for applicable tokens in the source text during prediction of the next token for the target text. It eliminates the bottleneck of encoding a text from the source language into a vector representation and then decoding the vector representation into text of the target language. Transformer models fall under three basic architectures: encoder models, decoder models, and encoder- decoder models. A key advantage of the transformer framework is that it provides parallel processing of input data and leaving it unaffected by the vanishing gradient issue making it achievable to train larger datasets for more efficient language models. However, additional performance capabilities are costly and tedious. The encoder model involves transforming only the input sequence (or left-hand side) of transformer architecture into a numerical representation. The BERT (Bidirectional Encoder Representations from Transformer) extracts semantics from a sentence from both directions. This transformer model is categorized as an encoder model and is distinctively used for text classification or extractive question answering making it ideal for AES (Firoozi et al., 2023).

In the field of computational text analysis, each piece of text is meant to be represented by a numerical vector in a vector area. Every text vector comprises of various dimensions containing numerical values (i.e. weights) allocated to each word as the fundamental features in the text representation. In text vector representation, weights are to be computed. The various text representation techniques include one-hot-coding, frequency-base, word embedding and contextual embedding. Mikitov et al introduced the Word2vec word embedding technique (Firoozi et al., 2023).

Word embedding techniques are unsupervised learning algorithms meaningfully convert a corpus of text into high-dimensional vectors such that semantically appropriate words are clustered together. There two versions of Word2vec are Continuous Bag of Words (CBOW) and Skip-Gram. With the use of the skip-gram version, the model learns by predicting the probability of the context of a word—gram neighbouring words-based on a set of words. Occurrence of words are captured a single pane at a time. Word2Vec produces the word vectors by feeding the text corpus into one learning model (Firoozi et al., 2023).

To fairly evaluate the AES systems are, the same evaluation metrics should be used. These are (1) quadrated weighted kappa (QWK), Mean Absolute Error (MAE) and Mean Square Error (MSE) (Ramesh, Sanampudi, and The Author(s), 2022).

There are limited studies done on the Skip-Gram approach as many machine learning models for AES still depend on the continuous bag of words (CBOW) approach (Lim et al., 2021). Despite the many benefits of word embedding techniques, there is still uncertainty on how fine-tune word embedding techniques with neural networks can greatly enhance the accuracy of AES models. For this reason, we will examine how a pre-trained, with fine-tuned hyperparameters Word2Vec embedding impacts the accuracy of the LSTM model (the effect of). In addition to that, there is little literature eon the comparison between sequential models like LSTM and transformer-based models like BERT for AES.

# Methodology

## Data

To investigate the use of the pre-trained Word2Vec text embedding technique within the LSTM model and compare with the transformer-based BERT model I used the “Automated Essay Scoring Dataset “dataset from 2019 found on Kaggle. This dataset includes a collection of essays scored across various prompts. Training, test and validation sets are available. The training set includes 8 varying essay sets

The first set has 1,785 level 8 essays with a final evaluation set size oof 592. The average length of 350 words. The resolved score range is between 2 and 12 which is computed from adding the first and second score which range from 1 to 6.

The second set has 1,800 level 10 essays with a final evaluation set size of 600. The average length of 350 words. Its scores are complied using a trait rubric. Scoring is computed from the resolved score of two domains. Two separate predictions are made for each domain.

Domain 1 (Writing Applications) Rubric range: 1-6

Domain 2 (Language Conventions) Rubric range: 1-4

Domain 1 (Writing Applications) Final score range: 1-6

Domain 2 (Language Conventions) Final score range: 1-4

The third set has 1,726 level 10 essays with a final evaluation set size oof 575. The average length of 150 words. Scoring involves 1st Reader Score, 2nd Reader Score, Resolved CR Score. If Reader‐1 Score and Reader‐2 Score are exact or adjacent, adjudication by a third reader is not required. On the other hand. If they are not adjacent or exact, then adjudication by a third reader is required.

Rubric range: 0-3

Resolved CR score range:0-3

The fourth set has 1,772 level 10 essays with a final evaluation set size of 589. The average length of 150 words. Scoring involves 1st Reader Score, 2nd Reader Score, Resolved CR Score. If Reader‐1 Score and Reader‐2 Score are exact or adjacent, adjudication by a third reader is not required. On the other hand. If they are not adjacent or exact, then adjudication by a third reader is required.

The fifth set has 1,805 level 8 essays with a final evaluation set size of 601. The average length of 150 words. Scoring involves Final, Score1, Score2. For the specific set of data, if there was a difference between scorer 1 and scorer 2, the FINAL SCORE was always the higher of the two. Final score range: 0-4, Rubric range: 0-4

The sixth set has 1,800 level 10 essays with a final evaluation set size of 600. The average length of 150 words. Scoring involves Final, Score1, Score2. For the specific set of data, if there was a difference between scorer 1 and scorer 2, the FINAL SCORE was always the higher of the two. Final score range:0-4, Rubric range:0-4

The fifth set has 1,730 level 7 essays with a final evaluation set size of 576. The average length of 250 words. Scoring involves Rater\_1, Rater\_2, Resolved\_Score. Scores summed independently for Rater\_1 and Rater\_2. Resolved Score = Rater\_1 + Rater\_2. Rubric range: 0-15, Resolved score range: 0-30

The eight set has 918 level 10 essays with a final evaluation set size of 305. The average length of 650 words. Scoring involves Rater1Comp, Rater2Comp, Rater3Comp, Resolved Score. Total Composite Score:

For most essays:

= (I\_R1+I\_R2) + (O\_R1+O\_R2) + (S\_R1+S\_R2) + 2 (C\_R1+C\_R2)

When there is Rater 3 set of scores for the essay then the Total Composite Score formula changes to:

= 2 (I\_R3) + 2 (O\_R3) + 2 (S\_R3) + 4 (C\_R3) or equivalently 2 (I+O+S+C) + 2 (C)

Rater1Comp Rubric range: 0-30

Rater2Comp Rubric range: 0-30

Rater3Comp Rubric range: 0-60

Resolved score range: 0-60

## Data Preprocessing

Here I will describe how I applied the LSTM with Word2Vec embeddings and the BERT model for AES. The code is implemented in Python. All the Python code used for development and analysis are provided.

### Loading the dataset

The training dataset contains columns essay\_id, essay\_set, essay, rater1\_domain1, rater2\_domain1, rater3\_domain1,domain1\_score, rater1\_domain2, rater2\_domain2, domain2\_score, rater1\_trait1, rater1\_trait2, rater1\_trait3,rater1\_trait4, rater1\_trait5, rater1\_trait6, rater2\_trait1, rater2\_trait2,rater2\_trait3, rater2\_trait4, rater2\_trait5, rater2\_trait6,rater3\_trait1, rater3\_trait2, rater3\_trait3, rater3\_trait4, rater3\_trait5, rater3\_trait6.

### Extract target variable

The first step is to extract the target variable y, “domain1\_score”

### Data cleaning

To deal with null values, I dropped columns with missing values.

To further filter my data, I additionally dropped columns that will not be needed “rater1\_domain1” and “rater2\_domain1”. This data is cleaned in this way to avoid noise during training.

## LSTM

### Text preprocessing

### Cleaning and Tokenizing text

I created the function essay\_to\_wordlist to clean and tokenizes text into individual words. Function essay\_to\_wordlist takes an essay text as a string input Firstly the function replaces non-letter characters([^a-zA-Z]) with a space. Secondly it modifies all the letters to lowercase then splits text into a list of words for uniformity. Lastly, it facilitates the removal of stopwords which are frequently occurring words like “the” and “and” which carry little semantic meaning to focus on meaningful content by making use of the NLTK stopwords list. The result of essay\_word\_list is a list of cleaned text.

Function called essay\_to\_sentences strives to divide an essay into sentences. These sentences are stored then passed through the essay\_to\_wordlist for tokenization. This function splits an essay into sentences, then tokenizes each sentence into words using essay\_to\_wordlist. Output is list of tokenized sentences, where each sentence is represented as a list of words.

### Feature Extraction

The makeFeatureVec function converts a list of words into a numerical feature vector using a Word2Vec model containing word embeddings with a size of 300 dimensions. Its input is a list of words from an essay. Attribute num\_features initializes the vector length to zero. The Word2Vec’s model vocabulary is accessed using model.wv.key\_to\_index (for Gensim 4.x). For each word in the input list, if it exists in the model's vocabulary, add its vector to the feature vector. An average is then computed by diving the sum by the number of words to ascertain that essays of different lengths produce vectors of the same size. The purpose of this extraction is to capture semantic relationships

### Feature Matrix

4. getAvgFeatureVecs function generates feature vectors for an entire set of essays. It receives an input of essays in the form of a list of words. A zero matrix of shape (number of essays, num\_features) is created to store feature vectors for all essays. For each essay in the input list, call makeFeatureVec to compute its feature vector and store it in the matrix. Increment the counter to process the next. Output: A matrix where each row is the feature vector for an essay.

### Model Architecture

The LSTM model consists of two layers where each is aimed at capturing sequential dependencies. The first layer applied a dropout rate of 0.4 was implemented as a way to prevent overfitting. The second layer uses a smaller hidden size of 64 to refine these dependencies for efficient computation. Additionally, dropout layers were added to both LSTM and dense layers for increased generalization. It mitigates overfitting by randomly deactivating neurons during the training process. The output layer is a single dense layer that employs ReLU activation for continuous score predictions. By so doing it ensures that all output scores are non-negative.

To evaluate this model, K-Fold cross-validation was applied with five splits. This method was tested on subsets of the data for increased model vigour while also addressing potential overfitting issues. Similarly, the Quadratic Weighted Kappa (QWK) metric was utilized as it has the capacity to measure agreement between predicted and actual scores while leaving room for the ordinal nature of essay scores.

## BERT

### Preprocessing

All missing entries to ensure consistency in training data. Retained only the essay and domain1\_score columns for simplicity.

### Tokenization

For tokenization, I employed the BertTokenizer to divide essays into subword components, followed by truncating them into 512 tokens, and padding shorter essays to maintain uniformity.

### Model Architecture

Fine-tuned the bert-base-uncased model to predict continuous scores for essays.

A regression head (dense layer) was added on top of BERT for an output of a singular numerical value for essay scores. The Bidirectional Contextual Understanding which is possessed by this model is essential for capturing relationships in essay corpora. By Leveraging BERT’s pre-trained language knowledge, there is a minimized need for extensive labelled data. A PyTorch-compatible Dataset class was applied to sort tokenized essays and scores for efficient batching and handling of inputs during training. The max\_length parameter ascertains essays follow to BERT's fixed input size aspect.

### Model Training

The batch size was set to 16 for training and evaluation to balance performance and memory constraints. The learning rate was configured at 5e-5, for fine-tuning pre-trained BERT models. To prevent overfitting while ensuring sufficient training Epochs were limited to 2. To enable quicker training and reduced memory usage I implemented mixed Precision Training (fp16). Steps of 2 were used to simulate a larger batch size for stability.

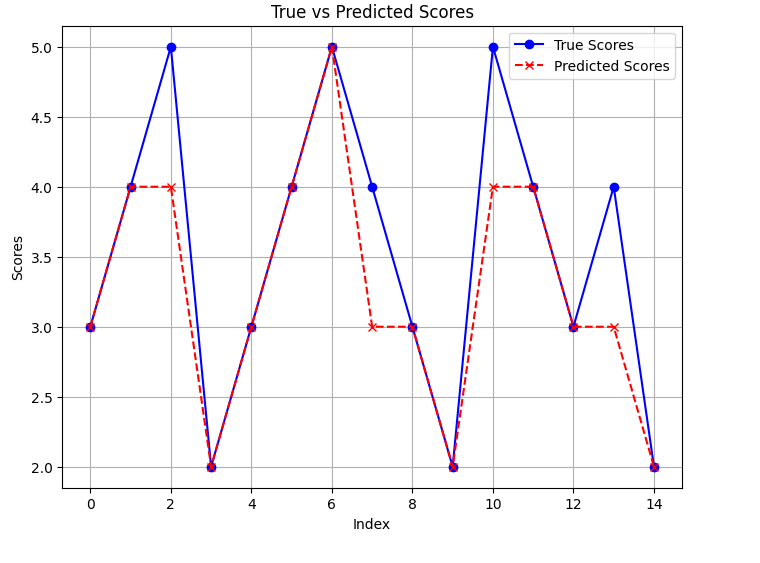
The Trainer API class from transformers trimmed the fine-tuning process, integrating training, evaluation, and metric calculation smoothly. After each epoch, evaluations were performed to monitor the model's progress and retain the best-performing checkpoint.

# Results and Discussion

| **Metric** | **LSTM** | **BERT (Epoch 1)** | **BERT (Epoch 2)** |
| --- | --- | --- | --- |
| **Quadratic Kappa (QWK)** | 0.9688 | - | - |
| **Mean Squared Error (MSE)** | 4.0 | 17.6991 | 11.9388 |
| **Training Loss** | 6.8146 | 21.1622 | 13.2029 |
| **Validation Loss** | - | 17.6991 | 11.9388 |
| **Mean Absolute Error (MAE)** | 0.2667 (Testing) | 1.5910 | 1.3283 |
| **R-squared (R²)** | 0.7458 | 0.7800 | 0.8516 |

## LSTM Model Performance

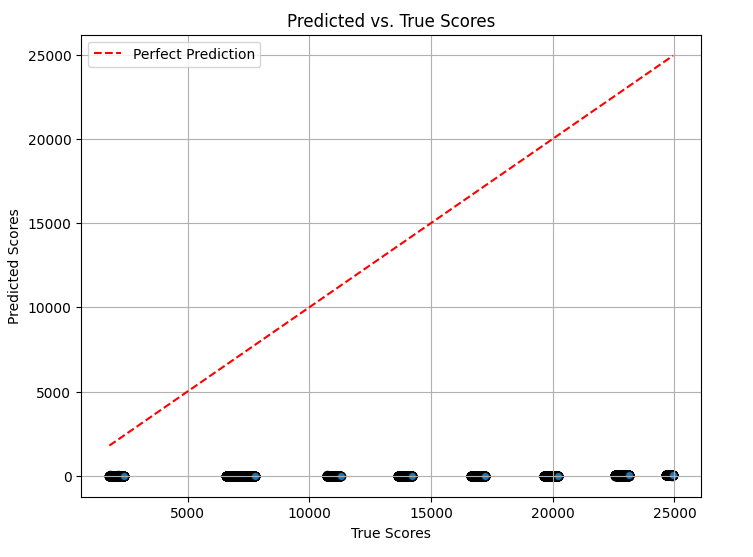
The LSTM model achieved a Quadratic Weighted Kappa (QWK) score of 0.9688. This demonstrates near perfect agreement with true essay scores. This is a firm indication of its precision and accuracy in evaluating essays. The Mean Squared Error (MSE) of 4.0 reflects a low average error in predictions made. Additionally, the Mean Absolute Error (MAE) is lies at 0.2667 which is quite low. This indicates deviations of most predictions from true values are by less than one-third of a point. The R² score of 0.7458 suggests that roughly 74.58% of the variance in the essay scores is captured by the model, implying a good but not perfect quality fit to the data.



: LSTM-True scores vs predicted scores

## BERT Model Performance

Notably, the BERT model's performance substantially improved between Epoch 1 and Epoch 2. In Epoch 1, the MSE was 17.6991, and the R² was 0.7800, indicating reasonable and moderate predictive power. As seen in Epoch 2, the MSE reduced to 11.9388, and R² increased to 0.8516, which indicates better performance with more training. The MAE dropped from 1.5910 in Epoch 1 to 1.3283 in Epoch 2, showing that the average deviation from the true scores decreased with training implying that there is a reduction in errors.



: BERT-True scores vs predicted scores

## Comparison of Models

Regarding precision The LSTM model surpassed BERT in terms of error metrics (MSE and MAE) and captured variance (R²) despite the latter's potential for contextual understanding. BERT demonstrated refinement of results across epochs. This implies that to understand essay content, it benefits greatly from longer training.

The LSTM model's strong performance outlines its suitability for tasks requiring sequence-based evaluation, such as essay scoring. It is efficient at capturing temporal dependencies making it adept at identifying patterns across essay texts. However, although BERT initially underperformed compared to LSTM it shows promising potential as it continues to train. As a result of its contextual embedding capabilities and aptitude to grasp deeper semantic relationships in text. With continued epochs and parameter tuning, BERT could possibly surpass LSTM.

For real-world deployments, the LSTM model offers a reliable, fast, and effective approach however this comes with high computational costs. BERT may be more beneficial in situations requiring intricate text analysis or multilingual essay evaluation.

In future a collaboration of LSTM for structural text processing and BERT for contextual understanding could evidently yield a hybrid model with superior performance. Further exploration of fine-tuning techniques on BERT for essay-specific datasets or incorporating additional linguistic features could further advance its capabilities.

# References

Attali, Y. and Burstein, J. (2006) “Automated Essay Scoring With e-rater® V.2”, The Journal of Technology, Learning and Assessment, 4(3). Available at: <https://ejournals.bc.edu/index.php/jtla/article/view/1650>.

Dikli, S. (2006) “An Overview of Automated Scoring of Essays”, The Journal of Technology, Learning and Assessment, 5(1). Available at: <https://ejournals.bc.edu/index.php/jtla/article/view/1640>.

Firoozi, T. et al. (2023) The Effect of Fine-tuned Word Embedding Techniques on the Accuracy of Automated Essay Scoring Systems Using Neural Networks, Journal of Applied Testing Technology. Article, pp. 21–29. <https://www.researchgate.net/publication/368236904>.

Landauer, T.K., Foltz, P.W. and Laham, D. (1998) 'An introduction to latent semantic analysis,' Discourse Processes, 25(2–3), pp. 259–284. <https://doi.org/10.1080/01638539809545028>.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv. <https://doi.org/10.48550/arXiv.1301.3781>.

Page, Ellis B. “The Imminence of... Grading Essays by Computer.” The Phi Delta Kappan, vol. 47, no. 5, 1966, pp. 238–43. JSTOR, <http://www.jstor.org/stable/20371545>.

Ramesh, D., 1, Sanampudi, S.K., and The Author(s) (2022) An automated essay scoring systems: a systematic literature review, Artificial Intelligence Review, pp. 2495–2527. <https://doi.org/10.1007/s10462-021-10068-2>.

Shermis, M.D. and Burstein, J. (2013) Handbook of Automated Essay Evaluation, Routledge eBooks. <https://doi.org/10.4324/9780203122761>.

Steffen Brandt (2021) Automated Essay Scoring Using Transformer Models, Psych, pp. 897–915. <https://doi.org/10.3390/psych3040056>.