A GRU-based pipeline approach for word-sentence segmentation and punctuation restoration in English

A GRU-based Pipeline Approach for Word-Sentence Segmentation and Punctuation Restoration in English

Jasivan Sivakumar, Jake Muga, Flavio Spadavecchia, Daniel White, Burcu Can
University of Wolverhampton
Wolverhampton, United Kingdom
{j.sivakumar, j.k.muga, f.spadavecchia, d.p.white, b.can}@wlv.ac.uk

***Abstract*— In this study, we propose a Gated Recurrent Unit (GRU) model to restore the following features: word and sentence boundaries, periods, commas, and capitalisation for unformatted English text. We approach feature restoration as a binary classification task where the model learns to predict whether a feature should be restored or not. A pipeline approach is proposed, in which only one feature (word boundary, sentence boundary, punctuation, capitalisation) is restored in each component of the pipeline model. To optimise the model, we conducted a grid search on the parameters. The effect of changing the order of the pipeline is also investigated experimentally; PERIODS > COMMAS > SPACES > CASING yielded the best result. Our findings highlight several specific action points with optimisation potential to be targeted in follow-up research.**

I. Introduction

Word and sentence segmentation, the restoration of punctuation, and capitalisation are long-standing tasks in computational linguistics. In particular, these tasks are relevant to the development of automatic speech recognition and transcription tools, since such fields require segmenting a continuous sequence of spoken language into individual words. However, most of the currently available speech recognition tools do not segment transcribed text into sentences or restore punctuation [1]. Moreover, hashtag segmentation is becoming important in understanding noisy text especially on social media [2]. Our work, therefore, contributes to the existing research by proposing an automatic tool to segment text and restore key punctuation features not only for white space delimited languages like English but also scriptio continua ones like Chinese.

The use of hashtagged expressions has become more prevalent in society, especially on social media. In offensive language detection, hate speech can be masked behind those expressions. Therefore, a good segmentation tool is required to help extract the meaning of such hashtags which do not include any space or punctuation between word tokens [2], which is one of the motivations behind this work. Only a handful of Asian languages are systematically written without breaks between words in a style known as scriptio continua. By far the most widely used language written in this way is Chinese, and a significant amount of research has been conducted on how to process this language as a separate task in computational linguistics.

With regards to all languages, automatic speech recognition needs to identify word and sentence boundaries, and punctuation. A recent study [1] addresses the lack of automatic speech recognition systems to implement punctuation restoration in addition to word and sentence segmentation. In their study, a BERT-based model is proposed to restore three classes of punctuation marks (commas, periods and question marks), which are noted as the most frequently occurring punctuation marks in Hungarian. The authors propose a sequence classification approach, which achieves performance comparable with the state-of-the-art.

Our novel approach uses bigrams to predict the need for restoring the aforementioned features using a multi-stage process in a pipeline setting. The model is based on a multi-layer Gated Recurrent Unit (GRU). To our knowledge, no research has been conducted in restoring both word and sentence segmentation along with punctuation and capitalisation within the same model. Moreover, these tasks have not been tackled as a bigram classification task before.

II. Related Work

A. Punctuation Restoration

Regarding the work on punctuation restoration, all recent state-of-the-art approaches utilise neural models [1, 3, 4]. Kim [4] approaches the task using a stacked bidirectional GRU model which processes tokens and previous hidden states from both directions to generate a class. The model involves a multi-layer attention mechanism before the final classification layer. Their model only identifies whether punctuation is required or not. On the other hand, Courtland, Faulkner and McElvain [3] and Nagy, Bial and Ács [1] use pre-trained BERT embeddings in their models. They both use a sliding window and generate multiple classifications for each token which they aggregate and then take the most probable one. This method allows them to use an attention within each window thus extracting more than a unique hidden state from adjacent inputs. Since these papers focus on punctuation restoration, they already assume correct word segmentation and thus look at the token level instead of the character level. On the contrary, Li et al. [5]

# of round	Feature restored	Input	Target
1	SPACE	"<SOS>helloworld<EOS>" [3264, 918, 542, 1452, 1458, 1822, 170, 2634, 2214, 300, 2650]	"<SOS>hello world<EOS>" [0, 0, 0, 0, 0, **1**, 0, 0, 0, 0, 0]
2	PERIOD	"<SOS>hello world<EOS>" [3264, 918, 542, 1452, 1458, 1884, 4162, 170, 2634, 2214, 300, 2650]	"<SOS>hello world.<EOS>" [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, **1**]
3	COMMA	"<SOS>hello world.<EOS>" [3264, 918, 542, 1452, 1458, 1884, 4162, 170, 2634, 2214, 300, 2663, 4145]	"<SOS>hello, world.<EOS>" [0, 0, 0, 0, 0, **1**, 0, 0, 0, 0, 0, 0, 0]
4	CASING	"<SOS>hello, world.<EOS>" [3264, 918, 542, 1452, 1458, 1882, 4094, 4162, 170, 2634, 2214, 300, 2663, 4145]	"<SOS>**H**ello, **W**orld.<EOS>" [**1**, 0, 0, 0, 0, 0, 0, **1**, 0, 0, 0, 0, 0]

TABLE I
EXAMPLE INPUT (INPUT CHARACTER SEQUENCES AND THE VECTORS OF BIGRAM FEATURE INDICES) AND TARGET OUTPUTS (OUTPUT CHARACTER SEQUENCE AND THE BINARY FEATURE VECTOR THAT CORRESPONDS TO THE TARGET FEATURE)

argue that for Chinese, language exploring a character level processing is more efficient than token level even when the latter is provided.

Relating to punctuation, Ramena et al. [6] focus on true-casing which is the restoration of capitalisation. They use a combination of Convolutional Neural Networks (CNNs) [7], Long Short Term Memory (LSTMs) networks [8] and Conditional Random Fields (CRFs). They also approach the task as a binary classification problem.

B. Word Segmentation

For word segmentation, the recent neural methods consider the problem as a classification problem [9, 10]. Shao et al. [9] use a bidirectional GRU model with a CRF to segment words and assign part-of-speech tags (PoS) simultaneously in Chinese. They use pre-trained GloVe [11] character embeddings. The word segmentation labels include B-beginning, I-inside, E-end, and S-single character. In addition to the character embeddings, they also use trigram character windows as well as bigrams. Che et al. [10] focus on different features but still considers the problem as a character classification task. They use a combination of character and bigram embeddings, normalised pointwise mutual information between adjacent characters. The embeddings are obtained using word2vec [12]. All of them are then fed into a bidirectional LSTM which outputs either B-begin or I-inside. Any "IB" pair would thus denote a word boundary. For the sentence boundary, they use a rule-based approach instead of a neural network.

III. METHODOLOGY

We propose a novel pipeline approach for word and sentence segmentation, punctuation restoration (including capitalisation) for English text whose formatting such as punctuation, capitalisation, and white spaces are lost; for instance, "unfortunatelytherhetoricdoesntmatchthereality". Therefore, the model basically seeks to restore four common classes of punctuation: spaces, casing, periods, and commas. The terms "punctuation marks" and "features" refer to the four features that our model aims to restore, and thus both will be used interchangeably throughout the paper.

A. Dataset and Pre-processing

We use the Brown corpus [13] for all experiments. The dataset contains approximately one million words tokens. Here, we describe how the dataset is pre-processed for the experimental setup:

1) Generation of pseudo-paragraphs: First, the entire Brown corpus was segmented into sentences and pseudo-paragraphs of four sentences. 14,335 pseudo-paragraphs were obtained in total. The data was split into training and testing sets in the ratio 80:20, which eventually contained 11,468 and 2,867 pseudo-paragraphs, respectively. The pseudo-paragraphs in the datasets were arranged in ascending order based on the number of word tokens to facilitate the batching process.

2) Removal and replacement of unwanted characters: Data was normalised to process diacritics and font styles that could impede the processing of the data. All other punctuation marks (apart from spaces, casing, commas, and periods) were replaced or removed from the dataset as follows:

- Exclamation and question marks were replaced with periods.
- Colons and semicolons were replaced with periods, and the initial letter of the following word was capitalised.
- Dashes and hyphens were replaced with spaces, and subsequently, all double spaces were replaced with single spaces.
- Percentage signs were replaced with the word "percent".
- All other punctuation marks, such as quotation marks, apostrophes, and currency symbols, were removed.

All other punctuation marks that are not mentioned above were removed from the dataset.

3) Addition of SOS and EOS tokens: Due to our model being based on bigrams, the model is unable to restore punctuation for characters in the initial and final positions of the character sequences. To ensure that the first character is considered for capitalisation and, likewise, to allow the model to predict punctuation marks at the end of the character sequences, <SOS> (start-of-sequence) and <EOS> (end-of-sequence) tokens were inserted in the initial and final positions of each sequence respectively. Table II shows the distribution of the features in both train and test sets. It is important to note that the training and testing sets have similar distributions of each feature. This ensures the validity and reliability of the testing.

4) Generating input and target data: In each component of the pipeline approach, two versions of the data were created: a list of input strings containing only previously learned features and a list of target strings additionally containing the specific feature being restored in that component. Table I provides an overview of the input and target sequences and corresponding feature vectors of each of the four stages of the restoration process. The input was subsequently tokenised into bigrams, vectorised and arranged in batches. The target strings were converted into target sequences of 0s and 1s, where 1 indicates

	Training		Testing		Clean Training		Clean Testing	
Features	**Count**	**Percentage**	**Count**	**Percentage**	**Count**	**Percentage**	**Count**	**Percentage**
Periods	45,509	1.1%	10,069	1.51%	56,154	1.36%	13,268	1.99%
Commas	49,139	1.19%	9,843	1.48%	49,139	1.19%	9,843	1.48%
Spaces	855,958	20.82%	151,109	22.72%	858,158	20.87%	151,328	22.76%
Capitals	109,463	2.66%	19,452	2.92%	112,234	2.73%	19,787	2.97%

TABLE II
SUMMARY OF THE FEATURES IN TRAIN AND TEST SETS.

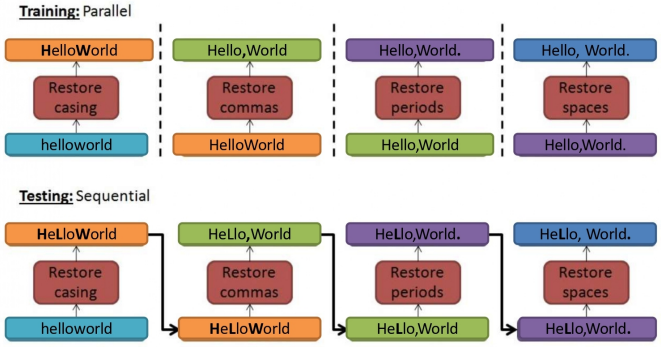

Fig. 1. The overview of the proposed pipeline neural network in training and testing.

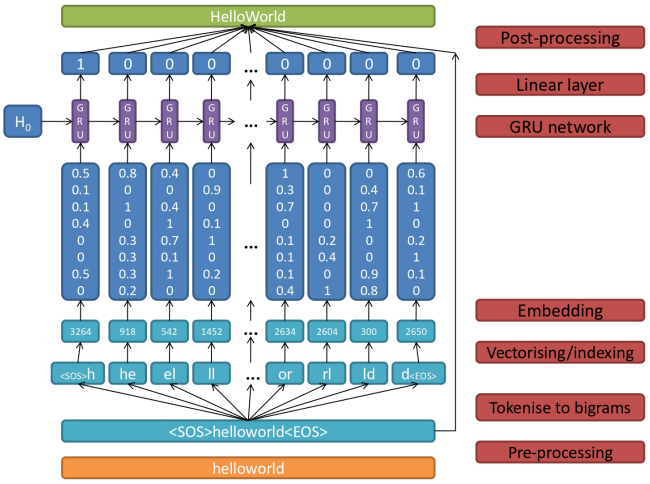

Fig. 2. The proposed GRU-based network for one particular feature restoration.

that the desired feature needs to be restored and 0 indicates no restoration is required.

B. The Pipeline Approach

We propose a neural network using GRUs [14]. Rather than restoring all punctuation features simultaneously, a pipeline network is proposed where each pipeline component is utilised to learn a particular feature. The overview of both training and testing are given in Figure 1. One GRU network is built for each specific task, which are all trained independently. However, during testing, each component processes the output of the previous component in a pipeline framework.

Embeddings are generated for character bigrams to be fed into GRUs throughout the pipeline, which are randomly initialised and trained along with other parameters in the model. Each component (for each particular task) of the pipeline network consists of three GRU layers and receives bigram embeddings as input, e_t, processes the output of the GRU layers through a linear layer with σ activation function, and outputs a feature, o_i (see Figure 2 for one component in the pipeline network):

$$\text{Layer 1: } (h^1_{t+1}, p^1_{t+1}) = GRU(e_t, h^1_{t-1}) \quad (1)$$

$$\text{Layer 2: } (h^2_{t+1}, p^2_{t+1}) = GRU(p^1_t, h^2_{t-1}) \quad (2)$$

$$\text{Layer 3: } (h^3_{t+1}, p^3_{t+1}) = GRU(p^2_t, h^3_{t-1}) \quad (3)$$

$$\text{Linear Layer: } (o_0, o_1) = \sigma(p^3_t) \quad (4)$$

$$\text{Final Layer: } output = i, \ where \ o_i = argmax(o_0, o_1) \quad (5)$$

p^i_t is the prediction generated by the GRU for the current time step. Moreover, all initial hidden states, h^i_0 are all set to 0.

To train the model, we use cross-entropy loss, where the paired output, (o_0, o_1) is passed through a softmax function and compared with the target output. We use Adam optimiser for training.

The output of the model is a series of 0s and 1s, which is combined with the input text to implement the restoration of each feature. For example, when restoring spaces, given the string "<SOS>Adogis<EOS>" and the labels $[0, 1, 0, 0, 1, 0, 1, 0]$, we would insert spaces between the respective bigrams to produce "<SOS>A dog is <EOS>". Subsequently, the <SOS> and <EOS> tokens are removed to produce "A dog is" eventually.

IV. EXPERIMENTS AND RESULTS

Each component in the pipeline network was trained independently on pre-processed input text. So we assumed that previously considered punctuation marks had all been restored correctly. For example, if we had restored spacing and we wished to restore capitalisation, we would have the following:

- Original text: "Hello, World."
- Pre-processed input text: "<SOS>hello world<EOS>" (only the correct spaces are included)
- Expected output: "Hello World"
- Expected label: $[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$ (the 1s are where the second character in the bigram should be capitalised)

When testing the models, we used the test data to pass it through each component in the pipeline sequentially (see Figure 1). Therefore, if there is an error in the first component of the pipeline network, that remains throughout the predictions, e.g. the capital L in the testing pipeline in Figure 1.

We evaluated the restoration of each component's features separately. For instance, at the third component in Figure 1, we aimed to restore commas. At that component, only commas were considered for the evaluation against the target string "Hello, World" containing the correct spaces and capitalisation. Finally, the final output of the pipeline was evaluated against the gold standard target string.

A. Pipeline search

When using a pipeline approach, the order of the processes may affect the overall result. Therefore, a pipeline search involving different orders of the pipeline was evaluated on the test data to find the pipeline order that achieves the best performance. For instance, it was hypothesised that restoring the punctuation in the order, spaces, commas, periods and casing would yield different results than another order, such as casing, periods, commas, and spaces. It could be also claimed that some orders are linguistically more motivated than the others. In our experimental setting, this type of search generated comparable results to support decision-making for the optimal pipeline model.

B. Grid search

We also conducted a grid search for the optimisation of the parameters in the model:

- batch size $\in \{32, \mathbf{64}, 128\}$
- hidden size $\in \{50, 100, \mathbf{150}, 200\}$
- embedding size $\in \{150, \mathbf{200}, 250\}$
- number of layers $\in \{1, 2, \mathbf{3}\}$
- learning rate $\in \{0.01, 0.005, \mathbf{0.001}\}$

After attempting 17 different combinations of parameter values, we decided on using the parameters shown in bold based on the run times and loss achieved.

C. Evaluation criteria

Five different evaluation metrics are utilised for the evaluation of the results: accuracy, recall, precision, F-score, and Damerau-Levenshtein distance. The Damerau-Levenshtein distance measures the edit distance of the output from the human-proofread target text. This method is a commonly used metric for evaluating the quality of text editing tasks. It considers four types of editing operations to compare two sequences: deletion, insertion, substitution, and transposition. In our evaluation, the distance was always computed between the input and target at each stage in the pipeline. Unlike with the Levenshtein-Damerau computation, when calculating accuracy, precision, recall and F-score, we only considered the feature that is being restored in that component. More importantly, all these metrics are calculated by analysing bigrams and their associated labels. For example, given the pipeline CASING $>$ COMMAS $>$ PERIODS $>$ SPACES, let us consider the testing example from Figure 1. In the second component, the accuracy will be 100% and, the recall, precision and F-score will be 1 because the comma is in the bigram "ob", which is correct despite the extra capital letter that is being carried over from the first restoration.

D. Results

Tables III and IV provide an overview of the performance of different pipelines (i.e. different orders in which the features were restored) in both training and testing. For the average accuracy, recall, precision, F-score across all feature restorations, we present the weighted average using the frequencies of the punctuation marks that need restoring to mitigate the effect of the imbalance between different features in train set (see Table II). For the final loss, we present the sum of the losses of each component in the pipeline. Finally, we present the maximum distance as the final edit distance.

The second pipeline (PERIODS $>$ COMMAS $>$ CASING $>$ SPACES) gives the overall best results. This is noteworthy, since this pipeline matches the distribution of the features within the initial training dataset from least frequent to most frequent. This is possibly due to possessing more information in restoring spaces. It is easier to ensure spaces are added near other punctuation marks than initially finding them. Moreover an accurate restoration of spaces weights the overall score more due to its dominance in the distribution.

One aspect of the results that particularly stands out is that the Damerau-Levenshtein distance is not consistent compared to other metrics, which will be discussed in the following section. Pipelines 3, 5 and 7 were all consistent in training and testing, and they all restored casing at the very end. Pipelines 2, 3 and 8 achieved lower loss, which may indicate over-fitting. In these cases, capitalisation was restored before spacing.

E. Discussion

The results show that the model is clearly capable of restoring certain features. Looking at the results, there seems to be a trend whereby metrics such as accuracy, precision, recall and f-score all generally agree with each other, while the Damerau-Levenshtein distance metric is more diverging. Having access to this variety of metrics is certainly a point of merit of this study, since it allows us to think more critically about which metrics are best for this task. The stark difference between the training and testing results of the 8^{th} pipeline would suggest overfitting, as this ordering of feature restoration did much better in training and it has the smallest cumulative loss, since dropout was not used. Conversely, the first pipeline performed the worst in training, while in testing it was somewhat better.

Training		Loss	Accuracy	Recall	Precision	F-score	Distance
Pipeline 1	Spaces	0.01	99.5	0.99	0.99	0.99	2.3
	Casing	0.01	99.6	0.94	0.89	0.91	1.9
	Comma	0.01	99.8	0.98	0.96	0.97	20.2
	Period	0.00	99.9	0.99	0.98	0.99	20.2
	Summary	0.03	99.5	0.98	0.98	0.98	20.2
Pipeline 2	Period	0.00	99.9	0.96	0.94	0.95	0.9
	Comma	0.01	99.8	0.95	0.91	0.93	0.7
	Casing	0.01	99.8	0.97	0.96	0.96	0.8
	Spaces	0.01	99.7	0.99	0.99	0.99	62.0
	Summary	0.02	99.7	0.99	0.98	0.98	62.0
Pipeline 3	Period	0.00	99.9	0.95	0.92	0.93	1.0
	Comma	0.00	99.9	0.95	0.92	0.93	0.7
	Spaces	0.01	99.7	0.99	0.99	0.99	1.5
	Casing	0.00	99.8	0.97	0.96	0.96	0.8
	Summary	0.02	99.7	0.99	0.98	0.98	1.5
Pipeline 4	Spaces	0.01	99.6	0.99	0.99	0.99	2.0
	Casing	0.01	99.6	0.94	0.89	0.91	2.0
	Period	0.00	99.9	0.99	0.98	0.98	20.3
	Comma	0.01	99.8	0.98	0.97	0.97	19.9
	Summary	0.03	99.6	0.98	0.98	0.98	20.3
Pipeline 5	Spaces	0.01	99.5	0.99	0.99	0.99	2.1
	Period	0.00	99.9	0.94	0.91	0.92	1.1
	Comma	0.00	99.9	0.94	0.90	0.92	0.8
	Casing	0.01	99.8	0.97	0.95	0.96	0.9
	Summary	0.03	99.6	0.98	0.98	0.98	2.1
Pipeline 6	Casing	0.01	99.6	0.93	0.88	0.91	2.1
	Period	0.00	100.0	1.00	1.00	1.00	15.9
	Comma	0.00	99.9	0.99	0.98	0.99	20.8
	Spaces	0.01	99.7	0.99	0.99	0.99	62.0
	Summary	0.02	99.7	0.99	0.98	0.98	62.0
Pipeline 7	Comma	0.01	99.8	0.95	0.89	0.92	0.9
	Spaces	0.11	99.6	0.99	0.99	0.99	1.8
	Period	0.00	99.9	0.96	0.93	0.94	1.0
	Casing	0.01	99.8	0.97	0.95	0.96	0.9
	Summary	0.12	99.7	0.98	0.98	0.98	1.8
Pipeline 8	Casing	0.01	99.5	0.93	0.88	0.90	2.1
	Spaces	0.00	100.0	1.00	1.00	1.00	0.0
	Period	0.00	99.9	0.99	0.98	0.98	20.3
	Comma	0.01	99.8	0.98	0.96	0.97	19.9
	Summary	0.02	99.9	0.99	0.98	0.99	20.3
Pipeline 9	Comma	0.01	99.8	0.95	0.91	0.93	0.8
	Casing	0.01	99.6	0.94	0.89	0.91	2.0
	Period	0.00	100.0	1.00	1.00	1.00	15.8
	Spaces	0.01	99.7	0.99	0.99	0.99	61.9
	Summary	0.02	99.7	0.99	0.98	0.98	61.9

TABLE III
PERFORMANCE OF DIFFERENT PIPELINES DURING TRAINING WITH A GREEN TO RED SCALE FROM BEST TO WORST PERFORMANCE

Testing		Accuracy	Recall	Precision	F-score	Distance
Pipeline 1	Spaces	95.0	0.89	0.89	0.89	11.7
	Casing	96.1	0.34	0.35	0.35	20.4
	Comma	92.2	0.36	0.23	0.29	37.9
	Period	90.5	0.50	0.18	0.27	50.7
	Summary	94.7	0.79	0.77	0.77	50.7
Pipeline 2	Period	99.9	0.96	0.94	0.95	0.9
	Comma	99.8	0.95	0.91	0.93	0.7
	Casing	99.8	0.97	0.96	0.96	0.8
	Spaces	99.7	0.99	0.99	0.99	62.0
	Summary	99.7	0.99	0.98	0.98	62.0
Pipeline 3	Period	98.1	0.34	0.44	0.38	5.0
	Comma	97.9	0.13	0.18	0.15	9.4
	Spaces	94.7	0.88	0.89	0.88	21.4
	Casing	96.4	0.31	0.37	0.34	29.4
	Summary	96.4	0.36	0.42	0.39	29.4
Pipeline 4	Spaces	95.0	0.89	0.89	0.89	11.7
	Casing	96.1	0.34	0.34	0.34	20.5
	Period	92.5	0.48	0.22	0.30	39.2
	Comma	90.2	0.37	0.18	0.24	51.9
	Summary	91.1	0.41	0.24	0.30	51.9
Pipeline 5	Spaces	95.0	0.89	0.89	0.89	11.7
	Period	97.8	0.33	0.38	0.35	17.0
	Comma	97.7	0.12	0.16	0.14	21.5
	Casing	96.0	0.33	0.33	0.33	30.3
	Summary	95.4	0.75	0.76	0.76	30.3
Pipeline 6	Casing	96.4	0.32	0.38	0.35	8.3
	Period	94.1	0.41	0.27	0.32	21.7
	Comma	89.4	0.36	0.16	0.22	43.9
	Spaces	74.9	0.85	0.48	0.61	106.9
	Summary	79.2	0.74	0.44	0.54	106.9
Pipeline 7	Comma	98.0	0.11	0.21	0.14	4.5
	Spaces	94.9	0.89	0.89	0.89	16.3
	Period	97.9	0.31	0.38	0.34	21.0
	Casing	96.1	0.31	0.33	0.32	29.7
	Summary	95.4	0.75	0.76	0.75	29.7
Pipeline 8	Casing	96.3	0.32	0.37	0.34	8.5
	Spaces	96.3	0.32	0.37	0.34	61.3
	Period	95.7	0.40	0.38	0.39	67.4
	Comma	94.6	0.27	0.33	0.30	72.5
	Summary	94.9	0.29	0.34	0.31	72.5
Pipeline 9	Comma	97.8	0.13	0.18	0.15	5.0
	Casing	96.4	0.31	0.37	0.34	13.3
	Period	94.1	0.40	0.27	0.32	26.0
	Spaces	78.8	0.89	0.53	0.66	83.3
	Summary	82.6	0.76	0.47	0.58	83.3

TABLE IV
PERFORMANCE OF DIFFERENT PIPELINES DURING TESTING WITH A GREEN TO RED SCALE FROM BEST TO WORST PERFORMANCE

Pipelines 6, 8 and 9 performed the worst; in these cases, capitalisation was restored before spacing. Indeed, we argue that the distance metric was much better at predicting how legible a sentence would be. For instance, this is the result obtained from the 3^{rd} pipeline, with the lowest edit distance: "`<SOS>But i ll be fairnowhe is coming back is nt he frank yes, ohyes, whatelse was there to say.<EOS>`", which is predicted as follows in the second pipeline that achieves the highest scores (except the distance): "`<SOS> , but.,ill.,be.,fairnow., heis.,comIng .,back.,is.,nt.,he.,frAnk. ,yes.,o.,Hyes.,whatelsewas.,thereto.,say <EOS>`".

The results seem to point to the fact that capitalisation needs to be restored after spacing, which agrees with our linguistic intuition of how capitalisation works in English. The Levenshtein distance ultimately appears to be the best indicator as to how close the sentence is to the target output.

F. Limitations

In particular, greater focus should be put on the selection and pre-processing of the corpus material. Although the Brown corpus was designed to include a range of text types, the

collection is now 60 years old and contains dated terms and topics. Therefore, the proposed model may achieve lower accuracy when used on modern texts. Furthermore, if the corpus was more domain specific and less broad, it might be more successful in reconstructing paragraphs of a similar genre. Alternatively, it might be necessary to gather even more training data if the scope remains as broad as the Brown corpus. No preselection of the sentences in the corpus was conducted and all sentences were incorporated into the dataset. A filtering approach could be used to eliminate any unusual or particularly short sentence structures, like "Systemic linkage 4. Socialization 5." to improve the quality of the input data.

Regarding pre-processing, it might be worth investigating the use of a different tokeniser, such as Byte-Pair Encoding (BPE) as well as the use of trigrams (which are more computationally expensive). At the same time, limiting the vocabulary size might also yield positive results.

Furthermore, the way pseudo-paragraphs were assembled was by concatenating four sentences. This means that each paragraph in the training and testing data contains exactly four full stops, which might affect how the model divides sentences, which could differ from the mere linguistic criteria. As the model has been trained on clean data only, the model's ability may be impaired when a component in the pipeline does not output a perfectly restored text.

V. Conclusion and Future work

We proposed a GRU-based pipeline approach to restore four main features: spaces, periods, commas, and capitalisation. We performed rigorous experiments both on hyperparameter tuning using grid search and pipeline processing order of the different tasks. The results are promising and shows that our model could be used as a word/sentence boundary detection and punctuation restoration tool in either speech recognition systems or word editing tools. It is also adaptable to any language but the complexity increases with the alphabet size.

In the future, a clear extension to this task would be to restore other non-alphanumeric characters. Multi-task learning of all features is also another potential future goal. Although the Brown corpus was used to train the model, further research could be carried out to see how well it performs on different types of data, such as domain-specific data or even noisy data from social media but also consider character level embedding as a head start. Moreover, the performance of the model on different languages should be considered and even aimed to design a model that is language agnostic. Finally, an eye-tracking experiment in conjunction with a full pipeline search could be conducted to explore whether the artificial system's punctuation restoration correlates with that of the human brain.

References

[1] A. Nagy, B. Bial, and J. Ács, "Automatic punctuation restoration with BERT models," *CoRR*, vol. abs/2101.07343, 2021.

[2] H. Ahn, J. Sun, C. Y. Park, and J. Seo, "NLPDove at SemEval-2020 task 12: Improving offensive language detection with cross-lingual transfer," in *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Barcelona (online): International Committee for Computational Linguistics, Dec. 2020.

[3] M. Courtland, A. Faulkner, and G. McElvain, "Efficient automatic punctuation restoration using bidirectional transformers with robust inference," in *Proceedings of the 17th International Conference on Spoken Language Translation*. Online: Association for Computational Linguistics, Jul. 2020.

[4] S. Kim, "Deep recurrent neural networks with layer-wise multi-head attentions for punctuation restoration," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7280–7284.

[5] X. Li, Y. Meng, X. Sun, Q. Han, A. Yuan, and J. Li, "Is word segmentation necessary for deep learning of Chinese representations?" 2019.

[6] "An efficient architecture for predicting the case of characters using sequence models."

[7] M. Valueva, N. Nagornov, P. Lyakhov, G. Valuev, and N. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997.

[9] Y. Shao, C. Hardmeier, J. Tiedemann, and J. Nivre, "Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017.

[10] W. Che, J. Guo, Y. Wang, B. Zheng, H. Zhao, Y. Liu, D. Teng, and T. Liu, "The hit-scir system for end-to-end parsing of universal dependencies," in *CoNLL Shared Task*, 2017.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[13] W. N. Francis and H. Kucera, "Brown corpus manual," Department of Linguistics, Brown University, Providence, Rhode Island, US, Tech. Rep., 1979.

[14] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014.