Domain Transfer for Punctuation Retrieval

Ng Xing Yu

March 31, 2021

Contents

1 Introduction

Punctuation Retrieval is an important aspect in any Automatic Speech Recognition (ASR) pipeline for two reasons: to improve readability of auto-generated transcripts for videos or podcast subtitling or voice dictation applications, and to better capture the meaning of speech transcripts to improve the performance of downstream Natural Language Processing (NLP) tasks. Much research has been conducted for Punctuation Retrieval, often utilising acoustic features or language features like Capitalisation or Named-entity tags to improve the model's performance. However, there has not been much research on ways to improve performance of transfer to low-resource domains for punctuation retrieval.

The objective of my research is to look at methods to improve the performance of a punctuation retrieval model on domains with limited or no data with punctuation labels (zero or low-resource). There are various existing approaches taken to address zero or low-resource sequence labelling for other NLP tasks such as Named-Entity (NER) or Part-of-Speech (POS) tagging, which will be considered in this paper.

2 Punctuation Retrieval

2.1 Punctuation Features

The majority of research on punctuation retrieval for English speech transcripts condense all punctuation into four classes — (Period .) (Comma ,) (Question Mark ?) and (None), using a custom-defined mapping function to replace other punctuation with the four classes. This mapping is used by other researchers to handle the issue of a large imbalance in punctuation labels in most datasets, with less frequent punctuation like semicolons or dashes occurring under 1% in the entire Open Subtitles v2016 English corpus, resulting in a very low performance on rarer labels. Some of the rare punctuation such as (!) depend heavily on other acoustic cues such as pitch, making it better to replace them with the generic class — period, but others like the semicolon or hyphen can be considered.

2.2 Data

The data sources for training punctuation retrieval tasks can be categorised into purely textual, aligned audio with text, and text with other labels such as Capitalisation or Named-Entity tags, or prosodic information. The most common source used for evaluating models for punctuation retrieval is the IWSLT dataset with the 2012 version for punctuation retrieval being a pre-processed selection of TED talks before 2012. My project mainly made use of the TED Talks, Open Subtitles and Switchboard Telephone

corpus. The preprocessed and split version of the three datasets used in this paper can be found on kaggle. Other splits were explored in the paper, but all experiments use data from the same preprocessed csv files.

2.3 Other ideas

Other possible areas that can be considered for punctuation retrieval but were not explored extensively in this research includes:

1. **Use of non-lexical features.**

   Training a model using both acoustic and lexical features (Sunkara et al., 2020a), or using other lexical information such as POS tags from the Penn Treebank Yi et al. (2020a) or Capitalisation marks can improve performance of the punctuation retrieval.

2. **Data Augmentation.** Simulation of various forms of word errors, such as inserting or substituting existing tokens with the unknown token and randomly deleting tokens would increase robustness to transcription errors from upstream tasks. (Alam et al., 2020)

   Other possible noise injection techniques include random splitting of words into subwords or random deletion / insertion / substitution with homonyms or synonyms at the subword level. This would make the model more robust to noise in the inference dataset.

3. **Overlapping chunks.** Nguyen et al. (2019) looks into the impact of overlapping successive chunks that are fed into the model for inference, and observed that labels at the ends of the output sequence tend to perform worse than at the labels at the middle. Thus, using a sliding window and merging the classifier outputs for the overlapping tokens in the sequence can contribute to a substantial improvement in inference performance of the punctuation retrieval model. As observed in Table 1 row 19, the performance of the middle 50% of tokens for all examples is slightly higher at 79.0 as compared to the performance of 77.4 when looking at all tokens. An illustration would be this sentence from the switchboard corpus "what savings and so on. And then how" changing from "what savings? and// so on. and then, how " to "what savings and so on, and then how", when the window is shifted to the right by 16 words. ("//" denotes the end of a window of size 128.) This difference in prediction for the word "savings" is caused its inability to look at the context of words beyond its next word, making (?) the most logical label given its visible context, and the use of a sliding window or overlapping chunks can resolve this issue.

4. **Consideration of consecutive punctuation.** Another idea I had was to allow for the insertion of consecutive punctuation (e.g. The car is the Joneses**'.**) by iteratively passing text with previously labelled punctuation into the model to predict missing punctuation marks. This can improve readability and allow for the capturing of additional punctuation like (") which was not considered in this research. An illustration would be converting "he said hey" to "he said "hey"" in the first pass through the model and to "he said, "hey!"" in the second pass.

3 Models

The model for punctuation retrieval is usually made up of a base layer to convert word tokens to hidden states, and a decoder layer to convert the hidden states into a sequence of punctuation labels with greatest likelihood.

3.1 Base Layer

The current state of the art models as of 2020 features a transformer base layer (e.g. BERT variants) with a variety of layer combinations above it. These models take a sequence of text represented as token or word embeddings as input, and output a sequence of corresponding punctuation labels, similar to other sequence tagging tasks (e.g. Part-of-speech (POS) or Named-Entity Recognition (NER) tagging). Due to the presence of the multi-head attention mechanism along with the positional encoding fed into the transformer, the model should be able to perform well without a Recurrent Neural Network, eliminating the need to process the sentence sequentially, thus improving the speed of training and inference.

Peters et al. (2019) demonstrates that the BERT-CRF model can outpeform the BERT-BiLSTM-CRF model on various NLP tasks when the base layer is unfrozen, and gradual unfreezing of 1 layer per epoch (unfreezing single layers and training for 1 epoch, from the last to the first layer) is necessary to match the performance of frozen BERT-BiLSTM-CRF.

Sunkara et al. (2020b) and Courtland et al. (2020) performed a comparison between various variants of BERT. In general, RoBERTa performs better than BERT, either due to its significantly larger pretraining corpus, or the removal of the Next Sentence Prediction task. Distillation of the model is a possible alternative in the case where training or inference efficiency is more critical than accuracy, being 12% smaller and with a 1.2x inference speed but a 9.1% lower accuracy (Courtland et al., 2020). My experiments observed a similar trend, with RoBERTa base performing the best, followed by Electra-base, and lastly BERT base. I chose to use Electra base for the bulk of my experiments due to its decent performance and its lighter weight than the RoBERTa model.

3.2 Tag decoders

The final stage of punctuation retrieval consists of a decoder which generates a sequence of punctuation labels from a sequence of representations. This is currently done with either a Conditional Random Field (CRF) or a Multi-layer Perceptron (MLP) layer.

3.2.1 Conditional Random Fields

CRFs (Lafferty et al., 2001) are discriminative, undirected Markov models which represents a joint probability distri-

bution of a structured output variable y given an observation x. These are commonly used as an alternative to the MLP head for token labelling tasks, and are shown to perform well in NLP tasks like POS or NER tagging. There have been various examples demonstrating the efficacy of CRF in improving performance on various token classification tasks. Rosvall (2019) observed a slight performance gain when using the CRF layer over BERT rather than a feedforward network on the task of NER.

Variations of the Linear-chain CRF have been used as well. For instance, Lu and Ng (2010) trained the model on punctuation prediction and sentence boundary labelling. They demonstrate the effectiveness of the dynamic CRF in improving performance of punctuation prediction, by learning a sentence boundary tag (e.g. start of question sentence or within declarative sentence) along with the punctuation tags. However, for my project, I kept to the linear-chain CRF to reduce the scope of experimentation.

My experiments were unable to replicate any positive results when using the linear-chain CRF or BiLSTM over the Transformer layer. This might be due to the high degree of imbalance of punctuation labels compared to other sequence tagging tasks such as NER, making the model converge more slowly. Makhija et al. (2019) and Yi et al. (2020a) utilised a BERT-BiLSTM-CRF model, but did not perform a comparison with a BERT with an MLP layer. Since using the dice loss with a MLP layer is more efficient and converges in fewer iterations than the CRF, and I was able to find a set of hyperparameters with the Dice Loss which achieved decent performance on the baseline experiment, I decided to use the MLP layer with Dice Loss for the rest of my experiments.

3.3 Dealing with class imbalance

There are various possible approaches to deal with imbalance across punctuation classes. Most research in punctuation retrieval absorbs the minority classes into a more generic class such as period or comma to improve the problem slightly.

3.3.1 Class weights

The multiplication of the component losses by a weight inversely proportional to the class frequency within the corpus will allow the model to converge faster on the weaker classes in cases of extreme class imbalance. For instance, the weighted CEL is given by:

$$L_{wcel} = -\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M} w_k \times y_m^k \times log(p_{mk}))$$

where w_k represents the class weights

p_{mk} represents the predicted probability of example m being of class k

M represents the number of training examples

K represents the number of classes.

This approach was utilised by Nguyen et al. (2020) in his binary classifier to identify weaker classes. However, Courtland et al. (2020) mentions that their implementation of focal loss or class weights did not outperform the generic cross-entropy loss.

3.3.2 Focal Loss

Yi et al. (2020b) demonstrated the effectiveness of Focal Loss (FL) over Cross-entropy loss (CEL), obtaining a clear improvement both in their controlled experiments and when compared to the Bert Punct (Base) model (Makhija et al., 2019) which uses Cross Entropy Loss. The FL is represented by the following formula:

$$L_{FL} = -\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M} w_k \times y_m^k \times (1-p_{mk})^{\alpha} \times log(p_{mk})$$

where α is a tunable focal parameter. They claim that the focal loss is effective in improving the problem of data imbalance by focusing the training on hard examples. Since the model would often perform better on labels with higher representation, the focal loss would indirectly increase the contribution of labels with lower representation to training, thus improving class imbalance.

3.3.3 Sørensen–Dice coefficient / F1 Score

The paper by Li et al. (2020) featured the idea of directly optimising for the F1 score using the Dice loss (DL). The Sørensen–Dice coefficient (DSC): $DSC(A,B) = \frac{2\times|A\cap B|}{|A|+|B|}$ is used to measure the similarity of the 2 sequences. While there are several inaccuracies in the paper, some of the ideas presented by the paper can be further explored when dealing with class imbalance. The use of the soft macro-averaged F1 score as a metric to be optimised (Dice loss) would allow us to weigh the punctuation loss contribution of different labels individually to account for the class imbalance of the dataset. My implementation of the DL is as follows:

$$L_{DL} = \frac{1}{\sum_k w_k}\sum_{i,k} w_k(1-\frac{2\cdot p_{ik}\cdot y_{ik}+\lambda}{p_{ik}+y_{ik}+\lambda})^{\alpha}$$

where α is the tunable focal parameter

λ is a small smoothing term (fixed as 0.01)

k represents a label

w_k is the label class weight

i represents an example (subword level)

p_{ik} represents the predicted probability of label k being assigned to example i

y_{ik} represents the actual label of example i being equal to k.

My hypothesis is that the α term causes the model to focus on harder punctuation classes since it is applied to each punctuation class loss before the losses of the different punctuation classes are averaged. This is compared to a per example weighting for the focal loss. Instead of assigning a fixed weight based on punctuation proportion, the dice loss assigns a weight based on the aggregate performance of the punctuation class per batch, down-weighing better performing classes such as the blank class and focusing on weaker classes instead. This implementation was inspired by Abraham and Khan (2019).

4 Domain transfer

Various papers have looked into transfer learning across domains.

4.1 Pre-training

Sunkara et al. (2020b) looks at the case of transfer learning to the medical domain which has a significant difference in vocabulary from the source domain used to pretrain transformer models. They demonstrated the effectiveness of further pretraining the BERT model with medical-domain data on the task of Masked Language Modelling to improve performance on punctuation retrieval for medical transcripts. However, this is only effective when there is a low overlap in vocabulary between the source and target domain (e.g. the appearance of medical terms which are not found in the source dataset) which severely impacts the performance on downstream fine-tuning tasks. Since none of the datasets I use contain many out of vocabulary terms when compared to the wikipedia or book corpus dataset which the transformer was pretrained on, I believe the use of pre-training is irrelevant for my project.

4.2 Learning Curriculum

Ma et al. (2019) presents a method to improve domain adaptation, with just unlabelled training examples from the target domain. The method features a BERT classifier which sorts source domain examples in decreasing order of similarity to the target domain. This sorted source domain data is then fed into the learning algorithm, allowing the model to learn examples of increasing difficulty (decreasing similarity), which they demonstrate to improve the model's performance on the target domain.

4.3 Domain Adversarial

Another common method for domain transfer is to add a domain classifier alongside the task classifier, and to use a gradient-reversal layer to maximise the domain invariance for features shared by the task and domain classifier.

Keung et al. (2019) used such a method for cross-lingual zero-resource transfer learning, feeding the mean-pool of the hidden states into the domain classifier. They observed consistent performance gains on the text classification task, but the approach was less effective on the NER labelling task.

However, as mentioned by Li et al. (2019), this approach is effective when domains have similar label distribution, but the model would be unable to capture severe target shifts from the source to unlabelled target domain in the case of zero-resource transfer learning.

Zhou et al. (2019) looks at the case of low-resource transfer learning. Their approach, termed DATnet, uses a similar idea to the domain adversarial method. For their base layer, they use a concatenation of character and word embeddings fed into a BiLSTM layer. For the domain classifier, they use a self-attention layer instead of a mean or max pooling. Unlike other domain adversarial models, they utilise a distinct punctuation classifier for each domain. They observed that using a shared BiLSTM layer led to higher performance, as compared to a separate BiLSTM layer for each domain, and the use of the self-attention layer along with the idea of adaptive weights, similar to focal loss, also led to an improvement in performance. As compared to the above model which only utilises a single task classifier, the use of two separate task classifiers (in their case, CRFs) would better allow the model to capture a target shift from the source to target domain, but this requires some labelled examples from the target domain to work, i.e. it does not work for zero-resource transfer.

5 Method

There are two key areas that I looked at in my experiments — improving performance of a model on the task of punctuation retrieval, and improving performance on zero or low-resource transfer learning. I explored the idea of the Dice Loss with a factor α to focus on harder examples (section 3.3.3) to deal with the punctuation class imbalance. I also looked at the Domain Adversarial model (section 4.3) to deal with zero or low-resource transfer, proposing a modification that led to performance gains in my experiments.

5.1 Data Processing

Table 2 presents the distribution of punctuation for each of the datasets. There are various factors that influenced my approach in the processing of data.

1. The punctuation distribution of the datasets vary slightly across time (e.g. decrease in exclamation for TED, or a slight increase in periods for Open Subtitles) but largely maintains its proportion.

2. There are clear differences in punctuation proportion across all datasets, especially for less common punctuation. For instance, the Open Subtitles corpus feature a substantially larger proportion of (?), (!) and (. . .) than the TED corpus, and the switchboard corpus contains very few (!) and (:). Thus, there would be some deterioration in performance of punctuation labels with less overlap across domains, and transfer learning techniques would likely be necessary to improve the model's cross-domain performance.

3. The Switchboard dataset contains disfluencies and interlocking conversations (e.g. "things like that get . . . Yeah. . . . spread real easy") between two speakers which contributes to the noise of the dataset.

In the pre-processing stage, I explored various ways of representing the punctuated text in a form to be read by the model, and eventually settled on storing the data as normal text with punctuation, only converting it to subword tokens and punctuation labels when training the model. This allowed me to reduce disk usage, easily change characteristics of the dataset such as the punctuation label maps, and easily insert new datasets with less time spent on pre-processing. Further details regarding the pre-processing pipeline can be found in annex A.

After performing the pre-processing step on each of the datasets, I performed a 0.8-0.1-0.1 train-dev-test split in chronological order, followed by a simple transformation to split all documents into shorter examples of around 1500 characters at sentence breaks (.!? ...), to allow each batch to have a similar length for more efficient training. During the training of the model, the train dataset of each domain is shuffled every epoch, and equal number of samples are drawn from each domain, repeating the smaller datasets to align the lengths of each dataset. The samples from all domains are then shuffled and fed into the model. Each batch of around 1500 characters consist of approximately 300-400 sub-word tokens.

Given the huge size of the open subtitles dataset, I used only 4% and 1% of the open subtitles data for training and testing respectively in experiments using all three datasets to reduce the difference in dataset sizes across the datasets and to speed up the training process. To simulate a low-resource target domain, I only used 2 labelled examples from the switchboard dataset, and the other unlabelled switchboard training examples are only used for the domain discriminative component of the model.

5.2 Use of data

5.2.1 Punctuation features

My research only looks at pure textual features with punctuation labels. The task of punctuation retrieval is reduced to a sequence labelling task, with the punctuation label being the last punctuation — in the set of punctuation marks to be predicted — following each word. I performed several experiments comparing the performance of masking the prefix, suffix or none of the subword tokens of each word in the calculation of the Punctuation loss, attaching the labels to each subword token in the case of no mask. Using no mask performed the best of the three, with the added benefit of being more robust to random splits of words. However, while many of my models were able to achieve a high f1 score of over 90 on the No-space label, assigning labels to all subword tokens without any masks would increase the difficulty of the task, removing pre-existing information about spaces in the source text, and leading to a reduction in performance of punctuation labels. Thus, the experiments presented in this paper only looks at suffix tokens in the prediction of punctuation marks, and all prefix tokens are masked.

My experiments predict the four labels: Space, Period (.), Comma (,) and Question (?). The use of a no space label can be included to provide a means to account and train for certain transcription errors in the upstream pipelines (e.g. incorrect splitting or joining of sub-word tokens), but is excluded for my research due to its irrelevance to the problem of domain transfer.

The labels — dash, emdash and colon are replaced by comma, and semicolon, ellipsis and exclamation are replaced by period. Only the last punctuation in the sequence of punctuation after each word is taken as the label, with all prefix sub-word tokens being masked in the calculation of the loss, and all leading punctuation being ignored.

5.3 Model

5.3.1 Punctuation classifier

I experimented with various punctuation classifiers to convert the hidden states into the punctuation labels, evaluating the performance of the MLP, CRF, and the addition of a BiLSTM layer right above the transformer to help in capturing positional information. As mentioned in 2.3, I did not utilise the sliding window with overlapping chunks in my experiments. However, I did use a stride of 0.25 in the training process to increase the number of training examples, which led to an improvement in performance. When using the MLP layer as the punctuation head, I considered the CEL, FL and DL with class weights, as well as the focal parameter α for FL and DL.

My research into transfer learning led me to consider the concatenation of domain features with the transformer hidden states as the input of the punctuation classifier, to allow the model to capture differences in punctuation representation between domains. The domain features I worked with were the domain classifier logits, but a possible alternative would be the output of the first hidden layer of the domain classifier. Unlike the method used by Keung et al. (2019), where the model does not capture the differences between domains, the inclusion of the domain features in punctuation inference can be interpreted as evaluating the similarity of the test example to the different training domains, and prioritising the information learnt from similar domains more. I believe this has some advantage over having a separate classifier per domain as it allows the model to generalise to other domains, and handle domain labels of varying difficulty (some ambiguity between domains).

5.3.2 Domain classifier

The hidden state output of the transformer layer is first passed through a gradient reversal layer to maximise the learning of domain invariant features within shared layers. These outputs are then pooled to obtain a representation per sentence containing information across all tokens within the sentence. I considered a variety of approaches for the pooling of the hidden states for the domain classifier. This includes a mean / max / concatenation of mean and max pooling of the hidden states of each sentence into a single tensor (i.e. batch-size x sequence-length x hidden-states $\rightarrow$ batch-size x hidden-states for mean or max pooling). I also considered a self-attention layer as utilised by Zhou et al. (2019), as well as no pooling, where the domain classifier considers each token separately. For the pooled outputs, The resulting tensor is then repeated along the second dimension before being concatenated with the original hidden states to be fed into the punctuation classifier.

5.3.3 Loss

After obtaining the punctuation and domain predictions in the form of soft probabilities, they are both evaluated with a loss function, with the punctuation loss function being one of CEL, FL and DL in the case of the MLP or the Negative log-likelihood for CRF, and the domain loss function being one of CEL, FL and DL. The domain loss is then

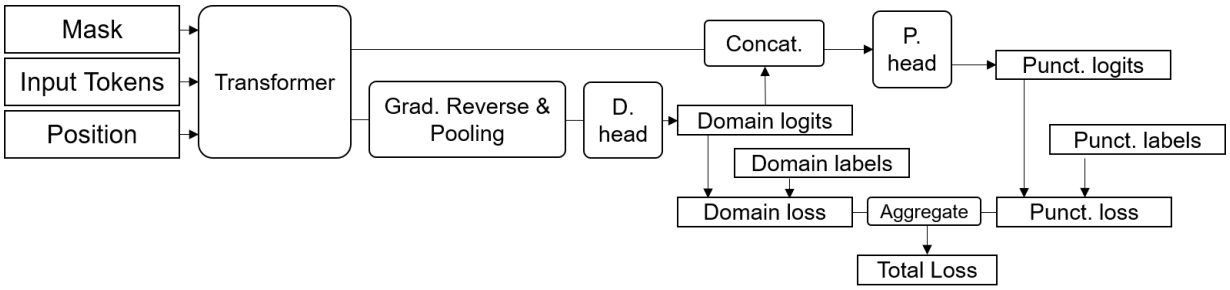

Figure 1: Proposed model using domain features in punctuation classifier

multiplied by a factor dependant on the current punctuation loss and the training step number, to prevent either of the losses from overwhelming the training. The exact formula used in the calculation of the aggregate loss of the punctuation and domain loss is as follows:

$$L_{total(n)} = L_{p(n)} + \left(\frac{2}{1 + e^{-10n/N}} - 1\right) \cdot \gamma \cdot L_{p(n-1)} \cdot L_{d(n)}$$

where γ is the contribution ratio of the domain loss
n is the current step number within the epoch
N is the total step count of each epoch
$p(n)$ is the punctuation loss at step n
$d(n)$ is the domain loss at step n

5.3.4 Class Imbalance

Several components in my model were used to improve class imbalance of punctuation labels. The use of the macro-averaged DL mentioned above in section 3.3.3 gives equal contribution to each punctuation label. I also worked with punctuation class weights when working with the CEL, FL and DL, formulating the weights based on the following formula:

$$w_k = \frac{n_k}{\sum_{i=1}^{K} n_i}^{\gamma}$$

where K is the number of punctuation labels and γ is a tunable parameter between 0 and 1 inclusive. For the domain imbalance, I made use of oversampling to obtain an equal representation of samples from each domain by repeating the entire training set from underrepresented domains. I also considered the use of class weights for the domain classifier to increase the loss contribution of low resource domains with the aim of increasing the contribution of high resource domains in inference.

5.3.5 Training procedure

Initially, I tested different combinations of parameters with the transformer layers frozen, and the top layer unfreezed after 8 epochs. However, the performance of these experiments did not come close to the existing published research, and like many of the existing research conducted, training with the entire transformer unfrozen led to improved performance. I then proceeded to train for 4 epochs with 6 layers of the transformer unfrozen from the start, and it came closer to the published results on punctuation retrieval for the baseline iwslt task. For the rest of my experiments, I kept to the following settings:

1. Use of Electra-base as transformer
   6 tranformer layers unfrozen
2. 2 MLP layers for Punctuation classifier
   Dice Loss with $\alpha = 2$ and class weight of 0
   Dropout of 0.2
3. 2 MLP layers for Domain classifier
   Total loss contribution of 0.005
   Dice Loss with $\alpha = 4$
   Dropout of 0.6
4. Stride of 0.25 (32 tokens) to increase training examples
5. Random swap of sentences denoted by sentence boundaries (. ? ! ...)
6. Batch size of 4, learning rate of 4e-05
7. Accumulate gradient every 2 batches
8. Sequence length of 128
9. Differential learning rate with each transformer layer being 0.8 times the layer above it.

I then proceeded to test various combinations of the following additions to the model:

1. Concatenation of Domain logits to hidden state input to Punctuation classifier (concat)
2. Use of domain discriminator (adversarial)

The various parameters such as the learning rate, the unfreezing strategy, the dropout rate, the dice loss α value for the domain classifier, the loss contribution of the domain classifier or the augmentation rates can be further tuned for better performance.

6 Results and discussion

I obtained some results demonstrating the effectiveness of the various additions to the model for Punctuation Retrieval.

Firstly, I trained and tested a baseline model on just the IWSLT 2012 dataset, and another on just the TED Talks dataset (Table 1 r1, 2). Both models performed similarly since they are both from the TED talks domain, but

did not outperform existing published results on punctuation retrieval. For instance, Alam et al. (2020) achieved a overall f1 score of 76.9 with the BERT-base-uncased transformer, and Nagy et al. (2021) achieved 79.8, as compared to the 67.6 in my experiment (Table 1 r7). This difference in performance might be due to several factors: training with fewer layers (6 instead of 12) unfrozen, the use of a MLP layer instead of a CRF for the punctuation classifier, or the lack of the use of a sliding window to improve performance on boundary tokens with less contextual information. However, due to time constraints, I kept to my configuration to allow for faster experimentation, where each epoch for the TED baseline example without stride takes 2 minutes to complete.

The use of a stride length of 32 on training data resulted in an increase in performance of the base model, likely due to the model being exposed to a greater number of training examples (Table 1 r2, 4). However, augmentation using random insertion, deletion and substitution of tokens with a probability of 0.2 each resulted in a drop in performance of the base model (Table 1 r2, 3). This might be due to the high rate of augmentation used resulting in the loss of important tokens. Possible developments include varying the augmentation rate and implementing rules when performing the augmentation, such as the use of synonyms or homophones for substitution.

Pooling choice for domain classifier. All methods of pooling were able to lead to an improvement in transfer performance over the baseline model, but using the attention pooling or CLS token to capture the domain information of each example performed better in my experiment (Table 1 r11, 14). Further experiments can be conducted to explore the difference between both pooling strategies.

Class weights. The use of class weights were ineffective in the case of three punctuation classes – period, comma and question, when training using the Dice Loss (Table 1 r5). However, when using the Cross-entropy loss, the use of class weights was necessary to overcome the significantly higher proportion of the space class (Table 1 r6). This suggests that the Macro-averaged dice loss is able to handle the class imbalance well. In experiments predicting punctuation labels with significant imbalance such as the exclamation mark, increasing the gamma factor of the class weights slightly is able to improve the performance on the rare classes, but more research has to be conducted to observe its impact on the more common labels.

Choice of transformers. All the results I obtained, apart from the three with different transformers, were conducted using the pre-trained Electra base discriminator. As expected, using a transformer with greater depth and pretrained on a larger number of examples leads to higher performance (Table 1 r7,8,9). My attempts using the Electra small discriminator were unsuccessful, being unable to predict any question marks.

Use of domain logits in punctuation classifier or domain discriminator. The concatenation of domain logits with hidden states and the use of the domain discriminator both proved effective in improving transfer learning performance. This can be observed in the low-resource transfer experiments using the TED + 2 labelled Switchboard dataset, with the overall F1 score increasing from 52.6 to 57.2 when using the token pool with the domain discriminator for the domain classifier (Table 1 r10, 14). Ablation studies performed on both the discriminator and the concatenation of domain features show that both factors are important. It also suggests that the discriminator is more important in improving the performance on the low resource domain (Table 1 r12, 13), but more experiments have to be conducted to better understand the importance of each component in improving the performance of low-resource transfer learning.

Performance on zero-resource domain I tested two variants of zero-shot learning with both components – concatenation of the domain logits and use of the domain discriminator, the first only using two high-resource domains — Ted talks and Open subtitles (Table 1 r17), and the second with the addition of unlabelled switchboard domain examples (Table 1 r18). Both models performed better than the baseline with no domain consideration (Table 1 r16). The model with additional unlabelled zero-resource domain examples also saw a slight improvement in zero-resource domain performance from a f1 score of 53.1 to 53.8, but more experiments have to be conducted to support this observation.

6.1 Interpretation of results

The two added components — backpropagating the negative gradient of the domain classifier and taking the domain logits as input into the punctuation classifier can be observed to improve the performance of both the high, and low-resource domains, with the greatest improvements observed in rarer punctuation classes (in this case the question mark).

Using either component alone also leads to a decent improvement over the baseline's performance, with the domain adversarial component appearing to be of greater importance than the concatenation component.

The use of the CLS token or the attention pooling for the domain classifier has similar performance in my results, so more experiments have to be conducted to compare the two methods of pooling.

My last experiment conducted on the expanded punctuation set suggests that other punctuation marks such as the hyphen or colon can potentially be considered in the punctuation retrieval pipeline. The better performance of the open subtitles dataset as compared to the other two datasets is possibly due to the inadequacy of oversampling in dealing with the domain imbalance, and future research can consider tuning the domain class weights or α for the Dice Loss to handle this issue.

7 Future Improvements

Here is a list of possible ideas that can be considered in future experiments.

1. Use of a sliding window with aggregation of predictions for overlapping tokens. Can consider the idea of the Gaussian Filter for each prediction window when

performing inference as a method of aggregation to minimise labelling errors at window boundaries.

2. Attempt to get a working retrieval model using the CRF classifier, and perform the same experiments using that baseline
3. Consider concatenating different domain representations with the hidden states, for instance the first layer hidden states of the domain classifier instead of the domain logits.
4. Perform a study on the effect of the α term for the Dice Loss on the performance of the model and its robustness to class imbalance. Find out how its performance differs from the use of class weights.

Appendices

The accompanying code for this project can be found here.

I am grateful to my mentor, Chieu Hai Leong, for his guidance throughout the course of this project

A Preprocessing Pipeline

The following preprocessing steps are taken to clean up each raw dataset

1. Remove speaker tags or tags that are added for better readability, such as sound effects e.g. (Narrator:) or (Applause).
2. Identify spoken text that are within square or round brackets and remove the brackets.
3. Remove music lyrics bounded by music note symbols as they contain minimal punctuation information.
4. Remove empty matching tags: square brackets, parentheses, single or double quotes as they are not covered under the scope of this research.
5. Identify and convert ellipsis to Unicode.
6. Remove Non-sentence punctuation (All punctuation that are non-readable (@$#%&+•= €²£¥) subtract (. ? ! , ; : - – —).
7. Replace en-dash with hyphen
8. Combining repeated patterns of punctuation, i.e. [. .] to [.], [!!!!] to [!].
9. Pronounce common symbols such as * to times, or 5.0 to 5 point 0
10. Remove excess white-spaces
11. Sort the entire corpus by chronological order
12. Remove duplicate examples
13. Perform train development test split of 0.8 0.1 0.1.
14. Split large document into smaller size (i.e. split large document into smaller chunks of 1500 characters) to reduce the size difference of examples.

References

Abraham, N. and Khan, N. M. (2019). A novel focal tversky loss function with improved attention u-net for lesion segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 683–687. IEEE.

Alam, T., Khan, A., and Alam, F. (2020). Punctuation restoration using transformer models for high-and low-resource languages. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 132–142, Online. Association for Computational Linguistics.

Courtland, M., Faulkner, A., and McElvain, G. (2020). Efficient automatic punctuation restoration using bidirectional transformers with robust inference. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 272–279, Online. Association for Computational Linguistics.

Keung, P., Lu, Y., and Bhardwaj, V. (2019). Adversarial learning with contextual embeddings for zero-resource cross-lingual classification and ner. *arXiv preprint arXiv:1909.00153*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *MACHINE LEARNING : INTERNATIONAL WORKSHOP THEN CONFERENCE*, page 282—289. Citeseer.

Li, X., Sun, X., Meng, Y., Liang, J., Wu, F., and Li, J. (2020). Dice loss for data-imbalanced nlp tasks.

Li, Y., Murias, M., Major, S., Dawson, G., and Carlson, D. (2019). On target shift in adversarial domain adaptation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 616–625. PMLR.

Lu, W. and Ng, H. (2010). Better punctuation prediction with dynamic conditional random fields.

Ma, X., Xu, P., Wang, Z., Nallapati, R., and Xiang, B. (2019). Domain adaptation with bert-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 76–83.

Makhija, K., Ho, T., and Chng, E. (2019). Transfer Learning for Punctuation Prediction. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 268–273. Code available at `https://github.com/panda-baba/bert_punct`.

Nagy, A., Bial, B., and Ács, J. (2021). Automatic punctuation restoration with bert models. *arXiv preprint arXiv:2101.07343*.

Nguyen, B., Nguyen, V. B. H., Nguyen, H., Phuong, P. N., Nguyen, T.-L., Do, Q. T., and Mai, L. C. (2019). Fast and accurate capitalization and punctuation for automatic speech recognition using transformer and chunk merging. *2019 22nd Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*.

Nguyen, T., Nguyen, D., and Rao, P. (2020). Adaptive name entity recognition under highly unbalanced data. *arXiv preprint arXiv:2003.10296*.

Peters, M. E., Ruder, S., and Smith, N. A. (2019). To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Rosvall, E. (2019). Comparison of sequence classification techniques with bert for named entity recognition.

Sunkara, M., Ronanki, S., Bekal, D., Bodapati, S., and Kirchhoff, K. (2020a). Multimodal semi-supervised learning framework for punctuation prediction in conversational speech.

Sunkara, M., Ronanki, S., Dixit, K., Bodapati, S., and Kirchhoff, K. (2020b). Robust prediction of punctuation and truecasing for medical asr.

Yi, J., Tao, J., Bai, Y., Tian, Z., and Fan, C. (2020a). Adversarial transfer learning for punctuation restoration. *ArXiv*, abs/2004.00248.

Yi, J., Tao, J., Tian, Z., Bai, Y., and Fan, C. (2020b). Focal Loss for Punctuation Prediction. In *Proc. Interspeech 2020*, pages 721–725.

Zhou, J. T., Zhang, H., Jin, D., Zhu, H., Fang, M., Goh, R. S. M., and Kwok, K. (2019). Dual adversarial neural transfer for low-resource named entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3461–3471.

	Experiment	Test	Comma			Period			Question			Overall		
			P	R	F	P	R	F	P	R	F	P	R	F
Electra base discriminator with 2 layer MLP punctuation classifier														
1	IWSLT baseline	IWSLT	69.1	74.0	71.5	79.9	87.7	83.6	63.9	84.0	72.6	71.0	81.9	75.9
	2021-03-26_15-57-24	SWI	64.7	61.2	62.9	51.9	65.7	58.0	16.8	66.7	26.9	44.5	64.5	49.3
2	TED baseline	TED	67.6	78.6	72.7	82.6	86.0	84.3	65.2	83.1	73.1	71.8	82.6	76.7
	2021-03-26_16-35-29	SWI	65.5	61.1	63.2	49.9	64.9	56.4	12.8	69.3	21.6	42.8	65.1	47.1
3	+augmentation	TED	69.1	76.7	72.7	82.0	86.2	84.1	61.6	84.1	71.1	70.9	82.4	76.0
	2021-03-26_15-32-09	SWI	64.9	53.8	58.8	50.5	57.9	53.9	9.5	71.6	16.7	41.6	61.1	43.2
4	− stride	TED	66.2	79.0	72.0	83.5	83.5	83.5	63.3	81.5	71.2	71.0	81.3	75.6
	2021-03-26_17-52-48	SWI	64.2	64.3	64.3	52.0	59.5	55.5	14.8	66.1	24.1	43.7	63.3	48.0
5	Class weight 0.2	TED	66.6	78.4	72.0	81.4	86.5	83.8	59.2	86.6	70.3	69.0	83.8	75.4
	2021-03-29_10-18-31	SWI	63.4	57.3	60.2	51.8	63.2	57.0	9.9	78.0	17.6	41.7	66.2	44.9
6	TED baseline using CEL	TED	55.9	84.1	67.1	77.3	85.7	81.3	35.2	92.4	51.0	56.1	87.4	66.5
	2021-03-29_12-14-21	SWI	57.8	61.6	59.6	48.6	54.9	51.6	5.9	87.1	11.0	37.4	67.9	40.7
Different transformers with TED baseline														
7	BERT base uncased	TED	60.8	70.0	65.1	74.6	78.8	76.7	52.2	73.8	61.2	62.6	74.2	67.6
	2021-03-26_23-01-13	SWI	58.7	56.2	57.5	43.8	52.8	47.9	17.5	59.5	27.0	40.0	56.2	44.1
8	Distilbert base uncased	TED	58.1	67.4	62.4	71.4	75.9	73.6	45.4	75.2	56.6	58.3	72.8	64.2
	2021-03-27_14-09-56	SWI	59.1	59.0	59.0	41.8	46.6	44.0	14.5	60.0	23.4	38.4	55.2	42.2
9	Electra large	TED	69.8	80.6	74.9	85.4	88.1	86.8	69.1	87.1	77.1	74.8	85.3	79.6
	2021-03-27_15-13-29	SWI	66.9	64.2	65.5	56.9	70.4	63.0	14.1	74.9	23.7	46.0	69.8	50.7
Domain adversarial / concat logits baseline														
10	TED+2 SWI	TED	70.3	73.6	71.9	80.3	88.0	84.0	62.4	84.8	71.9	71.0	82.1	75.9
	2021-03-27_07-41-03	SWI	69.5	67.1	68.3	55.9	72.2	63.0	16.3	69.3	26.4	47.2	69.5	52.6
Domain adversarial / concat domain logits (TED+2 labelled SWI + unlabelled (UL) SWI)														
11	Using Attn. pool	TED	71.1	73.4	72.2	79.4	89.2	84.0	66.9	83.0	74.1	72.5	81.9	76.8
	2021-03-27_08-30-38	SWI	68.1	72.9	70.4	58.7	68.6	63.2	26.7	62.6	37.4	51.2	68.0	57.0
12	- Concat	TED	72.6	70.7	71.6	78.4	90.4	84.0	66.8	83.6	74.2	72.6	81.6	76.6
	2021-03-27_10-40-44	SWI	69.7	69.4	69.6	57.4	71.8	63.8	25.9	64.2	36.9	51.0	68.5	56.8
13	- Adversarial	TED	71.1	73.4	72.3	80.1	88.6	84.2	62.8	84.4	72.0	71.4	82.1	76.2
	2021-03-27_09-45-43	SWI	68.8	70.9	69.8	58.6	71.1	64.3	24.0	65.9	35.2	50.5	69.3	56.4
14	Using Token pool	TED	71.3	72.7	72.0	78.6	90.0	84.0	66.5	83.0	73.9	72.2	81.9	76.6
	2021-03-27_10-42-57	SWI	68.7	71.2	69.9	57.8	71.9	64.1	26.6	63.6	37.5	51.0	68.9	57.2
15	Using Mean pool	TED	71.4	72.8	72.1	79.7	89.1	84.1	64.7	84.1	73.2	72.0	82.0	76.5
	2021-03-27_11-44-59	SWI	69.0	70.7	69.9	57.6	70.5	63.4	23.3	65.6	34.4	50.0	69.0	55.9
Zero-shot														
16	TED+OPE baseline	OPE	64.0	70.7	67.2	78.3	80.7	79.5	65.6	72.5	68.9	69.3	74.6	71.8
	2021-03-26_22-37-11	TED	67.2	78.9	72.6	83.6	83.5	83.6	70.0	82.8	75.9	73.6	81.7	77.3
		SWI	67.7	69.2	68.5	52.5	65.1	58.1	19.9	73.5	31.3	46.7	69.3	52.6
17	TED+OPE both	OPE	63.6	71.7	67.4	78.7	80.4	79.5	66.4	72.3	69.2	69.6	74.8	72.0
	2021-03-26_23-28-33	TED	66.8	79.4	72.5	83.9	83.3	83.6	71.4	82.4	76.5	74.0	81.7	77.5
		SWI	67.2	69.9	68.5	53.5	65.1	58.7	20.4	73.8	32.0	47.1	69.6	53.1
18	TED+OPE+UL SWI	OPE	67.0	64.4	65.7	76.1	82.8	79.3	62.1	73.7	67.4	68.4	73.6	70.8
	2021-03-27_11-38-32	TED	70.3	73.6	71.9	80.8	86.7	83.7	66.0	84.0	73.9	72.4	81.5	76.5
		SWI	71.2	67.0	69.1	54.8	71.8	62.1	18.9	74.3	30.2	48.3	71.0	53.8
Model trained on TED+OPE+SWI														
19	Middle 0.5 chunk TED	Full	69.7	74.7	72.1	81.2	86.7	83.9	70.9	82.4	76.2	73.9	81.3	77.4
	2021-03-27_18-00-46	Middle	70.1	76.0	72.9	83.7	87.8	85.7	71.7	86.3	78.3	75.2	83.4	79.0

Expanded punctuation set										
			!	,	-	.	:	?	...	F1
20	TED+OPE+SWI	OPE	22.8	49.2	41.0	61.1	65.5	62.7	23.3	46.5
	2021-03-28_11-56-37	TED	11.0	65.5	60.7	78.5	2.9	68.1	3.5	41.5
		SWI	0.00	67.8	88.4	43.0	0.0	48.9	29.0	39.6

Table 1: Study of effect of different parameters on performance of transfer to Switchboard dataset
SWI: Switchboard, OPE: Open Subtitles, TED: Ted Talks, TED+2 SWI: TED with 2 labelled examples from SWI, both: Concat + Adversarial, baseline: no domain consideration, UL SWI: unlabelled SWI train examples
View the hyperparamaters and results by searching for the listed dates: here.

Count of punctuation labels per dataset												
Dataset	IWSLT 2012			Ted			Open Subtitles			Switchboard		
label	train	dev	test	train	dev	test	train	dev	test	train	dev	test
,	162729	1124	2230	391670	45595	60974	24519706	3001818	2921821	192948	17282	25926
.	143020	923	1684	317959	39220	51692	47812347	5618205	5333118	94456	8163	13050
—	18295	106	46	25984	2799	4591	9077	1297	769	0	0	0
-	12298	57	94	29947	3717	4313	2235061	275756	265933	20789	1956	3285
?	11813	84	148	29134	3178	4462	13496823	1530633	1376747	6715	714	781
:	2763	12	20	7733	795	1186	286415	17439	24463	23	15	4
;	2845	6	18	4342	447	1057	60723	6072	5017	32	0	7
!	926	14	2	2742	241	298	6833770	692888	556099	75	1	11
...	603	1	9	1680	175	274	4237147	468868	457918	18307	2685	2761
,	0.46	0.48	0.52	0.48	0.47	0.47	0.25	0.26	0.27	0.58	0.56	0.57
.	0.40	0.40	0.40	0.39	0.41	0.40	0.48	0.48	0.49	0.28	0.26	0.288
—	0.05	0.05	0.01	0.03	0.03	0.04	0.00	0.00	0.00	0.00	0.00	0.00
-	0.03	0.02	0.02	0.04	0.04	0.03	0.02	0.02	0.02	0.06	0.06	0.07
?	0.03	0.04	0.03	0.04	0.03	0.03	0.14	0.13	0.13	0.02	0.02	0.02
:	0.01	0.01	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00
;	0.01	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
!	0.00	0.01	0.00	0.00	0.00	0.00	0.07	0.06	0.05	0.00	0.00	0.00
...	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.04	0.05	0.09	0.06

Table 2: Distribution of punctuation labels.