Efficient Automatic Punctuation Restoration Using Bidirectional Transformers with Robust Inference

Maury Courtland, Adam Faulkner, Gayle McElvain

Capital One, Vision and Language Technologies

Overview

Introduction

Results

Ablation Studies

Overview

Introduction

Results

Ablation Studies

Background: ASR + NLP Is Powerful

Medical dictation transcription and analysis

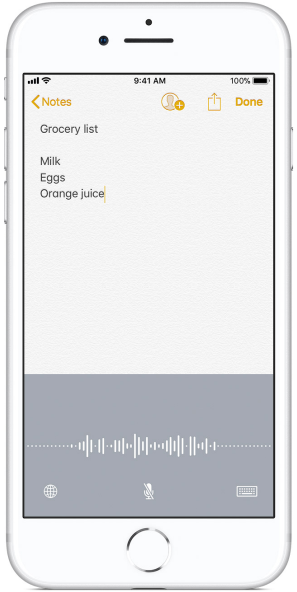
Real-time spoken language translation

Digital personal assistants

Background: ASR + Humans Is Important Too

Automatic video captioning

Automatic message transcription



Problem: ASR Outputs Just Tokens

Models

- For downstream NLP models, the lack of clausal boundaries can significantly decrease accuracy (e.g. a 4.6% BLEU decrease in NMT)[1]
- Probably because of the mismatch between training corpora and ASR output

Humans

- No punctuation can be as detrimental as 15-20% WER[2]
- Reading comprehension is also significantly slowed[3]
- Likely due to the removal of phrasing cues we rely on

[1]Vandeghinste et al., 2018. A Comparison of Different Punctuation Prediction Approaches in a Translation Context.
[2]Tündik et al., 2018. User-centric Evaluation of Automatic Punctuation in ASR Closed Captioning.
[3]Jones et al., 2003. Measuring the readability of automatic speech-to-text transcripts.

Design Philosophy

As implementing practitioners, we don’t always have:

- Control over the ASR model (vendor / different team)
- Budget for task-specific training data (leverage transfer learning?)
- The time to squeeze performance out of optimizations in order to run at scale

Solution: Automatic Punctuation

Our solution leverages (relatively novelly):

1. Bidirectional transformer architecture (i.e. BERT & Co.)
2. Pre-trained LMs at its core
3. State of the art optimizers

We also introduce (a.k.a. *necessity* + *running* = *...*):

4. Simultaneous predictions for all words in an input window
5. Prediction aggregation across multiple context windows

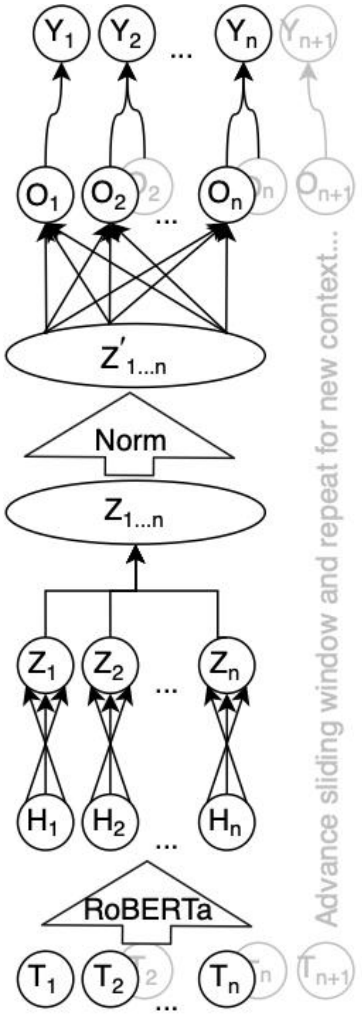
Overview

Introduction

Results

Ablation Studies

Technical Details: Architecture



Punctuation Labels

Aggregate Across Contexts

Sequence distributions

Linear Classifier Layer

Normalized sequence vector

Batch Norm and Dropout

Hidden sequence vector

Concatenate

Hidden token vectors

Linear Layer

Vocabulary distributions (before Fine-Tuning)

Pre-trained LM

Vocab IDs

Technical Details: Training & Inference

Training

- Gradient descent on cross-entropy of 4-class classification
  - Classes: 85.7% *no punctuation*, 7.53% *comma*, 6.3% *period*, 0.47% *question mark*
  - Optimization performed by LookAhead optimizer with RAdam as base optimizer
  - Training performed on individual window predictions (***no*** aggregation)
- 1st stage keeps pre-trained LM frozen, 2nd stage fine-tunes entire network

Inference

- Pool individual predictions from different overlapping windows

NOTE: Our network's hyperparameters have *not* been rigorously tuned in any principled way (e.g. grid search, bayesian optimization)

Result 1: Better Accuracy

48.7% relative improvement (overall F1, 15.3 absolute), likely due to:

1. More powerful architecture (transformers' self-attention)
2. Direct connections between long-distance dependencies (e.g. "wh" words)
3. Self-ensemble effect of aggregating across multiple context windows

	Comma			Period			Question			Overall		
Models	P	R	F1	P	R	F1	P	R	F1	P	R	F1
DNN-A (Che et al., 2016)	48.6	42.4	45.3	59.7	68.3	63.7	—	—	—	54.8	53.6	54.2
CNN-2A (Che et al., 2016)	48.1	44.5	46.2	57.6	69.0	62.8	—	—	—	53.4	55.0	54.2
T-LSTM (Tilk and Alumäe, 2015)	49.6	41.4	45.1	60.2	53.4	56.6	57.1	43.5	49.4	55.0	47.2	50.8
T-BRNN (Tilk and Alumäe, 2016)	64.4	45.2	53.1	72.3	71.5	71.9	67.5	58.7	62.8	68.9	58.1	63.1
T-BRNN-pre (Tilk and Alumäe, 2016)	65.5	47.1	54.8	73.3	72.5	72.9	70.7	63.0	66.7	70.0	59.7	64.4
Single-BiRNN (Pahuja et al., 2017)	62.2	47.7	54.0	74.6	72.1	73.4	67.5	52.9	59.3	69.2	59.8	64.2
Corr-BiRNN (Pahuja et al., 2017)	60.9	52.4	56.4	75.3	70.8	73.0	70.7	56.9	63.0	68.6	61.6	64.9
DRNN-LWMA (Kim, 2019)	63.4	55.7	59.3	76.0	73.5	74.7	75.0	71.7	73.3	70.0	64.6	67.2
DRNN-LWMA-pre (Kim, 2019)	62.9	60.8	61.9	77.3	73.7	75.5	69.6	69.6	69.6	69.9	67.2	68.6
RoBERTa$_{base}$ (Ours)	**76.9**	**75.4**	**76.2**	**86.1**	**89.3**	**87.7**	**88.9**	**87.0**	**87.9**	**84.0**	**83.9**	**83.9**

Result 2: Faster Inference

1.2x CPU speedup, 78.8x GPU speedup[1]

Likely due to:

1. Non-recurrent network computations allow for huge parallelization
2. Simultaneous token prediction saves lots of computations
3. Multiple aggregated predictions are independent (i.e. also parallelizable)

[1]Single predictions per token compared with most recent open source SOTA (best # predictions is 41.5x GPU speedup):

Tilk & Alumäe. 2016. Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration.

Result 3: Faster Training

SOTA model in ~1 hour on 3.16xlarge = ~$24

1. Leverages massive pre-training corpus (~33B words)
2. Allows SOTA training on just TED corpus (2.1M words -- IWSLT 2012 challenge)
3. Richer training signal given simultaneous token prediction

Overview

Introduction

Results

Ablation Studies

Ablation: Inference Parallelism

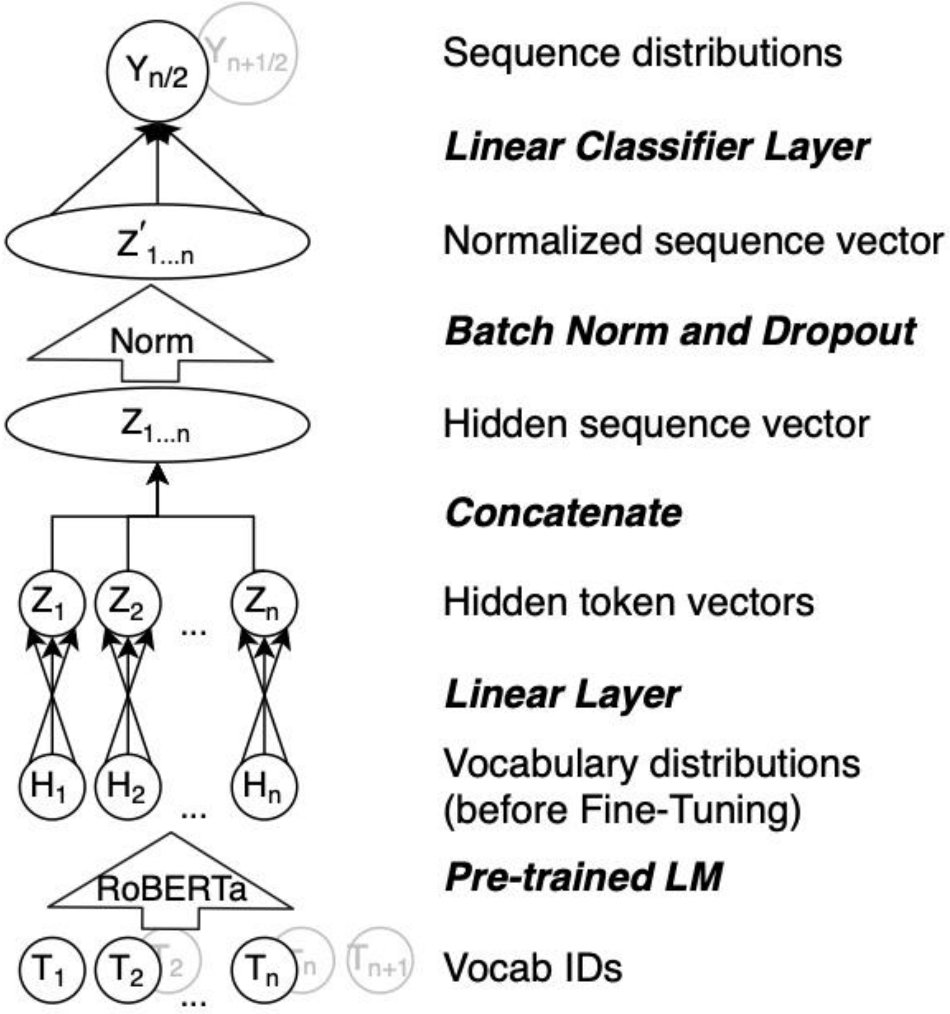
Same input, advancing window by 1 and predict for each word in sequence

Without ensembling, our network is:

- 15x faster, ***but*** 2.2% worse

With ensembling, our network is:

- 4x faster, ***and*** 5.4% better



Ablation: Speed/accuracy tradeoff

Predictions per token	F1 Overall	CPU Runtime	GPU Runtime
1	76.3	1x	1x
2	79.4	1.8x	1.1x
3	81.8	2.6x	1.2x
6	83.2	5.2x	1.5x
9	83.9	7.7x	1.9x
Processor	—	18x Intel Xeon (c5.18xlarge)	8x Tesla V100 (p3.16xlarge)

Ablation: Pre-train Trained LM

- RoBERTa$_{lg}$ not better → RoBERTa$_{base}$ is adequately parameterized
- Several SOTA pre-trained models are slightly worse (XLNet, T5, BERT)
  - RoBERTa > BERT likely due to training corpus size
- Sacrificing parameters sacrifices performance (ALBERT & DistilRoBERTa)

Models	Comma			Period			Question			Overall		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RoBERTa$_{base}$	**76.9**	75.4	**76.2**	**86.1**	89.3	87.7	**88.9**	87.0	**87.9**	**84.0**	83.9	**83.9**
RoBERTa$_{large}$	74.3	76.9	75.5	85.8	**91.6**	**88.6**	83.7	89.1	86.3	81.3	**85.9**	83.5
XLNet$_{base}$	76.6	74.9	75.8	84.6	90.6	87.5	82.0	89.1	85.4	81.1	84.9	82.9
T5$_{base}$	70.5	**77.2**	73.7	85.6	85.5	85.6	83.7	89.1	86.3	79.9	84.0	81.9
BERT$_{base}$	72.8	70.8	71.8	81.9	86.6	84.2	80.8	**91.3**	85.7	78.5	82.9	80.6
ALBERT$_{base}$	69.4	69.3	69.4	80.9	84.5	82.7	76.7	71.7	74.2	75.7	75.2	75.4
DistilRoBERTa	70.0	64.5	67.1	78.2	83.5	80.8	75.0	71.7	73.3	74.4	73.2	73.7

Ablation: Optimizer + Loss Function

Optimizer	Overall F1 Change
LookAhead + RAdam	0%
LookAhead + Adam	-1.5%
RAdam alone	-1.6%
Adam alone	-2.9%

Class imbalance "fixes"	Overall F1 Change
Cross-entropy class weighting	-7.3%
Focal loss	-3.9%
Class weighting + Focal loss	-7.3%

Conclusion

Our approach:

- Is 40x faster for inference (on GPUs)
- Achieves a 48.7% relative improvement (overall F1)
- Can be trained in ~1 hour for $24 (on AWS p3.16xlarge)

Next steps:

- Obtain in-domain human agreement metric (i.e. competitiveness w/ humans)
- Rigorous hyperparameter tuning
- Perform linguistic analyses (e.g. long-distance dependencies)
- Measure our model's benefits on downstream users and systems