Apostrophe Prediction on Sentences

Tuğrul Hasan Karabulut
Boğaziçi University
tugrul.karabulut@boun.edu.tr

Abstract

As deep learning applications for natural language processing have become prevalent, results on lots of crucial tasks like text classification, question answering, spelling correction, grammar correction have been improved dramatically. In this study, I investigate ways to predict (or restore) apostrophes, single/double quotes in a sentence. Experimentation on several models like Long Short-Term Memory (LSTM) and more recent Transformer and BERT-based models and different input/output representations have been carried out and results are reported in this paper.

1 Introduction

Usage of apostrophes is crucial in English and many other languages. There is one clear difference between apostrophe and other punctuation marks like comma, semicolon, etc.: apostrophe is usually in between the letters of a word. This makes it hard to detect incorrect usages. Some examples like "I'm", "He's", "They're" can be detected by some predefined rule set but there are lots of examples that require understanding of the context to correctly predict if an apostrophe is missing or not. One example is difference between "it's" and "its". While the former corresponds to "it is", the latter indicates possession. Also, possession form of proper nouns like "John's" requires context knowledge.

Other characters of interest for this study are single and double quotes. Quotes are used for various purposes. First usage that comes to mind is referencing a sentence, or part of a sentence, from a source, which can be a person, book or any other thing. Also, we can use quotes to indicate an uncommon usage of a word or an expression. This uncommon usage would make sense only in that context. There is no clear rule for when to use single quote or double quote. For example, it has been said that British and American people use it differently [13]. In any case, both usages can be seen as correct if it is clear that using quotes in a particular part of the sentence make sense.

Apostrophe, single quote and double quote characters are similar in terms of how they written and sometimes, grammar. They can be confusing and often misused, or unused by people. Just like spelling correction, punctuation restoration and similar tasks, it would be useful to, given a sentence, predict where to put one or more of these characters in that sentence. This could be useful for both writing tools, social media and also for post-processing for automatic speech recognition systems' output.

For the rest of this paper, I call this task "apostrophe prediction" but it refers to both apostrophe and quote characters.

In this study, I develop and compare several sequential deep learning models on apostrophe prediction task. I approach this problem as a **sequence labeling** task. Sequence labeling is the type of task that assigns a label to each element of the input sequence. Labels are from a predefined set of categories. There are lots of applications in natural language processing that conforms to this definition. For example, one of most common NLP applications is **named-entity recognition (NER)**. NER is the

task of recognizing special entities in a text like people, location, organization etc. Another sequence labeling examples include part-of-speech (POS) tagging, punctuation restoration.

2 Previous Work

In this section, I outline several work on related tasks. There seems to no study specifically for the problem at hand but there are works that tackle punctuation restoration, quotation recommendation and similar problems.

Tilk and Alumae (2016) proposed an architecture, for punctuation restoration, that were popular before transformers, which was a bidirectional recurrent neural network utilizing an attention mechanism [12]. Recurrent neural network (RNN) is a type of neural network that is used on temporal data. It is used for NLP, time series analysis and similar type of research fields. An RNN consists of sequential nodes. At time step t, given an input vector $\vec{x}_t$ and previous hidden state $\vec{h}_{t-1}$, it produces the current hidden state $\vec{h}_t$ and output vector $\vec{y}_t$. Hidden state is used to provide information flow across the sequence. Output vector at every time step can be used to do prediction for time step t. Most basic RNN formulation is as follows:

$$\vec{h}_t = \sigma_h(W_h\vec{h}_{t-1} + W_x\vec{x}_{t-1} + \vec{b}_h)$$

$$\vec{y}_t = \sigma_y(W_y\vec{h}_t + \vec{b}_y)$$

where W_h, W_x, W_y are weight matrices, $\vec{b}_h$ and $\vec{b}_h$ are bias vectors, σ_h and σ_y are non-linear activation functions. Weights and biases are learned through training.

Note that output vectors are used only if we work on sequence labeling, text generation or similar tasks. If our goal was to generate a prediction for whole sequence, we could use just the last output. As example task for this could be sentiment analysis. Further to this simple RNN architecture, there are more complex RNN variants like LSTM and GRU. These models are better at handling long-term dependencies. They have more computationally complex and have more parameters. For example, Tilk and Alumae (2016) used bidirectional GRU for processing the input sequence. After that, concatenated hidden states of bidirectional GRU is fed into another GRU that incorporates attention mechanism. Attention mechanism allows us to attend into different parts of the input sequence when generating an output for a specific time step t. It generates a context vector as a weighted average of hidden states of the input sequence. Attention mechanism is mostly used in sequence-to-sequence tasks like machine translation, summarization, etc. and there are different variants like Bahdanau Attention [1] and Luong Attention [8]. Followed by the GRU with attention mechanism, a fused state is generated and fed into a feed-forward layer followed by a softmax activation function to output probabilities for each punctuation.

Gale and Parthasarathy (2017) developed several models for punctuation restoration [6]. All models they have propose use LSTM that operates on character features. Character features are obtained through different methods that include static n-gram character embeddings, convolutional layers applied on character windows and so on. One of the architectures implemented in this study also uses static character embeddings which will be explained in later sections with detail.

Alam et al. (2020) used a hybrid approach to solve the punctuation restoration problem. First, they have used a pretrained transformer encoder to obtain contextual embeddings of every token. Then, these token embeddings are given as input to a bidirectional LSTM. Finally, output of each token step is given to a softmax layer to create probabilities for each punctuation. Transformer architecture allows us to fully utilize attention mechanism to generate contextual representations for each time step. In addition, this is done simultaneously for each time step as opposed to sequential processing like RNNs [14].

MacLaughlin et al. tackle quotation recommendation problem [9]. They have used an approach similar to questing answering. Given a document, find paragraphs and spans as candidates for quoting, then rank them. They also used BERT model for this task. This work is mostly irrelevant to the task at hand but it is the only one I have found for quote prediction on literature.

Guhr et al. also worked on punctuation problem. Unlike other works discussed in this section, they have used a broader set of punctuations, which includes apostrophe/single quote character. In addition, they have experimented with pretrained transformer encoder models like BERT, RoBERTa, DistilBERT [3] [7] [11]. It is the most similar approach to this study among the ones we discussed so far.

3 Method

In this section, I provide all steps of this study. After giving details of data and preprocessing, different model architectures implemented during the study are discussed.

3.1 Data

Since apostrophe and quote can be found in both formal and informal texts, we need to create a dataset that a model can generalize from. To achieve that, two different datasets are combined.

SILICONE benchmark [2] consist of several datasets for daily spoken language. These datasets include:

- DailyDialog Act Corpus (Dialogue Act)
- DailyDialog Act Corpus (Emotion)
- Interactive Emotional Dyadic Motion Capture (IEMOCAP) database
- HCRC MapTask Corpus
- Multimodal EmotionLines Dataset (Emotion)
- Multimodal EmotionLines Dataset (Sentiment)
- ICSI MRDA Corpus
- BT OASIS Corpus
- SEMAINE database
- Switchboard Dialog Act (SwDA) Corpus

Actually, it has been built for evaluation of sequence labeling tasks but the sentences in these datasets can be used for our apostrophe prediction task.

Example sentences from the dataset:

- "but i'd probably say that's roughly right ."
- "You're quite insufferable. I expect it's because you're drunk."

Simple Wikipedia [5] is another data source of this study. It is a simplified version of English Wikipedia. Documents are shorter than the original articles and simpler in terms of vocabulary and grammar. This data source is chosen to have examples from formal texts. Reasons for Simple Wikipedia rather than English Wikipedia are time and memory constraints because English Wikipedia dumps are a lot larger than Simple Wikipedia.

Example sentences from Simple Wikipedia:

- "This started the United States' involvement in World War II."
- "The Louisiana Purchase of French-claimed land under President Thomas Jefferson in 1803 almost doubled the nation's size."
- "In this example, the word "every" is a quantifier".

Both datasets are obtained using Hugging Face's Datasets library [4]. Sentences that do not have apostrophes/quotes in it have been filtered out. Finally, to keep the sentence lengths in a reasonable range, sentences that are shorter than 50 characters or longer than 500 characters are also filtered out. In the end, I had 328,034 sentences. Detailed statistics of the datasets are shown in Table 1.

Dataset	# Examples	Min. Length	Max. Length	Avg. Length
Simple Wikipedia	87,823	50	499	94.88
SILICONE	240,211	50	500	134.19

Table 1: Summary statistics of preprocessed datasets

3.2 Preprocessing

Basic preprocessing steps are applied on every sentence:

- Kept only numbers, letters, punctuation marks and spaces.
- There were spaces before some punctuation marks like "he 's" or "but , ". Removed those spaces.
- Found the indices of apostrophes in every sentence to later create the labels according to the model.
- Removed the apostrophes afterwards.

3.3 Char LSTM

I have developed a character-level LSTM-based model. Input characters are encoded into unique integers. Output representation is designed to indicate whether a character is followed by an apostrophe/single or double quote or not. Input and output sequence for a sentence has the same length. In the output sequence "0" indicates that no character should be put after that character in that position. "1" indicates apostrophe/single quote. "2" indicates double quote. Overall, the problem is represented as a multiclass sequence labeling problem.

Model architecture has 3 different type of layers. First, every character is mapped into an embedding vector by the use of an learnable embedding layer. After that two bidirectional LSTM layer is followed. Outputs of forward and backward LSTM are concatenated and fed into a feed-forward layer with softmax activation. This is applied at every time step to create probabilities for 3 categories. Model architecture can be seen in Figure 1.

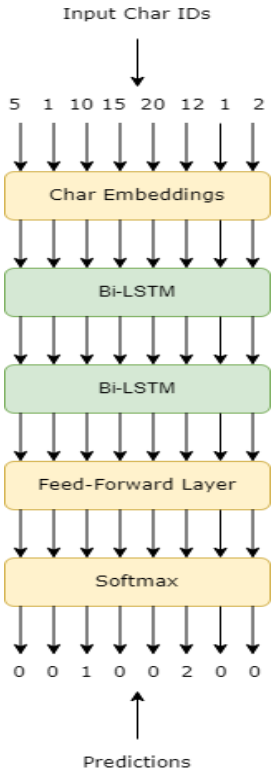

Figure 1: Char LSTM Model

3.4 Fine-tuning on BERT

Another model that I have implemented is fine-tuned BERT. BERT is an encoder-only transformer model [3]. BERT makes use of novel unsupervised pre-training strategies to create a powerful language model. It can be further used on downstream tasks like question answering, sequence labeling etc. BERT has two main novel pre-training ideas: Masked Language Modeling and Next Sentence Prediction. Masked Language Modeling, as opposed to previous left-to-right, or right-to-left approaches, works by predicting masked words in a text by using unmasked words. This allows the model to gain a better understanding of language. However, masking has one disadvantage. It is used during pre-training but not during fine-tuning. This creates a mismatch between pre-training and fine-tuning phases. To overcome this, during pre-training, authors select random 15% of tokens and for every selected token, 80% of the time, they replace it with [MASK] token, 10% of the time with a random token, 10% of the time they keep it unreplaced. Next Sentence Prediction helps the model to learn relationship between sentences in a text. While choosing consecutive sentences for pre-training, A and B, 50% of the time B is a sentence from an unrelated document. Then, model tries to predict if B is the next sentence or not.

In this modeling approach, I have used a pre-trained BERT model. I have added a feed-forward layer at the end for fine-tuning. That layer produces probabilities for each token.

Labeling scheme is different than previously discussed Char LSTM model. BERT operates on token sequences, not characters. As you know, apostrophes can be in different positions in a word. To employ that information into our BERT model, we should use a different labeling technique. I have implemented different labeling methods for apostrophes/single quotes and double quotes. After analyzing the texts in the dataset, most apostrophes are between first and sixth positions from the right of words. So, I have labeled the tokens with the index of the positions of apostrophes in them and I limit the labels up to 6. If there was not an apostrophe in a word, I have labeled it with 0 as before. For double quotes, I have specified 3 different label types. A word can have one double quote at the beginning, one double quote at the end or two double quotes around it. I have labeled these

with 7,8 and 9, respectively. BERT uses a subword tokenization algorithm to create its vocabulary. It works by, building up from characters, merging most frequent character n-grams into subwords. Because of subword tokenization, a word can be represented by a span of tokens. We have to align the labels with tokens. For this, I have used IOB tagging approach which is widely for sequence labeling tasks [15]. I, O and B stands for inside, outside and beginning respectively. You can view the labels in Table 2

ID	Label
0	O
1	B-AP-1
2	I-AP-1
3	B-AP-2
4	I-AP-2
5	B-AP-3
6	I-AP-3
7	B-AP-4
8	I-AP-4
9	B-AP-5
10	I-AP-5
11	B-AP-6
12	I-AP-6
13	B-QU-7
14	I-QU-7
15	B-QU-8
16	I-QU-8
17	B-QU-9
18	I-QU-9

Table 2: Labeling scheme used for transformer models

3.5 Fine-tuning on T5

T5: Text-to-Text Transfer Transformer is a text-to-text framework using transformers [10]. It is capable doing many tasks. Authors proposed an architecture that is mostly similar to original transformer [14]. They have made small changes in layer normalization and used relative learned positional embeddings instead of fixed, sinusoidal positional encodings. They have trained the model on mixture of unsupervised and supervised tasks. I have used a pre-trained T5 model and fine-tuned on the dataset.

Since T5 is a text-to-text transformer, I have designed the output as a text that contains the IOB tags.

4 Experimentation

In this, I give the details of experiments for each model. Note that for all models, I have used 90%-10% train-test split. Reported results are on the test set.

4.1 Char LSTM

For this model, I have used hidden dimension of 256 across all layers, both char embeddings and LSTM states. I have used the Adam optimizer with initial learning rate of 10^{-3}. Also, to overcome class imbalance problem, I have used weighted cross-entropy loss. I have weighted 0 labels with 1 and other labels with 100.

I have trained the model with batch size 128. Used early-stopping with patience of 5 epochs.

Class	Precision	Recall	F-1 Score
No Apostrophe/Quote	1.00	0.99	0.99
Apostrophe/Single Quote (')	0.53	0.97	0.68
Double Quote (")	0.12	0.94	0.21
Overall	0.55	0.97	0.63

4.2 BERT

I have used $\text{BERT}_{\text{BASE}}$ developed by [3]. It has 12 transformer layers with each attention has 12 heads. Hidden dimension size of token embeddings is 768. Overall model has around 110M parameters.

I have fine-tuned $\text{BERT}_{\text{BASE}}$ on the dataset. Trained it for 10 epochs with batch size of 128, learning rate of 3×10^{-5}, weight decay of 10^{-2}. Also, I have used 16-bit mixed float precision training to speed up the training process.

Class	F-1 Score
AP-1	0.72
AP-2	0.98
AP-3	0.98
AP-4	0.72
AP-5	0.83
AP-6	0.74
QU-7	0.72
QU-8	0.70
QU-9	0.78
Overall	0.86

Table 3: Results of Fine-Tuning on $\text{BERT}_{\text{BASE}}$

4.3 T5

I have used T5_{SMALL} model because time and memory constraints. It has an encoder and decoder with 6 attention layers each. Embedding size is 512 across all layers. It contains around 60M parameters. I had to use smaller data with smaller batches because of time and hardware limitations. I have used a random sample of 100,000 sentences. Batch size was 12, which was the maximum that could fit into GPU. Used a smaller learning rate 10^{-5} because of small batch size. Training for only 2 epochs was enough to achieve high performance.

Class	F-1 Score
AP-1	0.997
AP-2	1.000
AP-3	0.991
AP-4	0.981
AP-5	0.969
AP-6	0.978
QU-7	0.998
QU-8	0.996
QU-9	0.998
Overall	0.998

Table 4: Results of Fine-Tuning on T5_{SMALL}

5 Code

Colab is used as the development environment to access GPUs. All code for data preparation, preprocessing and model training is available in this repo: `https://github.com/tugrulhkarabulut/apostrophe-quote-prediction`. Experiments can be reproduced by executing the notebooks. Structuring the code into python modules and pushing models to HuggingFace hub are future ideas.

6 Conclusion

In this study, I have explored ways to predict, or correct, apostrophe and quote usages in sentences. I have mainly focused on models and different representations of output. We have seen that Fine-Tuned T5 model is capable of achieving high performance on this task. This might be due to T5 being trained lots of similar supervised tasks.

Further study could be using an independent corpus to evaluate how well models that I have trained perform. For example, detailed analysis how the model performs quotation might help validating the results. Also, most punctuation restoration models do not have support for apostrophes and quote. A research, or application idea could be building a more powerful punctuation restoration model that is capable of restoring more punctuation marks than current models in literature.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *ArXiv* 1409 (Sept. 2014).

[2] Emile Chapuis et al. "Hierarchical Pre-training for Sequence Labelling in Spoken Dialog". In: *CoRR* abs/2009.11152 (2020). arXiv: 2009.11152. URL: https://arxiv.org/abs/2009.11152.

[3] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[4] Hugging Face. *Datasets*. URL: https://huggingface.co/docs/datasets/index.

[5] Wikimedia Foundation. *Simple English Wikipedia*. URL: https://simple.wikipedia.org/wiki/Simple_English_Wikipedia.

[6] William Gale and Sarangarajan Parthasarathy. "Experiments in Character-Level Neural Network Models for Punctuation". In: *Proc. Interspeech 2017*. 2017, pp. 2794–2798. DOI: 10.21437/Interspeech.2017-1710.

[7] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[8] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *CoRR* abs/1508.04025 (2015). arXiv: 1508.04025. URL: http://arxiv.org/abs/1508.04025.

[9] Ansel MacLaughlin et al. "Context-Based Quotation Recommendation". In: *CoRR* abs/2005.08319 (2020). arXiv: 2005.08319. URL: https://arxiv.org/abs/2005.08319.

[10] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *CoRR* abs/1910.10683 (2019). arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

[11] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *CoRR* abs/1910.01108 (2019). arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

[12] Ottokar Tilk and Tanel Alumäe. "Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration". In: Sept. 2016, pp. 3047–3051. DOI: 10.21437/Interspeech.2016-1517.

[13] Reno University of Nevada. *British vs. American English*. URL: https://www.unr.edu/writing-speaking-center/student-resources/writing-speaking-resources/british-american-english.

[14] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[15] Wikipedia. *Inside–outside–beginning (tagging)*. URL: https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_(tagging).