CV Scientist Exercise

Multi-modal Machine Learning for Multiclass Classification of Receipt Info

Christian Kavouras

The Task

We are given an OCR engine that scans receipts, and outputs text data instances alongside the coordinates of their bounding polygons.

Task:

First, to create a model that can accurately classify these data instances across distinct classes common to receipts.

Next, to group product values for each identified purchased item.

*Assumption:

There is a usable amount of *labeled* training data available (otherwise, there would be limited ability to use ML for modeling)

Sample Training Data (JSON)

Sample Receipt Dimensions:
Height: 200
Width: 100

Receipt Center: (0,0)
Bottom Left: (-50,-100)
Bottom Right: (50, 100)

Top Left: (-50,100)
Top Right: (50, 100)

```
{
  "text": "Murphy",
  "polygon": [-49, 90, -20, 90, -49, 78, -20, 78],
  "label": "store_name"
},
{
  "text": "305s KING FF BPACK",
  "polygon": [-1, 90, 20, 90, -1, 80, 20, 80],
  "label": "product_description"
},
```

Modeling Considerations

Input data is **multimodal**, both **text** and **numerical** data for each input instance.

- How to leverage both of these in training a model?

There are dependencies between text instances on a given receipt

- i.e., store location data more likely to occur near to each other, same for product data, etc

Modeling Considerations

Various types of modeling approaches to try:

- Rule-based, Statistical, Feature-engineered ML techniques, Deep Learning techniques
- In practice, can try to leverage various aspects of all the above (ensemble methods) to continue to improve model performance + build best model

Effectiveness can depend on the amount of available training data

- Want to build a model that can/will ultimately scale out

Model

My Solution:

Treat all text data on receipt in sequence (rows of information)

Process receipt as a sequence of text as model input

Build a deep learning model that can learn to tag text tokens in a sequence

(similar to Part-Of-Speech tagging of a sentence)

Model

My Solution - Deep Learning:

Possible DL Methods for learning sequence data:

Recurrent Neural Network

Transformer based network:

- Known for state of the art performance on modern NLP tasks, leveraging the "attention" mechanism to identify long-distance dependencies in sequence data

* Possible good alternative starting point / baseline using a feature engineering approach could be a **conditional random field**

Building the Deep Learning Network

1. Need an encoder that can learn vectorized representations of text

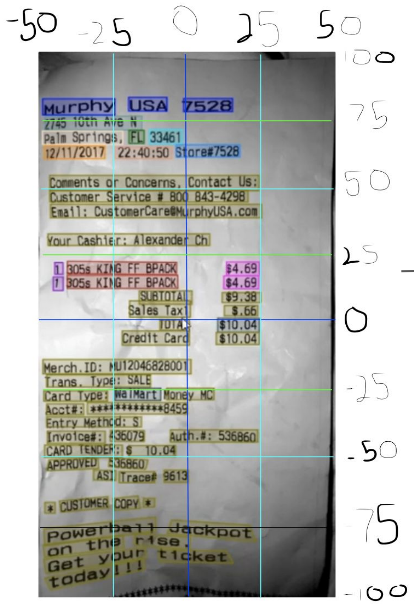
2. Want to leverage "attention" mechanism to learn dependencies between sequences of text tokens

3. Model will have a configurable number of hidden layers (transformation of input embeddings with learned weights of various sizes)

4. Model output will decode a probability distribution over the output classes

    - Most likely class will be predicted, per token

High Level Modeling Schematic



-50
-25
0
25
50
100
75
50
25
0
-25
-50
-75
-100

Murphy USA 7528
2745 10th Ave N
Palm Springs, FL 33461
12/11/2017 22:40:50 Store#7528

Comments or Concerns, Contact Us:
Customer Service # 800 843-4298
Email: CustomerCare@MurphyUSA.com

Your Cashier: Alexander Ch

1 305s KING FF BPACK $4.69
1 305s KING FF BPACK $4.69
SUBTOTAL $9.38
Sales Tax $.66
TOTAL $10.04
Credit Card $10.04

Merch.ID: MU12046828001
Trans. Type: SALE
Card Type: WalMart Money MC
Acct#: *************8459
Entry Method: S
Invoice#: 436079 Auth.#: 536860
CARD TENDER: $ 10.04
APPROVED 536860
ASI Trace# 9613

* CUSTOMER COPY *

Powerball Jackpot
on the rise.
Get your ticket
today!!!

→

[
Murphy,
USA,
7528,
2745 10th Ave N,
...,
Walmart
]

→

hidden rep of "Murphy"
hidden rep of "USA"
hidden rep of "7528"
hidden rep of "2745 10th Ave N"
...
hidden rep of "Walmart"
x N

→

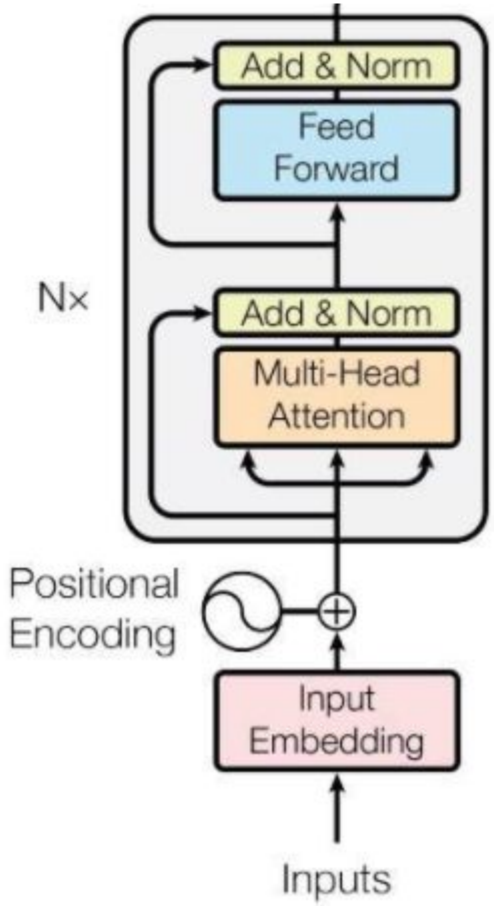
probability vector	store_name
...	...
probability vector	store_address
...	...

How to learn these text representations?

Transformer Block

Hidden layer represented by linear output of transformer block (right)

Multiple blocks can be stacked, creating various levels of encodings



Transformer Block (in detail)

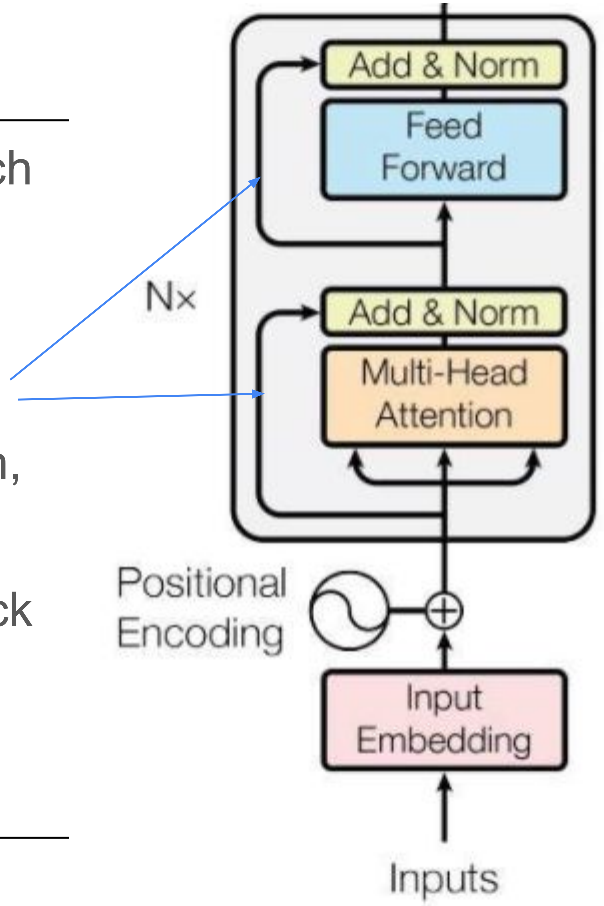
Attention Layer: Vector operation to gather how much each token pays attention to each surrounding token

Layer Normalization: Normalizes outputs

Residual Connections: Adding input back into normalized output layer (known to smooth surface of the loss function, for training)

Feed Forward Layer: Condenses outputs for the next block

Positional Encoding: Added to maintain concept of which values are where in sequence

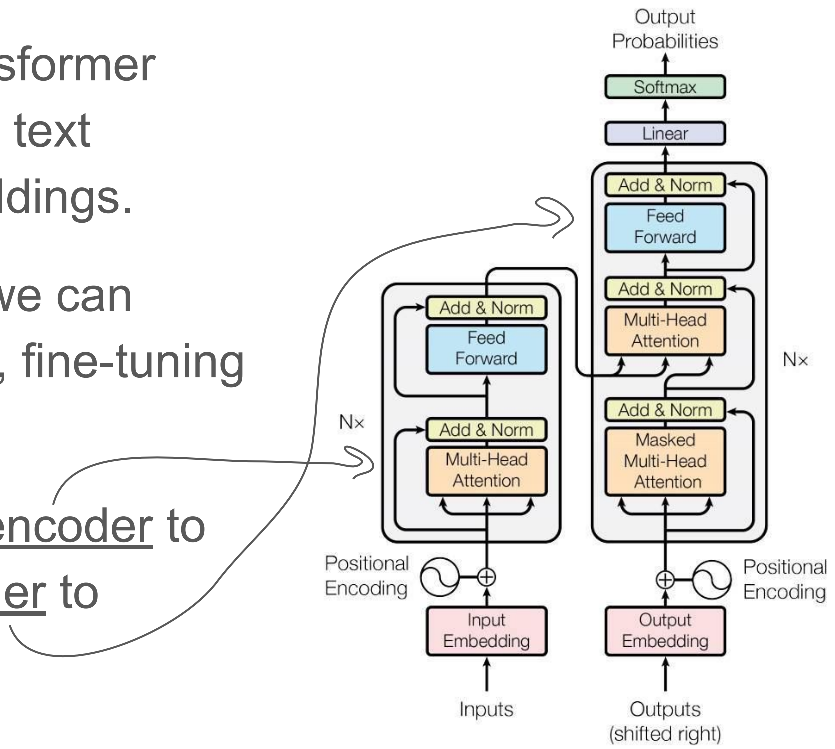


BERT - Full Encoder / Decoder Model

BERT is large scale, publicly available transformer based model. It was pretrained on massive text corpuses to learn generalizable text embeddings.

Instead of writing our model from scratch, we can leverage BERT's pretrained model weights, fine-tuning it on our data set.

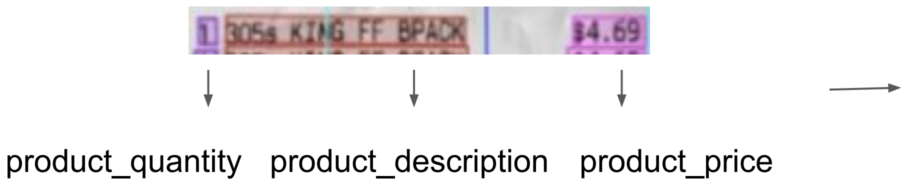
BERT's architecture uses the transformer encoder to encode text input, and a transformer decoder to decode its output



Final Step: Post Processing for Semantic Grouping

Once the model predictions are made on test data, need to group product level data in json format

Can manually scan for a sequence of product labels (quantity, description, price) found in contiguous order, and parse them out as a single entity



```
"Items": [
      {
      "product_quantity": 1
      "product_description": "305s KING FF BPACK"
      "product_price": 4.69
      }
...
]
```

Further Work & Considerations

Full Fine Tuning of the model

- Tuning of model hyper-parameters (epochs, batchsize, learning rate, etc)
- K-fold cross validation across true dataset

Dissecting error & understanding the data

- Possible that dataset is imbalanced
    - -Use stratification of training data into balanced number of classes
    - -Want to observe which labels are hard to predict - can supplement model through, for example, additional rule-based or feature engineered mechanisms to account for it

Continually training / tuning on new data as the model grows to scale