Product Price Suggestions for Online Marketplaces

Emmie Kehoe (emkehoe), Ian Naccarella (inaccare), Javier Raygada (jraygada) *CS 230: Deep Learning*

Problem Definition

- Our aim was to build an algorithm that **suggests prices to online sellers** based on various attributes of these items, thus allowing these sellers to **save time and resources** when determining the value of their products

- Using **~1.4M item descriptions** on the **Mercari shopping** website, we trained an algorithm which predicts the prices of items based on **description, condition, brand, and category**

Inputs

- **Item description**:
  - **Bag of Words (boW)**: binarized vector of length |vocab| indicating presence or absence of each word in vocab at its index (Multi-Hot vector)
  - **Word2Vec (w2v)**: 100-dimensional GloVe vector or vector trained on our own corpus to capture meaning of word
- **Item condition**: (scale of 1-5) One-Hot vector
- **Brand**: (~5K different brands) One-Hot vector
- **Category**: Multi-Hot vector

Outputs

- **Item description**:
  - **SoftMax**: sorted the prices into buckets (12 and 20 buckets) using *cross entropy* loss function
  - **Linear**: output exact price using *Root Mean Squared Logarithmic Error* cost function

Models

Neural Network (2 hidden layers)

- **SoftMax Output (12 buckets)**
- **Only item description Input**
  - **Bag of Words** 1
  - **Word2Vec (using pre-trained GloVe vector &** + **averaging vectors of words in description)** 2

Recurrent Neural Network with LSTM

- **SoftMax Output (12 buckets)**
- **Only item description Input**
  - **Word2Vec (using pre-trained GloVe vector)** 3
  - **Word2Vec (trained on our corpus)** 4
- **Word2Vec (trained on our corpus)** +

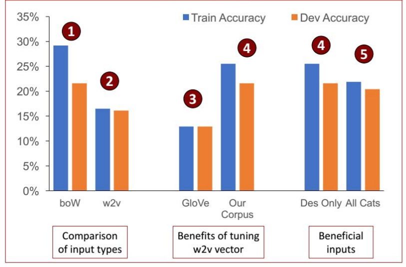
- **SoftMax Output (12 buckets)**
- **Word2Vec (trained on our corpus)**
  - **Item descriptions only as input** 4
  - **All categories as input** 5
    - **+ multiple FC layers** 10

- **All categories used as Input**
- **Word2Vec (trained on our corpus)**
  - **SoftMax Output (20 buckets)** 6
    - **+ multiple FC layers** 8
  - **Linear Output** 7
    - **+ multiple FC layers** 9

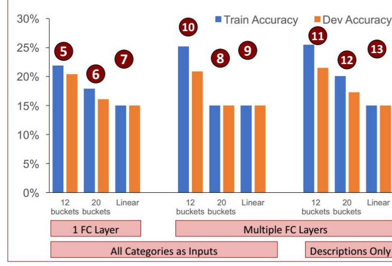
- **Only item description Input**
- **Multiple FC layers**
- **Word2Vec (trained on our corpus)**
  - **SoftMax Output (12 buckets)** 11
  - **SoftMax Output (20 buckets)** 12
  - **Linear Output** 13

Results

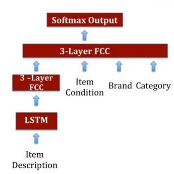
Input Tuning



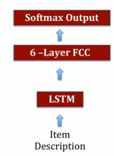
Output/Architecture Tuning



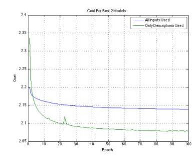
Best Models



LSTM Network with multiple FC Layers, 12 Buckets Output



LSTM Network with multiple FC Layers, Descriptions Only, 12 Buckets Output



Other Inputs seemed to add noise in predicting price

	All Inputs Used	Only Descriptions Used
Train Accuracy	23.4%	21.6%
Dev Accuracy	25.0%	23.4%

Challenges

- We had to retrain the word2vec vectors for our application
- Bag of Words took too long to run so we couldn't do very many epochs
- Bi-Directional LSTM took too long to train so we just did normal LSTM
- Had to pivot from storing all of our sentences encodings in CSVs to doing the encoding step at each mini-batch due to memory complexity

Acknowledgements

We want to thank all of the wonderful CS 230 TAs and specifically our mentor, Soroosh Hemmati! We'd also like to thank AWS for the free credits!

Discussion

Bag of Words vs. w2v

- Bag of words at first seemed to outperform Word2Vec
  - Main con of BoW was that it was very slow
- Standard NN that used BoW achieved same accuracy on dev set

NN vs. RNN

- RNN (LSTM) improved accuracy on train set by 10% on training set and 5% on dev set
- Final architecture used an LSTM architecture followed by a Fully Connected Multi-Layer Perceptron network
- Though accuracy seems low, we believe that this is because this is an inherently hard problem and that the Bayes error is not much lower than our model's

Future Work

- Deeper architecture
- Bi-directional LSTM instead of normal LSTM
- Attention Model instead of LSTM
- More hyperparameter tuning
- Characterize which combination of inputs add the most value in predicting prices
- Find optimal bucketing strategy such that the buckets are not only representative of the market but also cover a more similar price range
- Do more error analysis and penalize the most common errors more to speed up learning
- Filter data used to train to account for outliers