Amazon Price Predictor

Stephanos Kantzidis
Jagadish Rangrej
Haoran Zhang

September 10, 2017

Overview

Objective

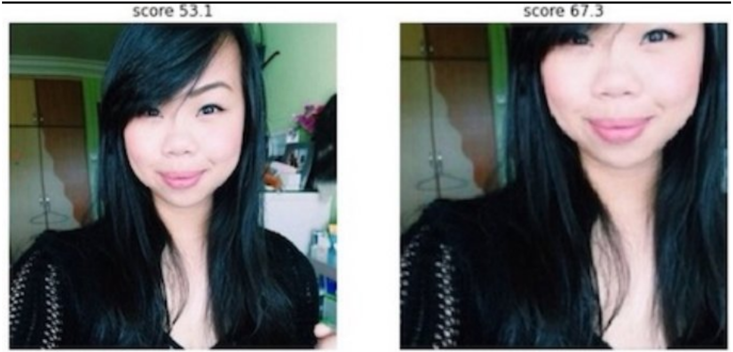
To create an application that returns the predicted price of an item, given an image of the item and its title.

Purpose

The application might be useful in the following ways:

- Allows a merchant to determine what price they should sell their products for, without having to do any prior research.
- Allows a consumer to determine the appropriate price for an item, without having to do any prior research.
- Allows a merchant to choose which image, in the consumer's eyes, is of greatest value.

An Analogy



An Analogy

Demo - Stroller Prices

Objective
Purpose
Hypothesis and Assumptions
First Steps
The Obstacle
The Final Model
Productionizing


```
Brown stroller: $142.85
```

Demo - Stroller Prices


```
Black stroller: $48.85
```

Demo - Stroller Prices

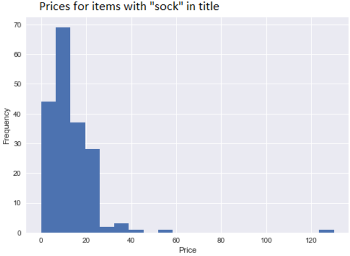

```
Stokke stroller with car seat: $274.55
```

Assumptions

- R-CNN image features represent intrinsic aesthetic features of the product, which we assume has a relationship with price.
- Meaning of title words are representable by bag-of-words model (ex: no negation)
- Regression model is capable of selecting most relevant features from a large pool of features.
- The prices set by the merchants on Amazon are fair market prices of the items.

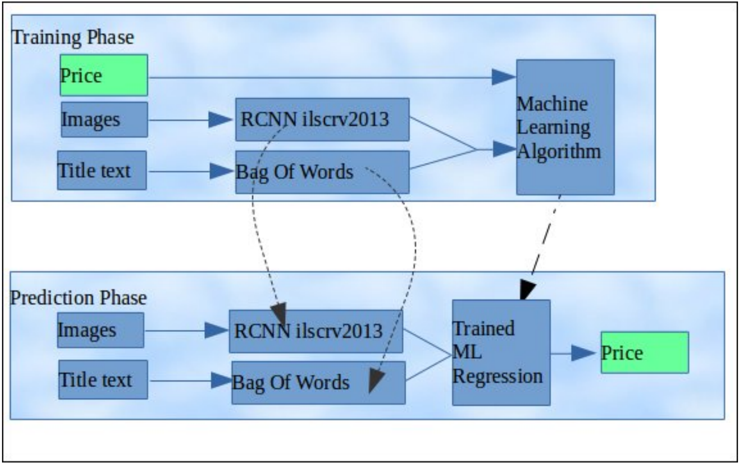
Hypothesis

- The title of the product is important in capturing the bigger class of the product.



- The image of the product sufficiently captures aesthetics of the product which is directly correlated with price.

Schematic



Training Phase
Price
Images
Title text
RCNN ilscrv2013
Bag Of Words
Machine Learning Algorithm
Prediction Phase
Images
Title text
RCNN ilscrv2013
Bag Of Words
Trained ML Regression
Price

Pre-Processing

1. Read in metadata and reviews datasets

```
reviews = getDF('reviews_Baby.json.gz')
meta = getDF('meta_Baby.json.gz')
```

2. Aggregate reviews by product ID, take the average Unix review time

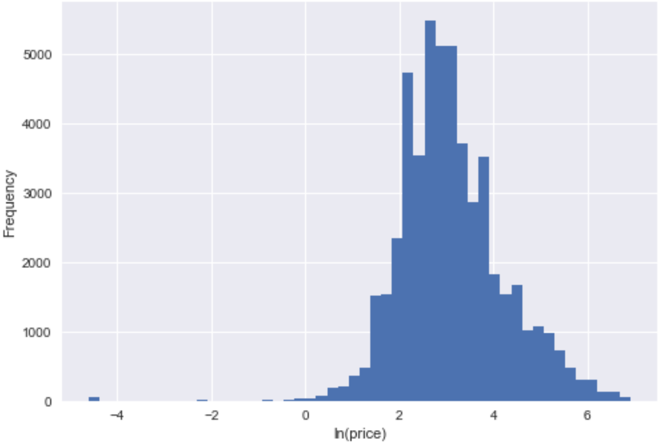
```
reviewsDF = reviews.set_index('asin').groupby(level =
    0)['unixReviewTime','overall']
      .agg(np.average)
```

3. Merge the two dataframes

```
df = meta.merge(reviewsDF, how = 'inner', left_index
    = True, right_index = True).dropna(how = 'any')
```

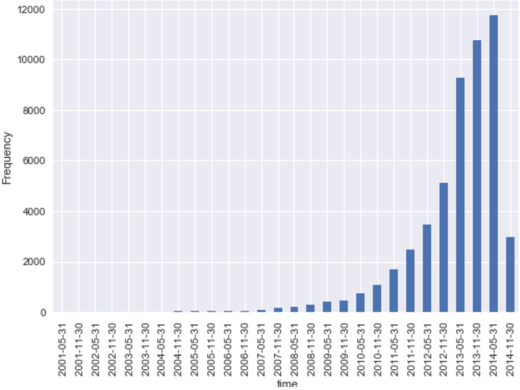
Exploratory Analysis


```
np.log(df['price']).plot.hist(bins = 50)
```



Exploratory Analysis

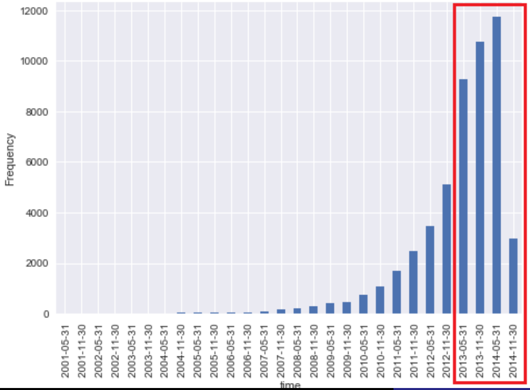
Semi-annual Frequency of unique products with average review time



Exploratory Analysis

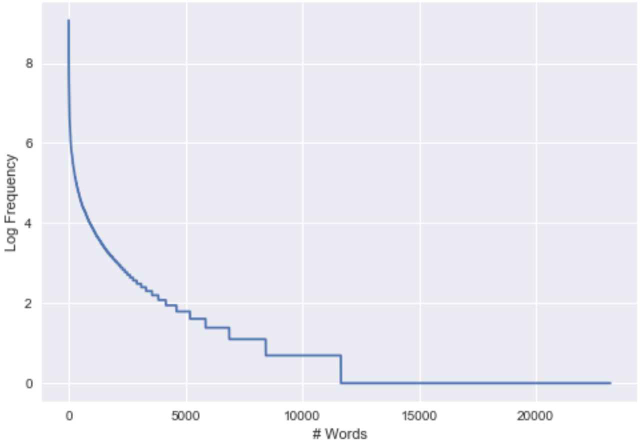
4. Remove all items with average review date before 2013-01-01. 33378 items remain.

```
df = df[df['time'] > '2013-01-01']
```

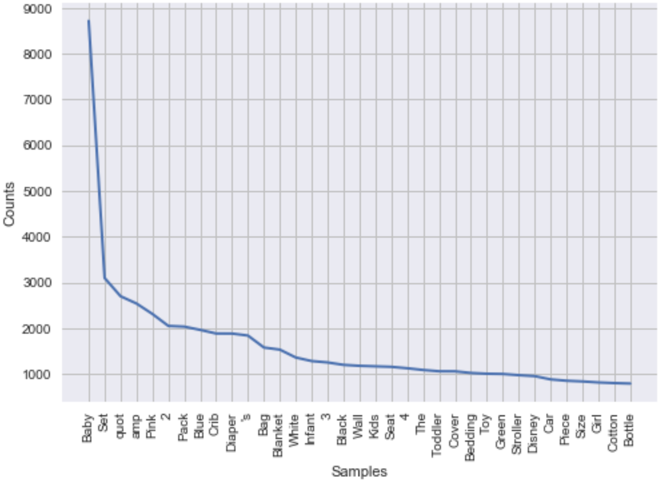


Exploratory Analysis

5. Tokenize titles, remove stopwords



Exploratory Analysis



The Obstacle

4.2 Visual Features

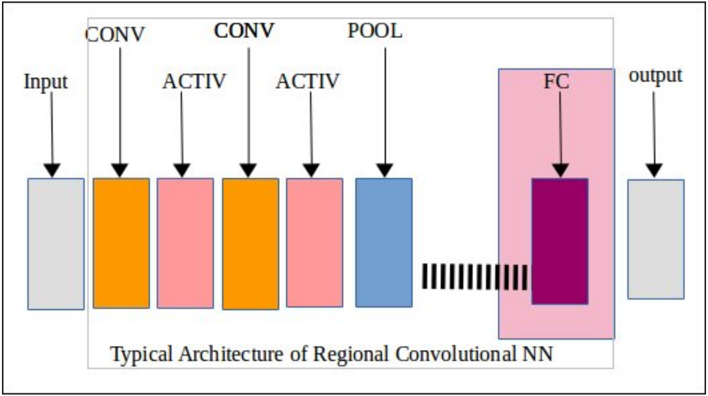
To extract a visual feature vector f_i for each item i in the above datasets, we employ a pre-trained convolutional neural network, namely the Caffe reference model [15], which has previously been demonstrated to be useful at capturing the properties of images of this type [26]. This model implements the architecture proposed by [21] with 5 convolutional layers followed by 3 fully-connected layers and was pre-trained on 1.2 million ImageNet (ILSVRC2010) images. We obtain our $F = 4096$ dimensional visual features by taking the output of the second fully-connected layer (i.e., FC7).

- Unable to find their pre-trained model and replicate their results
- Image features provided can only be used to train the model, but can't be used to evaluate new products

The Solution

- Imagenet: A collection of 14 million+ labeled images.
- Caffe Framework: A collection of tools used to build R-CNN models.
- BVLC Reference ILSVRC2013: A pre-trained R-CNN model. Part of the Caffe Framework. Typically used for classification.
- But... we don't have a classification problem!

The Solution



Each image is downloaded, scaled and sent into this model.
4096 floats are extracted from FC7.

The Model

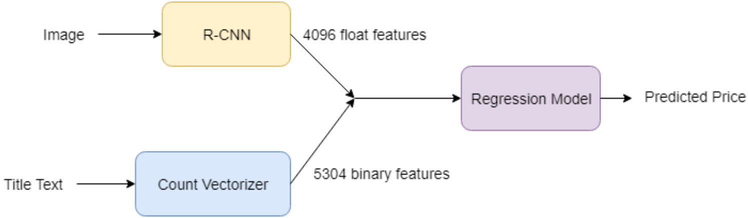


Image
R-CNN
4096 float features
Title Text
Count Vectorizer
5304 binary features
Regression Model
Predicted Price

The Model - Textual Features

Count Vectorizer

- Word tokenization
- Stop word removal
- Minimum frequency = 5
- Binary output

The Model - Regression Model

Tried these regressors:

- Gradient Boosted Decision Tree
- K-Nearest Neighbors
- SVM

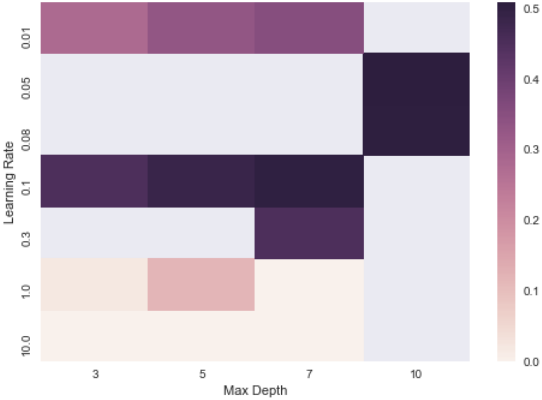
Evaluating using R^2 as metric.

Gradient boosting gives the best results from a preliminary analysis.

The Model - Regression Model

Optimizing gradient boosting using a grid search:

- Using R^2 as metric
- 3-fold cross validation



The Final Model

Final Parameters ($R^2 = 0.509$):

- 500 estimators
- Learning rate of 0.05
- Max depth of 10

Feature	Importance
IF782	0.008250
IF1001	0.004347
IF4095	0.003223
stokke	0.003119
IF1729	0.003085
IF49	0.002951
future	0.002684
monitor	0.002653
IF2707	0.002522
IF946	0.002493
IF2203	0.002378
caden	0.002325
petunia	0.002299
camera	0.002282
storksak	0.002215

Considerations for Productionizing

Scaling up:

- Repeat the above process, with a different regression model for each product category.
- Automatic detection of what product category an image belongs to, then use the appropriate regression model for that category.
- Acquire more memory and computational power, both to train and optimize models, and also to predict on new images.
- Reduce computational complexity by reducing number of features.

Considerations for Productionizing

Model Improvement:

- Use finer grid search parameters
- Explore more regression models
- Include rating, n-grams, number of reviews, time, or other features

The ideal final product:

- A website where the user could upload their images and receive the price estimate directly
  - No need to install any libraries or dependencies client side.
- A phone application that allows the user to take images with their cell phone camera, and receives the price estimate right away.