E-Commerce Recommender System Design

Team 2
Increment 3

Wayne Nguyen

Hongcheng Jiang

Zhaobin Zhang

Fei Wu

ABSTRACT

As we already achieved the coarse results for the whole pipeline in last Increment. This increment we will focus on refine current results and prepare for final integration. Additionally, since the end goal is to give recommendations to the user, the application should not exceed a certain time threshold. If it takes too long, this can make the user not use the application. Further on, we would want to incorporate 'live' images of the fashion item.

1 INTRODUCTION

A broad use case is to have the user import any image and have our application recommend the fashion item of interest. This of course is a huge problem to tackle. A better starting scenario is to test staged or cataloged images. The fashion item image should be centered, in high definition, and formatted to a certain size. There should be no to little background noise in the image. With this, we would run a three-stage classifier module on the item. The first stage is a deep learning algorithm to classify what the item is. Second, a machine learning algorithm (Clarifai) to classify its design elements and features. The third stage would be a logo recognizer algorithm to see if there is any logos attached to the fashion item. Once the module is done, the image is now correctly tagged with what the item is (watch), what details it has (silver with diamonds), and what brand (Rolex). This tag can then be run through a knowledge recommendation algorithm to show the user where to buy that or a similar item.

2 RELATED WORKS

TensorFlow-for-poets: https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/

Docker: https://www.docker.com/

Clarifai: https://www.clarifai.com/

Given shallow learning algorithms (class spreadsheet): https://docs.google.com/spreadsheets/d/1TkZalFcjQ8MCWLdkQ kTdEhUYG5V8hsUF4KzMNmy_Rto/edit?ts=5a5e212f#gid=742 202525

Deep learning [3] has become a promising technique to perform various tasks, especially in image/video-based tasks. For examples, face recognition, classification and retrieval to name a few. In [1] a novel E-Commerce system has been proposed which is based on VGG-16 [2].

3 PROPOSED FRAMEWORK

3.1 Recommender system pipeline

As illustrated in Fig. 1, the fashion dataset will first be fed to initial 3-stage image classifier module. It will be fine tuned and afterwards, test images will be used to validate the accuracy. The test/formatted output from the test pipeline will be leveraged to generate the recommendation decisions.

3.1 System Specification

IDE: IntelliJ, SBT, Pycharm, Android Studio

Services: Image annotation - Clarifai API https://www.clarifai.com/, openimaj, Adobe spark, TensorFlow

https://spark.apache.org/, object detection - CS5542 lab tutorials.

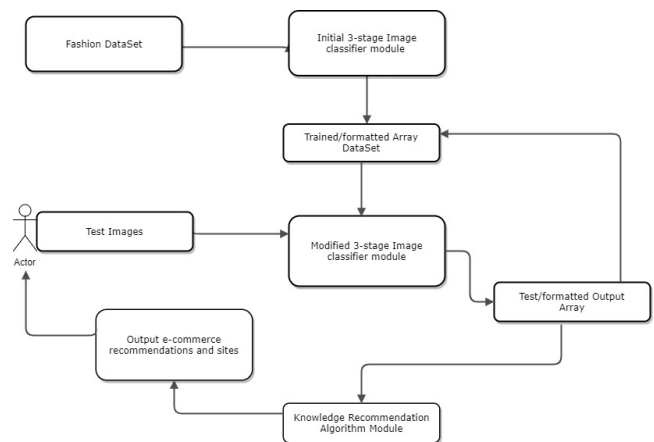Sources @ https://github.com/waynenguyen303/deep-learning-visual-eCommerce/tree/master/algorithm-sources

Figure 1: Framework of the proposed recommender system.

4 DATASETS

4.1 Initial Dataset Categories

When first starting this project, the initial intention was to finely categorize the apparel that we would search for online as shown in Fig. 2. This could be shoes, coats, T-shirt, pants, dresses, polos, etc. With this notion, we made 9 generic categories for online apparel: dresses, watches, shoes, pants, coats, shirts, suits, Adidas, and hats. The images (.jpg or .png format) were mostly found with Google searches and filed according to their categories. The images were intended to have a lot of diversity, so some had models in them, some with background (live images). Some images had more than one category, but the main subject was of that category. With this, each category had around 100 images each. Of course, we recommend having thousands of images per category for better training but without a proper company/website to facilitate these images, our data set tends to be rather low for each category. The data set was then broken down into training and testing sets. The training set with about 90% of the images per category and the testing set with 10%. Validation images consisted of about 1% of the total images.
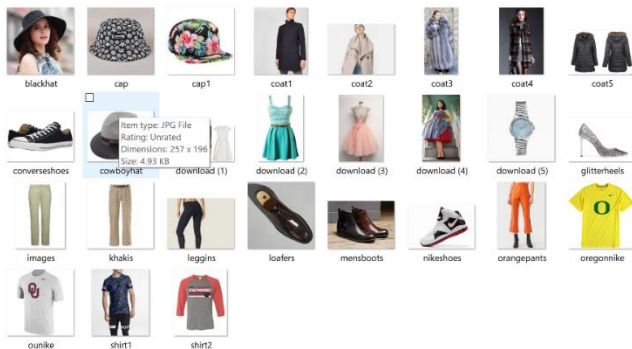

Figure 2: The initial dataset which consists of 9 categories. The dataset convers most of clothes and hats people usually wear.

As we will see further on, this type of data set was not ideal for image classification or object detection. With how some of the algorithms work, some of the categories were too similar like T-shirts, coats, and polos. Also, background and multi-category images tend to make the data set very noisy.

4.2 Improved Dataset Categories

With this new knowledge of how the classification and object detection algorithm works. Our data set was further refined to get better accuracies. First, we limited the images to catalog style photos. This is where there is very little to no background. No models are included and only the category was in that image, no mixed categories. Also, categories were refined to be only sports related and their logos. So, our new category are: sport tops, sport bottoms, sport shoes, sport hats, and watches. Logo categories are: Nike, Adidas, Puma, Under Armor, and New Balance. These categories were chosen to maximize their differences in shape and to hopefully improve accuracies of the implemented classification and object detection algorithms. Again, each category has around

100 images each. Around 90% of the images in each category were for training and 10% for testing.
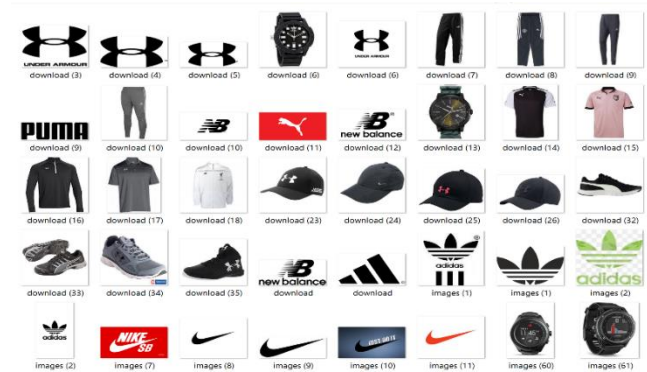

Figure 3: The improved dataset which only consists of only 5 categories. They are more distinguishable compared with the initial dataset categories.

5 EXPERIMENTAL RESULTS

In this section, we will show the experimental results. The first section is conventional algorithms and the second section is deep learning-based algorithm.

5.1 Shallow Learning Classification

To do our shallow learning image classification, we will be using the Adobe Spark library using the IntelliJ IDE. We will compare the Naïve Bayes, decision tree, and random forest algorithms on our initial data set and see their accuracies. Given in class, the image classification application from tutorial 4 was modified to run our data sets and the above algorithms. The application can be downloaded here at https://umkc.app.box.com/s/xdf9p87cuuwxxy3ajt3gzog2mcbe3l50. To run this application, the test images needs to be resized and modification to the code needs to be done for each algorithm.

```
|=================== Confusion matrix =========================
0.0  0.0  4.0  1.0  4.0  0.0  0.0  0.0  1.0
0.0  1.0  4.0  1.0  1.0  0.0  0.0  0.0  2.0
1.0  1.0  3.0  1.0  0.0  2.0  0.0  0.0  2.0
0.0  0.0  6.0  0.0  0.0  3.0  0.0  1.0  0.0
2.0  0.0  5.0  0.0  3.0  0.0  0.0  0.0  0.0
1.0  0.0  3.0  0.0  1.0  0.0  0.0  0.0  4.0
0.0  0.0  2.0  2.0  1.0  4.0  0.0  0.0  1.0
0.0  1.0  6.0  1.0  1.0  0.0  0.0  0.0  1.0
0.0  1.0  5.0  1.0  0.0  3.0  0.0  0.0  0.0
0.08888888888888889
```

Figure 4: Random forest tree accuracy for the initial dataset.

Fig. 4-Fig. 6 are the accuracy of shallow learning algorithms for the initial dataset categories. Each column in the confusion matrix represents a category in the "IMAGE_CATEGORY" list. Accuracies for the 3 shallow learning algorithms on this data set was very low. The random forest algorithm was at 8.88%, decision tree algorithm at 11.49%, and Naïve Bayes algorithm at

8.04%. This is mostly due to the low number of training data images and also the diversity of the images. With this though, the best shallow learning algorithm for this data set is the decision tree. Still these low accuracy scores means there needs to be refinement in the code or data set. We are leaning towards the later.

```
|================== Confusion matrix ========================
1.0  2.0  2.0  4.0  1.0  0.0  0.0  0.0  0.0
1.0  0.0  1.0  4.0  3.0  0.0  0.0  1.0  0.0
1.0  0.0  0.0  2.0  3.0  0.0  1.0  1.0  0.0
0.0  0.0  1.0  7.0  2.0  0.0  0.0  0.0  0.0
0.0  0.0  2.0  6.0  2.0  0.0  0.0  0.0  0.0
0.0  1.0  1.0  4.0  1.0  0.0  0.0  0.0  0.0
1.0  0.0  1.0  3.0  3.0  0.0  0.0  1.0  1.0
1.0  0.0  2.0  2.0  4.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  6.0  3.0  0.0  0.0  1.0  0.0
0.11494252873563218
```

Figure 5: Decision tree accuracy for the initial dataset.

```
|================== Confusion matrix ========================
0.0  3.0  2.0  1.0  2.0  0.0  0.0  0.0  0.0
0.0  0.0  2.0  0.0  0.0  0.0  0.0  1.0  1.0
1.0  0.0  2.0  0.0  0.0  2.0  0.0  0.0  2.0
1.0  1.0  3.0  1.0  0.0  1.0  0.0  0.0  1.0
0.0  1.0  2.0  0.0  1.0  3.0  1.0  0.0  1.0
0.0  1.0  1.0  0.0  0.0  1.0  1.0  0.0  1.0
0.0  1.0  1.0  0.0  3.0  0.0  1.0  0.0  0.0
0.0  0.0  2.0  0.0  3.0  2.0  1.0  1.0  0.0
0.0  1.0  1.0  2.0  2.0  1.0  1.0  0.0  0.0
0.08045977011494253
```

Figure 6: Naïve Bayes accuracy for the initial dataset.

5.2   Shallow Learning for Improved Dataset

Fig. 7-Fig. 9 illustrate the accuracy for improved dataset using shallow learning algorithms. For the new data set, the random forest algorithm its accuracy is 47.05%. The Naïve Bayes accuracy is 45.09% and decision tree at 37.25%. This is much better than the old data set and with more training images, we believe we can get the accuracy to go up even more. Still, we would like to be in the 80% accuracy level before we want to implement this kind of shallow learning algorithm

```
Random Forest Tree
|================== Confusion matrix ========================
6.0 3.0 2.0 0.0 0.0
1.0 3.0 3.0 3.0 0.0
0.0 2.0 8.0 0.0 0.0
3.0 1.0 2.0 4.0 0.0
0.0 0.0 5.0 2.0 3.0
0.47058823529411764
```

Figure 7: Random forest accuracy for the improved dataset.

```
|================== Confusion matrix ========================
7.0 0.0 4.0  0.0 0.0
0.0 1.0 8.0  1.0 0.0
0.0 0.0 10.0 0.0 0.0
1.0 0.0 4.0  5.0 0.0
0.0 0.0 10.0 0.0 0.0
0.45098039215686275
```

Figure 8: Naïve Bayes accuracy for the improved dataset.

```
|================== Confusion matrix ========================
6.0 2.0 0.0 3.0 0.0
2.0 3.0 3.0 2.0 0.0
0.0 7.0 3.0 0.0 0.0
1.0 5.0 0.0 4.0 0.0
0.0 5.0 2.0 0.0 3.0
0.37254901960784315
```

Figure 9: Decision tree accuracy for the improved dataset.

5.3   Clarifai Machine Learning for Object Detection

For object detection, we will use the Clarifai API and its machine learning algorithm. This can be implemented on a web page or on the Android Studio IDE. A Clarifai API key is needed to run its libraries and can be issued at https://www.clarifai.com/. To use Clarifai, an image is sent to its server and a Json response is sent back. The Json has the object detected in the image and also its accuracy.
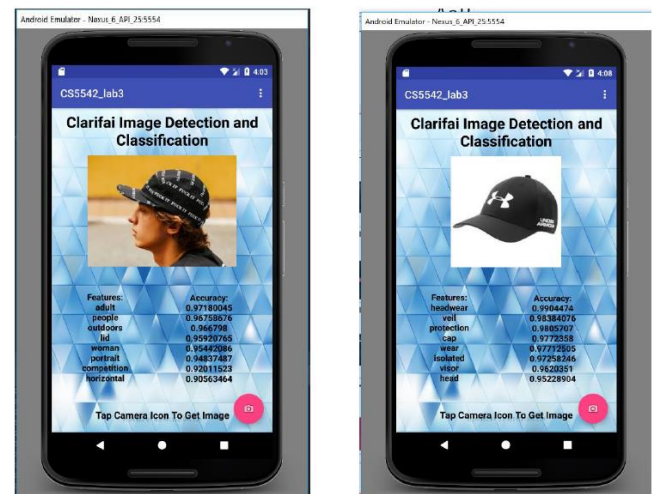

Figure 10**:** Clarifai image detection and classification with Android interface.

From the images above, we can see the Clarifai object detection algorithm in action. Its algorithm is accurate and its response time is fast compared to the shallow learning algorithms. What makes a difference in these two is what type of image is

sent to Clarifai. For the picture on the left, Clarifai detected 'adult', 'people', 'outdoors', and then 'lid. For the picture on the right, 'headwear', 'veil', 'protection', and 'cap' was the top detected words. For our project, we would want the feature names on the right picture. Again this is due to the image selected and sent to Clarifai. This is just another case for the correct data set to be used.

5.4 Tensorflow for Deep Learning

For another take at image classification and get better accuracies, we will use TensorFlow and its deep learning algorithm. We will mostly look at the SoftMax implementation. The CNN implementation will be done at a further date. Here we will be using Docker and tensorflow-for-poets to train and test the data set. A tutorial can be seen here at https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/?utm_campaign=chrome_series_machinelearning_063016& utm_source=gdev&utm_medium=yt-desc#0 to implement TensorFlow.

Figure 11: Tensorflow for deep learning. The accuracy is much better than shallow learning methods.

Here we have trained the categories for: bottoms, hats, shoes, tops, and watches. When testing the images, the accuracy is very good. TensorFlow predicted the image on the left as a shoe at 99.99% and for the image on the right, it predicted it as a shoe with 91.11% accuracy. This is very good compared to the shallow learning algorithms. Also, the training time was considerably less using TensorFlow versus the 3 shallow learning algorithms. We will probably be using this algorithm for object classification in our project.

6 CONCLUSIONS

In summary, we have learned that the data set to train and test plays a very important role in how accurate algorithms can be. The less noise the image has and the about of training data is vital

for accuracies. This means that more attention to getting the correct data set needs to be the first priority. Second then should be what kind of algorithm to use. For image classification, we would use TensorFlow's deep learning algorithm versus the 3 shallow learning ones. Clarifai's machine learning is good for object detection but we still have to be careful of what images to send it. Maybe a mix of the shallow learning algorithms and Clarifai can improve the list of object detected or wanted.

A PROJECT MANAGEMENT
Completed:

Initial data set (hours = 5): Wayne Nguyen
Algorithms research (hours = 6): all members
Algorithm usage/code (hours = 6): Zhaobin Zhang, Wayne Nguyen
IDE application implementation (hours = 12): all members
Paper Incrementation (hours = 2): Zhaobin Zhang
Image/data set categorization (hours = 3): Hongcheng Jiang
Application design/workflow (hours=3): Fei Wu, Zhaobin Zhang
Tensorflow Algorithm: Wayne Nguyen
Clarifai machine learning: Wayne Nguyen, Hongcheng Jiang
Paper Increment 3: Zhaobin Zhang
Participation: all members 25% each

To be Completed:

Integrate into Android Application (hour = 5): Wayne Nguyen
Refine application design/workflow (hours = 3): Zhaobin Zhang
Start knowledge recommendation algorithm (hours = 5)
Paper Final Report (hours =2): Fei Wu, Hongcheng Jiang
UI implementation (hours = 5): Wayne Nguyen

A.3.3   Analysis of the Dynamic Coupling as a Function of the Gap Size

REFERENCES
[1] Devashish Shankar, Sujay Narumanchi, Ananya H A, et al, Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce, https://arxiv.org/pdf/1703.02344.pdf, 2017.
[2] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, CVPR, 2014.
[3] Yann LeCun, Yoshua Bengio and Geoffrey Hinton, Deep Learning, Nature, 2015.