ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 9 N. 3 (2020), 49-68
eISSN: 2255-2863
DOI: https://doi.org/10.14201/ADCAIJ2020934968

Awjedni: A Reverse-Image-Search Application

Hanaa Al-Lohibi, Tahani Alkhamisi, Maha Assagran, Amal Aljohani, and Asia Aljahdali

Collage of computer Science and Engineering, Jeddah University, Jeddah, Saudi Arabia
1675753@uj.edu.sa, 1506942@uj.edu.sa, 1605216@uj.edu.sa, 1605696@uj.edu.sa, aoaljahdali@uj.edu.sa

KEYWORD

Reverse-image search; Deep learning; Localization algorithm.

ABSTRACT

The abundance of photos on the internet, along with smartphones that could implement computer vision technologies allow for a unique way to browse the web. These technologies have potential used in many widely accessible and globally available reverse-image search applications. One of these applications is the use of reverse-image search to help people finding items which they're interested in, but they can't name it. This is where Awjedni was born. Awjedni is a reverse-image search application compatible with iOS and Android smartphones built to provide an efficient way to search millions of products on the internet using images only. Awjedni utilizes a computer vision technology through implementing multiple libraries and frameworks to process images, recognize objects, and crawl the web. Users simply upload/take a photo of a desired item and the application returns visually similar items and a direct link to the websites that sell them.

1. Introduction

With the technological advancement of today's world, there are billions of photos on the internet. Therefore, when people browse through social media, they might find an attractive item such as a bag, shoe or whatever in any post on internet and they may want to buy that item. But they have a problem on searching that item via the normal text search. So, they could spend a lot of time trying to guess the appropriate keyword would lead to that desired item. This is where computer vision come into play.

Over the years, computer vision has been evolved many area of field including extraction of image pattern and information interpretation. Computer vision is a combination of image processing and pattern recognition. The output from computer vision is image comprehension. Therefore, computer vision is the specialty of extracting information from images, and it depends on the computer technology system, whether it is related to image quality, image improvement or image recognition (Wiley, V et al.,2018).

Deep learning is an open, advanced technology that has solved many complex problems in computer vision. Therefore, many new and complex applications have become possible to implement. At present, the latest trend in research and development in machine learning is deep learning, because digital learning methods have made pioneering developments in computer vision and machine learning(Patel, P et al.,2020).Deep learning is to approximate and reduce large and complex data sets into high-resolution predictive and transformative outputs, thus greatly facilitating human-centered intelligent systems. Deep learning architectures can be applied on all types of data including visual, audio, digital, text, or some combination (Hatcher, W et al.,2018).

In this project, we are developing a visual search application that enables people to search for a product either by uploading its image or taking its photo using phone's camera. As a result, the application returns similar products in an efficient amount of time using a Deep Learning technique.

The project intends to develop an application that aims to facilitate searching for finding an item and its similarities considering minimizing the required time to find the desired item. Our focus is on utilizing visual search technology by developing an application that prompts users to either upload or take a photo. The photo is then analyzed with localization algorithms that localize and classify every separate item within the photo, as well as extracting the important feature (i.e. color, shape. . . etc.). To implement the application, we plan on utilizing independent open source tools built by different companies. The main point that we need to take care of is ensuring that these tools work together smoothly. To overcome of this obstacle, it is vital to pay meticulous attention to the data types, protocols, and APIs (Application Programming Interfaces) used. The application itself will be built on a Windows coding platform.

The paper is organized as follows, section 2 provides a background sight about the domain that the proposed application is covered. Section 3 discusses related work to our project and similar applications. Section 4 presents our proposed solution. The system design and implementation are presented in section 5 and 6. Section 7 provides the paper's conclusion and future work.

2. Background

To fully understand this project, it is vital to have some background knowledge regarding the field of study (domain) this project is based on. This project utilizes the use of computer vision in the context of image search, which falls under deep learning. We will provide a description of the most related terminology to that field.

ML (Machine Learning) An Artificial Intelligence is a subcategory where computers (machines) are built to «learn» how to perform a specific task using past experiences (trial and error).

Training Data Set a set of data (text, images... etc.) is used to «train» a computer to master a certain skill, like image recognition for example.

Learning Algorithm: is used to find patterns in training data sets to achieve a learning goal; it is used to make computers imitate the learning process of a human. The algorithm itself learns by finding patterns in the given data.

Training Process: is the process of applying a learning algorithm to a computer Learning Models: are the output of a training process. Learning models are used to make better predictions in the future. They are like generic functions that can be applied to a computer to approximate a real-world scenario.

Neural Networks: are artificial neural networks that mimic biological neural networks found in animal brains. They are used by computers in the field of machine learning to make better performance predictions in the future. Also, they are used to extract information that is later used as inputs for clustering/classification algorithms. Neural networks have a depth of three layers at most.

Deep Learning: a subcategory of machine learning that is based on neural networks. In the field of deep learning, neural networks are given massive amounts of training data to build learning models that are capable of processing data in a new and exciting way that have different applications, e.g. computer vision.

Deep Neural Networks: a complex neural network with many layers (more than three). Computer Vision: is the automation of tasks performed by the human visual system, e.g. detecting certain objects in images (trees, traffic signs) and face recognition.

Query Image: A query image is an input image into a computer vision program by a user of that program. **Reverse Image Search:** is a type of search engine technology that uses an image as an input query (query image) and returns images that are identical to or related to the query image.

Memory Footprint: is the amount of memory (RAM) software uses when running.

Binary Signature: is a binary representation of an object within a picture; it represents the color, shape, size, and location of an object and is used by computers to find visually similar objects in other pictures.

Object Detection: is a technology used to label all objects in the input picture.

Object Recognition: is a technology used to find/recognize a specific object in the input picture.

Object Localization: is the process of locating the prominent (or most visible) object in the input picture. **Classification:** is the process of identifying the category of a newly observed object based on the knowledge gather using past experiences (data sets and training models).

Category Recognition: is the process of implementing a classification algorithm to identify object categories. **Metadata:** is the description of data; used to help us organize, find, and understand the described data and computer files. Some examples of metadata include file names, file descriptions, file authors, creation and modification dates...etc.

Ranking: is the position of a search result, i.e. the place at which a page is ranked with relevance to all the other search results.

Crawling: is moving through all websites that exist on the internet, one page at a time, until all pages (including content and metadata) have been stored in the search engine's server, which is called the Search index.

Indexing: is the process of collecting, parsing, and storing data used by the search engine, i.e. adding the contents of a web page to Google. To put it into perspective, a search engine crawls the internet to index all the websites that exist.

3. Related Work

In this section, we will discuss four of the most popular implementations used for reverse image search today. These applications have been implemented by well-known companies such as eBay, Bing, Pinterest, and Alibaba.

3.1. Visual Search at eBay

The eBay has been developed the eBay ShopBot visual search. It aims to search for similar items in an eBay with accuracy and low memory footprint; to achieve that they performed category recognition, binary hash extraction, and aspect prediction in one pass through a shared deep neural network (DNN). The core services are:

1). Category recognition can recognize the accurate category of the item. For example, it can distinguish between maternity dresses and dancing dresses which is done using the residual network (ResNet), that provides a balance between accuracy and complexity. It predicts the most accurate categories of the query image. Therefore, the system search for most similar images just in a top predicted category instead of in all image in the database which reduces search load. 2). Besides a category recognition, it is using an aspect prediction model based on the shared category recognition network and share its attributes like color, brand, and style with the main DNN model. This integrates the two models into one representation which saves storage and computation time. It provides the aspects of a query image like its color, brand, and style that use in re-rank images. 3). In order to reduce storage requirements, they use a hash generation that represents the image as a binary signature instead of a real value feature vector. Therefore, they have added a layer that connects to the previous shared layer and used a sigmoid activation function that generates and outputs neural network which equals 1 if a weighted sum of its input plus a bias greater than 0.5, while it equals 0 in otherwise. The image ranking takes top predicted categories from a category recognition and image binary signature from a hash generation then matches a binary signature with the top predicted categories in an image database and generates the initial results. To improve the initial results to be more relevance, they matched between aspects of a query image and all images in initial results. Therefore, the system adds a reward point for each image that has an exact match with an aspect in a query image. So, the system rearranges image based on the final score of images. In order to handle one of the most challenges in an eBay shopBot, they built an image ingestion and indexing model which solves the problem that arises from numerous inventory image updates per second. The model computes the photo hashes for any new image and stores the image identifier for any image hashes in a Bigtable. For indexing, the system will scan all image identifiers along with their category IDs then returns a list of images identifiers for any category. Therefore, for any returned list, the system extracts the image identifier and lookup up for an image hash preceded by the same image identifier. The images hashes are saved in a binary file. While any category has its own binary file that is stored in cloud storage (F. Yang et al.,2017).

3.2. Web-Scale Responsive Visual Search at Bing

The workflow of the visual search is as follows: the user upload image or take a shot by camera, and the output will be the similar images and items; the user can choose whether to buy or to explore items. Visual search system of Bing includes three major stages:

1. Understanding the Query (image): from the query image they extract a variety of features to describe its content, including category recognition features, face recognition features, color features and duplicate detection features, deep neural network (DNN) encoders, and object detection.
2. Image retrieval: retrieves visually similar images based on the extracted features and intents.
3. Model training: several deep neural network models are held in the system to improve the relevance of the output, such as AlexNet, ZFSPPNet, GoogleNet, ResNet, also a joint k -Means

algorithm is utilized to build the inverted index in level-0 matching, and here level 0 means the small similarity with the query image. They train all networks using training dataset; the datasets are collected for certain domains, such as shopping. To supervise the DNN training, they employ more than one loss functions, such as SoftMax loss, Pairwise loss and Triplet loss.

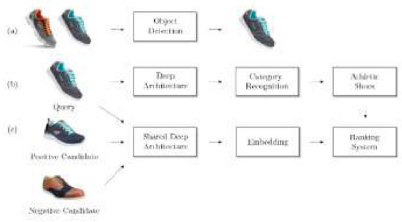


Figure 1: The application scenarios of the DNN models used in the proposed system (H. Hu et al.,2018).

Figure 1 explains three deep neural network (DNN) Models used in the visual search: (a) object detection model, they use object detection localization and the matching semantic category classification on the input query (image). (b) Item classification network, it uses a SoftMax loss: It contains thousands of categories in caps, hats such as sun hats, baseball caps and other categories. SoftMax loss is important in deep learning applications, it can map a vector to a probability of a given output in binary classification, so it will classify the item. (c) Triplet network, it uses Euclidean space, and the distances between the feature points is identified to image dissimilarity. A ranking model is named Lambdamart, it produces a final ranking score with images by taking the feature vector. The final results of similar items/images are sorted by the ranking scores, and then it returned to the user (H. Hu et al.,2018).

3.2.1. Pinterest Visual Search

Pinterest is a bulletin-board-themed social media network that allows users to create different boards for different categories that interest them. On November 2017, the company added a visual search feature into their product called “Similar Looks» that allows users to select an item within an image, by placing a border around it, and returns visually similar images/items. The main goal of this feature is to help users finding things they can’t name. Before getting into the visual search feature, it’s important to first discuss Pinterest’s incremental fingerprinting service (IFS), since the visual search feature depends on it. The IFS system extracts all features from all images on Pinterest and store the collected data. The features are then used to create a “fingerprint» for every image. That is, each image has a fingerprint that consists of all the separate objects that are contained within that image (e.g.: shoes, bags. . . etc). Every object has a box label that specifies the name of the class the object belongs in. The main drawback is that the real-time feature extractor is expensive. A suggested improvement would be to run the extractor only once every t amount of time, instead of having it run all the time. Pinterest’s Similar Looks product follows a two-step approach: object detection and localization. It utilizes object recognition technology to localize and classify fashion items (objects) in Pin images. The features are then extracted from the objects and used to generated similar looks. In other words, it shortens the task of multi-class object detection into category classification so that, instead of searching for a match in all images on Pinterest, it first fetches the images that are in the same category. This is an excellent approach because the query image is compared to images with a high probability of being similar. The prior filtering step increases the rate of true positives. The system is based on deep learning concepts, such as convolutional neural networks (CNN) and deep-learning-based object detectors. It was built using an open-source framework, called Caffe, which was also used to carry out the training (learning) process. The system's data is stored on Amazon S3, which is a product from Amazon web services. The system doesn't only rely on visual search; it combines text and visual inputs for object detection and localization. For instance, a tote and a crossbody would both be visually categorized as "bag.» The addition of descriptive metadata helps the system display more relevant results. To illustrate, if a user clicks on an image that contains a tote, the tote's back-end object box label will say "bag,» and it will consider any other object with an equivalent box label as a positive match, including other bag subcategories such as crossbody and duffle. This leaves some room for improvement. Instead of adding an additional step of filtering via text and metadata, it would be wiser to make use of every item's distinctive shape to improve search accuracy, creating a purely visual search system. However, such improvement would be extremely costly, and it would drastically slow down the system's performance, thus becoming counter-productive by hindering the user's experience. Overall, Pinterest's similar looks service is a great example that demonstrates the effective use of commercial and open-source computational platforms and tools in the field of machine learning. Adding this service shows an increase in user engagement as well as giving less-popular images a chance to be recommended to users. This proves that with the growing and overwhelming number of photos being uploaded on the internet, we'd still be able to give every image a chance to be seen (Y. Jing et al.,2015).

3.3. Visual Search at Alibaba

Alibaba is a leading Chinese e-commerce company that has online platforms which provides services from consumer to consumer (C2C), from company to consumer (B2C), and from company to company (B2B). They have developed an electronic business intelligence application called «Pailitao». Pailitao means shopping through the camera. It is an innovative image extraction product based on extensive learning and machine learning techniques on a large scale. It achieves the function of «image search» by utilizing the visual search service. Figure 2 showcases the general visual architecture.

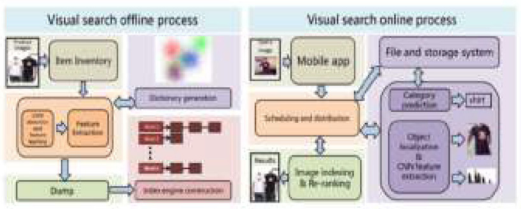

Visual search offline process
Item Inventory
Dictionary generation
Feature Extraction
Dump
Index engine construction
Visual search online process
Mobile app
File and storage system
Scheduling and distribution
Category prediction
shirt
Object localization & CNN feature extraction
Results
Image indexing & Re-ranking

Figure 2: Overview of the overall visual search architecture (Y. Zhang et al.,2018).

Category Prediction Item inventory selection, there are large quantities of items and images, so indexing is done to determine the images to filter based on shopping preferences and image quality. Due to the presence of a lot of similar elements without filtering, this leads to a bad experience for the user so remove the very similar images and improve the indexing documents. Model and search-is based fusion, the system covers categories of (dress, shoes, bags) with a visual and semantic similarity. It deals with user preferences and narrow the search space. And uses the SoftMax loss benchmark for task classification. The search part collects 200 million images for reference with a base category and used binary search to retrieve the top 30 results in a reference group.

Table 1: The strengths and weaknesses of the Visual Search at eBay, Bing, Pinterest, and Alibaba.

Implementation	Strengths	Weaknesses
Visual Search at eBay	Commercially Scalable. Reduced storage requirements	1. Restricted to top predicted classifications 2. Compromise between accuracy and latency
Visual Search at Bing	1. CPU optimization 2. Pre-calculations used to reduce time cost	1. Reduced search space 2. Compromise between accuracy and storage
Visual Search at Pinterest	1. Scalable and cost effective 2. Reduced computational cost	1. Feature extraction is not cost effective 2. Relies on metadata, not purely visual
Visual Search at Alibaba	1. Distributed infrastructure effectively handles massive data 2. Scalable and cost effective	1. Subcategories not included in the search due to noisy/wrong labels from the users on the website 2. Slower performance due to non-binary metadata storage

Image Indexing and Retrieval Large-scale search of billion-scale images uses multiple machines to store data. It sets search each subset and returns the nearest k adjacent and adds pieces dynamically to improve performance and memory. It conducts extensive testing to evaluate the performance of each system unit and uses GoogLeNet V1 model as the basic model for class prediction and learning features (Y. Zhang et al.,2018).

3.4. Critical Analysis of The Existing Implementations

All the applications that have been discussed above implement visual search in different contexts and on different devices including smartphones and desktops. Where the visual search is implemented as an additional feature rather than a core function of standalone application, this feature is meant to attract consumers to the company's website. For example, visual search at eBay only displays products available on eBay. However, none of these applications combine smartphone technology and e-commerce in one visual-search-focused application that would display results from multiple online shopping websites in a comprehensive and organized way to serve the consumer. Table 1 explores the strengths and limitations of the previously discussed implementations. The implementations were reviewed in a critical way from the back-end perspective.

3.5. Similar Mobile Applications

The top five reverse image search applications we found (based on user reviews) are Reversee, CamFind, Google Lens, Reverse Image Search, and PhotoSherlock. It is vital to note that Google Lens is not a standalone app available for download from the app store. Rather, it is an extension to the infamous search engine, Google Images, which works on the built-in iOS internet browser, Safari, and on Google's internet browser, Chrome. The main advantage shared by all these apps is that they are free to download and use. Although reverse and reverse Image Search offer additional in-app purchases, like searching multiple search engines besides Google and removing ads. We have investigated all these applications and summarized their strength and weakness in Table 1. Additionally, we have compered these applications based on their features as shown in Table 2. After studying these applications, we noticed that is these applications do not focus on online shopping. CamFind uses image recognition technology to extract descriptive text from the images and then uses the extracted text to search in the web, without using object detection to detect every item in the image. Google Lens offers object detection and has a focus on online shopping and does not allow a live shot using camera's phone.

Table 2: The strengths and weaknesses of the top five applications.

	Strengths	**Weaknesses**
Reversee	1. True results that similar/identical to the Query image. 2. Very fast. 3. f the algorithms give a false recognition it will also provide similar results compared to the query.	1. if you interested in nding one object in the image you should crop the image to that object. 2. does not provide a live shot by camera.
CamFind	1. In most cases, returns product identical for a query image. 2. When user await the result, it is give appropriate feedback for a user.	1. Return undesirable results like videos and posts.
Google Lens	1. Easy to use. 2. Provides more than one search service such as shopping. 3. Recognize the object before searching. 4. Enables the user to send comments on the application.	1. Search results for similar images are not satisfactory.
Reverse Image Search	1. Quick and easy. 2. Returns images identical to the queried image. 3. Has the option to take a photo. 4. Users can crop the queried image before searching.	1. Relies on Google Images's reverse search algorithm. 2. Does not implement an object detection algorithm.

Table 3: Comparison of The Reverse Image Search Applications Features.

	Reversee	**CamFind**	**Google Lens**	**Reverse Image Search**	**Proposed System**
App or Extension	App	App	Extension	App	App
Upload	yes	yes	no	yes	yes
Take Photo	no	yes	yes	yes	yes
Image Recognition	yes	yes	yes	yes	yes
Object Detection	no	no	yes	no	no
Check Availability Online	no	yes	no	yes	yes
No Ads	no	yes	yes	no	yes

To fill this gap, we plan on developing an application that combines as many missing features as possible. A reasonable set of features that our application should include is uploading/taking photos, object detection and finding online shopping websites that sell the user's desired item.

4. Proposed Solution

We proposed an application that does the following:

1). Upload image from gallery/Take picture: our application will have two options for getting the image from the user either by uploading the image from gallery or by taking a live photo using mobile phone's camera. 2). Image classification that includes image preprocessing (removing noise and background from an image), and extract object features. 3). Measure the similarity between the object and the objects in the database. 4). Retrieval and show similar objects.

The biggest problem people faced while using the traditional methods of search is taking a long time to find the desirable results, so one of the most important non-functional requirements in our application is the performance. This ensures that the user's time is maintained, and the results are shown in the shortest time. Ease of use is also one of the most important non-functional requirements that our application seeks to provide because our application targets users of different ages and genders. Since most machine learning algorithms are implemented in Python, the initial plan was to use a Python-based coding platform; however, there were restrictions on the usage of certain libraries for the purpose of mobile development. Due to this, a Dart-based platform is be used instead. The application builds on the most popular coding platform known as Visual Studio Code (on Windows and Mac OS), using Flutter. Flutter is an open source coding platform for building applications on iOS and Android in the Dart language. To satisfy the functional requirements previously mentioned, many libraries will be used. To perform object recognition, we use Convolutional Auto encoder and Classifier provided using Keras and Tensorow, which offer APIs and libraries. This service is used to implement the artificial intelligence aspect of the application for computer vision. Also, it used to train our application on specific training models for the purpose of online shopping. The main limitation of this coding platform is that it is in Dart, while many of the libraries used are implemented in other languages. As a result, we need to ensure that all libraries used are compatible with each other and with the operating system (iOS). Figure 3 shows the usecase diagram fro the proposed system.

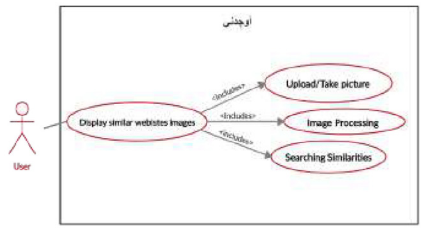


Figure 3: Usecase Diagram.

5. System Design

The design process is one of the most important processes when building an application. It focuses on visualizing the way the user interacts with the application; which, in turn, determines the quality of the user experience. It also helps the developer to implement an effective and easy way to use the application. These principals help us building a successful application and achieving user experience goals of ease, simplicity, and creating something enjoyable. Creating a friendly and understandable design is significant step for our application, as the target users include different ages and genders. We will also explain how our database will be implemented. Layered architecture diagram. Figure 4 shows the system components and how these components communicate with each other. It divides the system into three layers (3-tire). Presentation tier which represents the interface of a system that communicates directly with the user by display useful information or enable the user to insert data. Logic-tier which represents the core of a system that responsible for performing all calculations, logical operation, application function and move data between presentation tier and data tier. Data-tier contains a database of a system.

5.1. Database

Our application relies on database that contains data from websites, we have used web crawlers which are specialized programs that extract data from the Internet. Web crawlers work as follows: to get started, there must be a list of web pages to crawl. The web crawler then visits each web page in the list and downloads all the content. The downloaded page is then analyzed to identify and retrieve links. The crawler repeats the process for each link on the page

The database initially contains two tables: category and product table. Figure 5 depicts the relationships in an entity-relationship diagram, which are as follows: 1) Category: Each category must have at least one or more products. 2) Products: Each product belongs to one category. The names reflect the contents of the tables; the category table will just contain the name and the id of the category; the Products table contains the information needed for products within a website such as product name, direct URL to this product, product price and location of a product image.

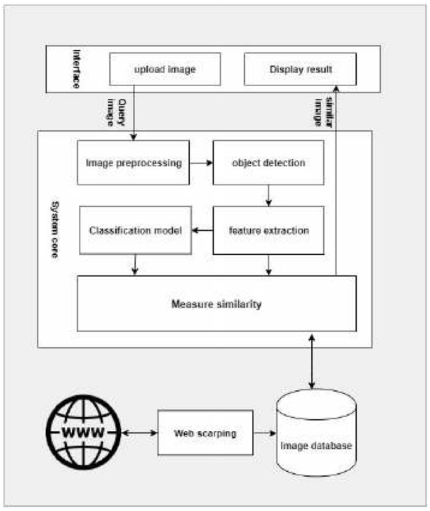


Figure 4: Layered architecture diagram.

5.2. Data Set for Machine Learning

The proposed system relies on artificial intelligence to recognize image elements and shows similarity to images. As a result, we need a data set for machine learning. A data set is a collection of data. In machine learning projects, training data sets are used to train the model for performing various actions and making better predictions to enhance the performance. Figure 6 shows how data sets are used to train a machine learning algorithm to create a predictive model.

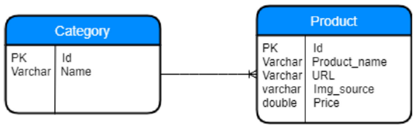


Figure 5: Entity-Relationship Diagram

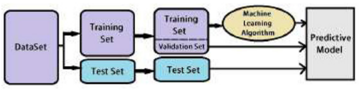


Figure 6: Data sets used to train a machine learning algorithm.

Training Data: is the part of data we use to train our model. This is the data which our model actually sees (both input and output) and learn from.

Validation Data: is the part of data which is used to do a frequent evaluation of model, fit on training dataset along with improving involved hyperparameters (initially set parameters before the model begins learning). This data plays it's part when the model is actually training. Testing Data: once the model is completely trained, testing data provides the unbiased evaluation. When we feed in the inputs of testing data, the model will predict some values (without seeing actual output). After prediction, we evaluate the model by comparing it with actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training

6. Implementation

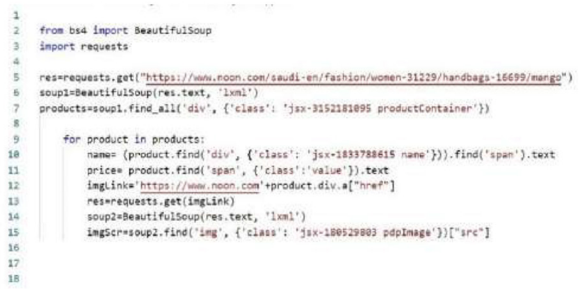
The application's implementation is divided into four parts: 1. Cloud Database using Firebase 2. User interfaces using Dart (Flutter framework) 3. Model using Python (Tensorow back-end) 4. Connection of user interfaces and classifier using Flask. The next subsections will cover the four parts of the application and the significant lines of code.

6.1. Cloud Database

In our application, building the database depends on obtaining data from different online shopping sites. So before starting discussing the structure of the database, we will show the methods that used to extract large data from the websites (web scraping). Python programming language provides a collection of libraries that support web scraping. One of these libraries is a BeautifulSoup, which we used to obtain data that will be saved in the database. Below we will explain in detail how to extract data, build database, save data and how to retrieve it.

6.1.1. Extract data (web scraping)

Before we get started web scraping, we installed BeautifulSoup and requests libraries via below commands: pip install bs4, which requests library that is used to send HTTP/1.1 request to get a web page. Below we have explained how to extract products from Noon.com. Figure 7 shows the code used in web scraping. Let's get started to explain it. In lines 2 and 3, we imported libraries that we are going to use. We used a get method in request library to retrieve HTML document for given URL. We created instance from BeautifulSoup to save returned HTML document as text. Inline 7, we used "find all method to select a div HTML element that has a class attribute with value «jsx-3152181095 productContainer». This method returned all products in a page and stored it in products variable. From line 9 to 15, we created a for loop that iterates on products to extract required data from each product.



```
1
2   from bs4 import BeautifulSoup
3   import requests
4
5   res=requests.get("https://www.noon.com/saudi-en/fashion/women-31229/handbags-16699/mango")
6   soup1=BeautifulSoup(res.text, 'lxml')
7   products=soup1.find_all('div', {'class': 'jsx-3152181095 productContainer'})
8
9       for product in products:
10          name= (product.find('div', {'class': 'jsx-1833788615 name'})).find('span').text
11          price= product.find('span', {'class':'value'}).text
12          imgLink='https://www.noon.com'+product.div.a["href"]
13          res=requests.get(imgLink)
14          soup2=BeautifulSoup(res.text, 'lxml')
15          imgScr=soup2.find('img', {'class': 'jsx-180529803 pdpImage'})["src"]
16
17
18
```

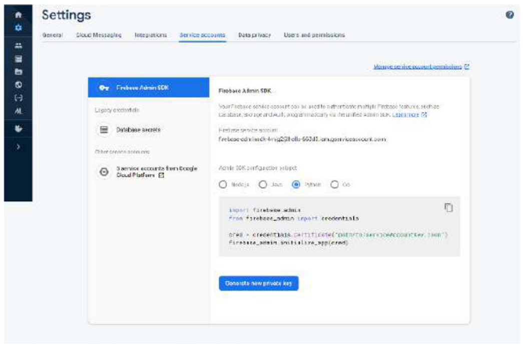
Figure 7: Web scraping for noon website.

6.1.2. Build database

In our application, we stored data in a Cloud Firestore which is a NoSQL database provided by Firebase and Google Cloud Platform. Before we start using the database, we must make sure that we have a server app which means our Flutter application connects successfully with Firebase project. After that, we need to do the following steps: 1) Install the Firebase admin SDK via this command: pip install - user _rebase -admin. 2) Initialized the Cloud Firestore SDK on our server by go to our Firebase project » setting » service accounts as shown in Figure 8. Click on Generate new private key and save the JSON file in application folder. Also, copy the code in a gray box into python file. 3) After that, we have created a database and required collections in our Firebase project by selecting database » cloud firestore » start collection.

6.1.3. Insert data

Cloud Firestore stores data in "Documents", which are stored in "Collections". So, the first step before inserting data into Cloud Firestore is to store data into a dictionary (associative array) as shown in line 78 in Figure 9. After that, we have created a document in a specific collection as shown in line 86 and stored the references of the document in ref variable; which in turn is used a set method to insert data dictionary into a database.



Firebase Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase features, such as Database, Storage and Auth, programmatically via the unified Admin SDK. Learn more

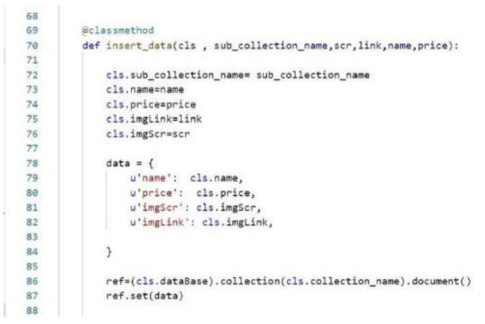
Firebase service account
firebase-adminsdk-4mjq2@hello-663d3.iam.gserviceaccount.com

Admin SDK configuration snippet
Node.js Java [✓ Python] Go

```
import firebase_admin
from firebase_admin import credentials

cred = credentials.Certificate("path/to/serviceAccountKey.json")
firebase_admin.initialize_app(cred)
```

Figure 8: Firebase admin sdk.

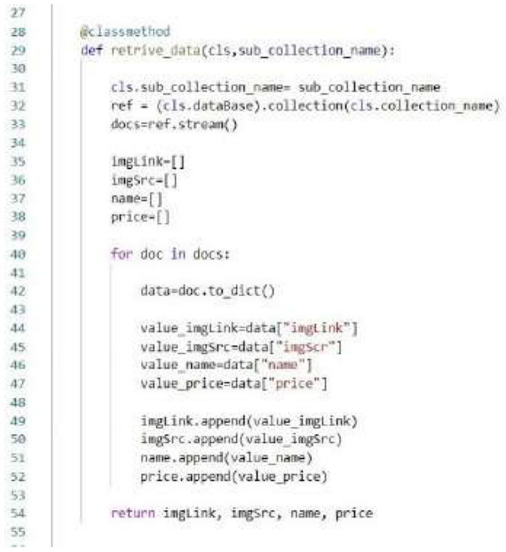


```
68
69    @classmethod
70    def insert_data(cls , sub_collection_name,scr,link,name,price):
71
72        cls.sub_collection_name= sub_collection_name
73        cls.name=name
74        cls.price=price
75        cls.imgLink=link
76        cls.imgScr=scr
77
78        data = {
79            u'name':  cls.name,
80            u'price':  cls.price,
81            u'imgScr': cls.imgScr,
82            u'imgLink': cls.imgLink,
83
84        }
85
86        ref=(cls.dataBase).collection(cls.collection_name).document()
87        ref.set(data)
88
```

Figure 9: Inserting data.

6.1.4. Retrieve data

In order to retrieve all documents in a specific collection, we store the references of the collection in a ref variable as shown inline 32 Figure 10. Then, we use stream method to retrieve all document in a collection and store it's in a docs variable. Then, we have created a "for loop" that iterates on docs to extract required data from each document as shown in line 40 to 52,.

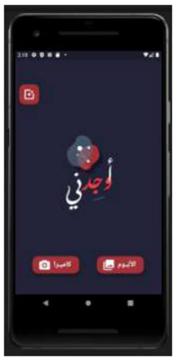


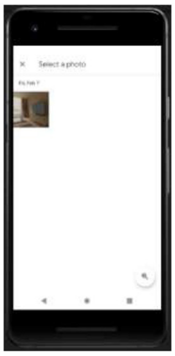
```
27
28      @classmethod
29      def retrive_data(cls,sub_collection_name):
30
31          cls.sub_collection_name= sub_collection_name
32          ref = (cls.dataBase).collection(cls.collection_name)
33          docs=ref.stream()
34
35          imgLink=[]
36          imgSrc=[]
37          name=[]
38          price=[]
39
40          for doc in docs:
41
42              data=doc.to_dict()
43
44              value_imgLink=data["imgLink"]
45              value_imgSrc=data["imgScr"]
46              value_name=data["name"]
47              value_price=data["price"]
48
49              imgLink.append(value_imgLink)
50              imgSrc.append(value_imgSrc)
51              name.append(value_name)
52              price.append(value_price)
53
54          return imgLink, imgSrc, name, price
55
```

Figure 10: Retrieving data.

6.2. User Interfaces

Figure 11 shows the first page when the application start, and Figure 12 shows the second page which is for displaying the results.

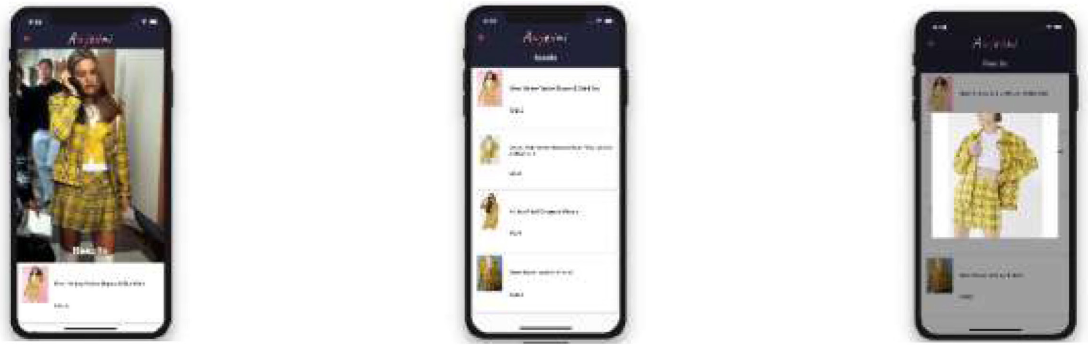




a) Home Page b) Gallery button clicked

Figure 11: Screenshots of the Home Page

Figure 12 shows the results page, which appears to the user after the visual search has been completed. It displays the websites that sell visually similar products, expandable preview of the product is available.



a) Results page

b) User scrolls to display more results

c) Expandable preview

Figure 12: Screenshots of the Results Page

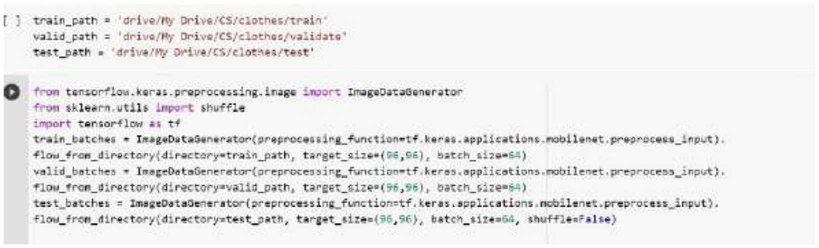
6.3. Building Convolutional Neural Networks Model (CNN)

In building CNN model, our focus was on two significant factors: the similarity and the size of the feature vector (FV) as we are running in a mobile environment. We have tried multiple procedures and different datasets on different CNN models to choose the best based on these two factors, we have tried mobileNet without training, mobileNet with Fashion MNIST training, Autoencoder with colored datasets and Fashion MNIS dataset, and lastly with different number of layers of each model. We will discuss in detail the last model which we have implemented in our application. In order to classify and extract features from an image, we have used transfer learning to train and modify a pre-trained model in Keras mobile net. Transfers learning means untrained the convolutional layers that are closer to the input layer (layers freezing) and remove the last layers. In most cases, two to three fully connected layers are added after the freezing layers, which are used in training and modify the model. In our project, we have trained and tested the model on 5025 images that belong to 10 classes (Ankle boot, Bag, Coat, Dress, Pullover, Sandal, Shirt, Sneaker, T-shirt, Trouser), which are obtained from different web pages through web scraping. In following subsection, we will explain how to prepare data, build the model, train and test the model and measure similarities between images.

6.3.1. Preparing the Data

The dataset was divided into three sets: train, validate, and test, where each set has 10 classes. As seen in Figure 13, the directory of each dataset was assigned to the variable. After that, the ImageDataGenerator.ow function from directory() was used to split any dataset into a batch of images (a batch refers to the number of images the model can see at a time). Moreover, the preprocessing function parameter is used to do preprocessing for all images.

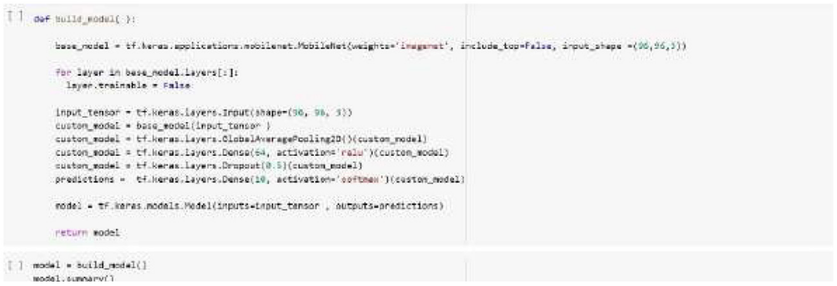
6.3.2. Build the model



```
[ ] train_path = 'drive/My Drive/CS/clothes/train'
    valid_path = 'drive/My Drive/CS/clothes/validate'
    test_path = 'drive/My Drive/CS/clothes/test'

    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from sklearn.utils import shuffle
    import tensorflow as tf
    train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).
    flow_from_directory(directory=train_path, target_size=(96,96), batch_size=64)
    valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).
    flow_from_directory(directory=valid_path, target_size=(96,96), batch_size=64)
    test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input).
    flow_from_directory(directory=test_path, target_size=(96,96), batch_size=64, shuffle=False)
```

Figure 13: Preparing the data.



```
[ ] def build_model( ):

      base_model = tf.keras.applications.mobilenet.MobileNet(weights='imagenet', include_top=False, input_shape =(96,96,3))

      for layer in base_model.layers[:]:
        layer.trainable = False

      input_tensor = tf.keras.layers.Input(shape=(96, 96, 3))
      custom_model = base_model(input_tensor )
      custom_model = tf.keras.layers.GlobalAveragePooling2D()(custom_model)
      custom_model = tf.keras.layers.Dense(64, activation='relu')(custom_model)
      custom_model = tf.keras.layers.Dropout(0.5)(custom_model)
      predictions =  tf.keras.layers.Dense(10, activation='softmax')(custom_model)

      model = tf.keras.models.Model(inputs=input_tensor , outputs=predictions)

      return model

[ ] model = build_model()
    model.summary()
```

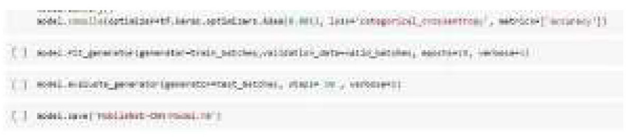
Figure 14: Building the model.

To build the model, we have created an instance from the mobile net model which is previously trained on the ImageNet dataset without a top fully connected layer. We have frozen all layers in the base model to keep it unmodifiable. Instead of the top fully connected layers that have been removed, we have added a new fully connected layer that is used to modify the model. So, only the last few layers will train on the new dataset.

6.3.3. Train and test model

To train the model, we call "compile" method that takes three different training parameters:1) The loss function that evaluates the model prediction. So, if the prediction veers too much from the actual results, the value of a loss function will increase. 2) An optimizer which uses to minimize loss. 3) Metric that uses to evaluate the performance of the trained model.

To start the training process, We call "fit_generator()" function that takes a training batch, validating batch and epochs, which are specifing how many times the model passed on all batches in a dataset. Also after training, we have used the "evaluate_generator()" function to test the model. Finally, we have saved the model in binary file to reuse it. After we have built and trained the model as a classifier, we have started looking for a way to find similarities between the images. We found that the ideal method to find similarities between images is to extract the feature vector, and then use a metric to calculate the error rate. So, we have adopted this method in our application. After extracting feature vector, we have used principal component analysis (PCA) that improves the speed of similarity search by reducing the length of feature vectors. Also, we have used scikit-learn machine learning library for finding nearest neighbors of a query image by comparing the feature vector of a query image to all feature vectors of the images in the database that have the same category of a query image. We have created a nearest neighbors model and used a brute-force algorithm to train the model to find the nearest neighbors by calculating the distance between the vectors using Euclidean distance.



```
    model.compile(optimizer=tf.keras.optimizers.Adam(0.001), loss='categorical_crossentropy', metrics=['accuracy'])

[ ] model.fit_generator(generator=train_batches,validation_data=valid_batches, epochs=[illegible], verbose=1)

[ ] model.evaluate_generator(generator=test_batches, steps=[illegible], verbose=1)

[ ] model.save('[illegible]-CNN-model.h5')
```

Figure 15: Train and test model.

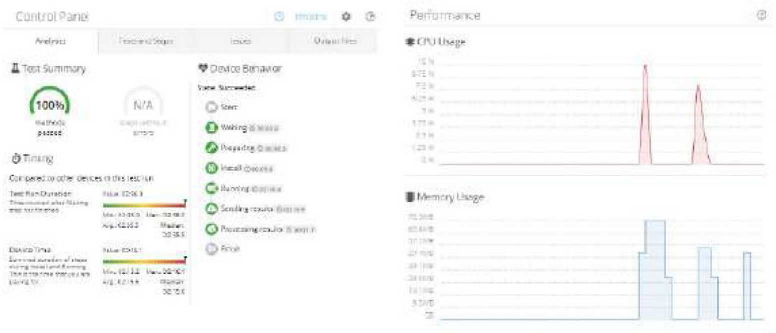


Figure 16: Performance Testing results.

The system has been successfully tested. We have tested several units of the application separately, then performed integration testing to test the associated units together to check their performance. System testing is performed with all possible scenarios to check the interaction between all the system's units. Finally, performance testing is done that examines responsiveness, stability, scalability, reliability, speed, and resource use. Bitbar tool has been used in testing the application.

6.4. Experimental Analyses

Analysis and Visualization of Classifier Performance: Measuring Classifier Performance is a critical step that shows if a classifier better enough or needs further development. There are multiple ways for measuring the classifier's performance. In our project, we look at the confusion matrix as a much better way of evaluating the performance and presenting it in a simple way. The confusion matrix also provides basic parameters that can be used to derive other performance measurements. Figure 17 shows the confusion matrix for our classifier. The figure shows that there are 10 possible predicted classes: Ankle boot, bag, coat, dress, pullover, sandal , shirt, sneaker, and trouser. The total number of prediction (the size of test data) is 1668. 200 (sum of bag row) out of 1668 are bags, 195 of them have been predicted correctly. The confusion matrix is a useful method that is used to compute the accuracy and misclassification rate for a classifier:

- Accuracy: which compute by sum the values of cells in a main diagonal and divide it by the sum of all cells
  Accuracy = $\frac{1254}{1668} \times 100 = 75\%$
- Misclassification rate (error rate) : it is a sum of all cells minus the sum of cells in the main diagonal divided by the sum of all cells
  Error rate = $\frac{1668-1254}{1668} \times 100 = 25\%$

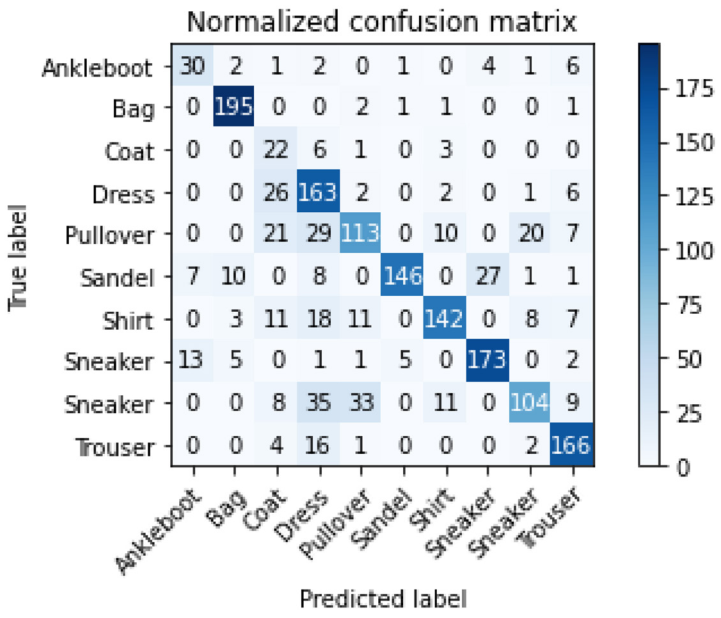


Figure 17: Confusion Matrix.

Program Execution Time : one of the most important factors that affects user satisfaction is the response time. When the time taken to execute a program is little, the response time is decreased. Considering reverse image search applications, the time-consuming becomes a more critical factor because reverse image search applications use different algorithms to reprocess, recognize an object, and search for similar images; all these processes usually take a long time. In our application, we have used different algorithms including principal component analysis and nearest neighbors in order to reduce execution time to 2 seconds.

Development Cost of Awjedni Application. Awjedni has not been published on any platform yet, we can say that the development cost is zero (until now). In general the cost in software development depends on a variety of different factors:

- Complexity and number of features: Awjedni provides two features: recognizing photos, determining the full information of the photo to the user such as the price, source of the photo.
- Complexity of UX/UI design: We use a simple design containing three colors, the logo and the name of the app both show exactly the purpose of the app.
- Development approach: Awjedni is a cross-platform Application.
- Back-end infrastructure: Awjedni uses two APIs.

In order to estimate the cost if we published the application online we should encounter important factors: App infrastructure services:

The Database: we are using Cloud database free plan due to this it has a certain usage limit, if we want to extend and add additional unlimited features we must move for paid plans, paid plans prices depend on the usage, for example GB storage price start from $0.026/GB.

Publishing The App: Awjedni is a cross-platform application due to this we can publish it in multiple platforms. iTunes require 99$ every year, Google store require one-time payment of $25, the sum of all is 124$.

AI infrastructure. For higher accuracy in classifying products we need to train the network using powerful cloud GPUs, it starts from $0.35 per GPU.

7. Conclusion and Future Work

Awjedni is a reverse-image search application built for iOS and Android smartphones; it is meant to help people searching for items, they can't name it, using photos. Users can either upload a photo from the phone's camera roll or take a photo of an item using the camera. The main functionalities include object detection and web crawling. Awjedni was built using VS-Code as the programming environment in the Dart and Python languages. The frameworks used in the implementation are Flask, Keras, TensorFlow, and Firebase. In the near future, we plan to expand the application to allow multiple object detection per image. We also plan on broadening the range of search to include non-fashion-related items such as stationary supplies, skin care products, and makeup. It would be interesting to evolve the application to be more social-media friendly by adding user profiles where users can save their favorite looks and can receive automated recommendations based on his/her recently searched item.

8. References

Wiley, Victor, and Thomas Lucas. «Computer vision and image processing: a paper review.» International Journal of Artificial Intelligence Research 2.1 (2018): 29-36.

Patel, P., & Thakkar, A. (2020). The upsurge of deep learning for computer vision applications. International Journal of Electrical and Computer Engineering, 10(1), 538.

Hatcher, William Grant, and Wei Yu. «A survey of deep learning: Platforms, applications and emerging research trends.» IEEE Access 6 (2018): 24411-24432.

F. Yang, A. Kale, Y. Bubnov, L. Stein, Q. Wang, H. Kiapour, and R. Piramuthu, "Visual search at ebay,» in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 2101-2110.

H. Hu, Y. Wang, L. Yang, P. Komlev, L. Huang, X. S. Chen, J. Huang, Y. Wu, M. Merchant, and A. Sacheti, “Web-scale responsive visual search at bing,» in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’18. New York, NY, USA: ACM, 2018, pp. 359-367.

Y. Jing, D. Liu, D. Kislyuk, A. Zhai, J. Xu, J. Donahue, and S. Tavel, “Visual search at pinterest,» in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’15. New York, NY, USA: ACM, 2015, pp. 1889-1898.

Y. Zhang, P. Pan, Y. Zheng, K. Zhao, Y. Zhang, X. Ren, and R. Jin, “Visual search at alibaba,» in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’18. New York, NY, USA: ACM, 2018, pp. 993-1001.