Technical Report on the Quiz-based Matching Algorithm (Public Version)

Quang Tran*

Minerva - SK Encar Team

December 25, 2018

1 Executive Summary

This paper explains the features of a quiz-based matching algorithm powering the car recommendation web application developed in a project hosted by Minerva Schools at KGI and its civic partner SK Encar. The quiz which the algorithm is based upon asks for the users' age, gender, wage, preferred country of origin of the car's brand, car type, size, and environmentally friendly indicator (electric/hybrid or not). We used the data set on SK Encar website's search logs provided by the company to infer car preferences

*Email quang.tran@minerva.kgi.edu

of certain demographic groups. The engine uses the K-prototype technique in clustering the data, and suggest to the users five cars on the cluster that is closest to their inferred preferences. The engine's strengths and limitations are also discussed.

2 Introduction

The quiz which the algorithm is based upon asks for the users' age, gender, wage, preferred country of origin of the car's brand, car type, size, and environmentally friendly indicator (electric/hybrid or not). The answer to each question is categorical, except that the answer for the wage question is treated as continuous by the algorithm. For question on the preferred brand, there are four options: Korea, Germany, Japan, and Luxury. It is obvious that the Luxury option and the other three are not mutually exclusive. Therefore, we assumed in the algorithm that if the answer is Luxury, the user does not care about the nation of origin as much as the cars' prices and cars with higher prices are deemed more suitable with the user. We encountered the same problem with the options for the car type question (Sport, SUV/RV, and Normal). The normal category is about size and is not mutually exclusive with the other two options, so we made the same interpretation as with the question on preferred brand: if users indicate that they prefer 'normal' cars, size matters to them more than whether the cars. As for the car size, three categories are considered: Small/Compact, Middle, and Big.

3 Data Processing

Two data files are attached with this report, `car_data.csv` and `demo_data.csv`, both of which were provided by SK Encar. `car_data.csv` provides attributes of models of cars, while `demo_data.csv` includes information on search logs according to cars' models and users' demographic information.

As the file `car_data.csv` does not contain any information on how different demographic groups prefer car models differently (note that the columns `Gender` and `Age` in the file do not reflect this type of information; they are just remnants from some preprocessing data from before), we exploited the file `demo_data.csv` for this information. First, we add a column `age_approx` to dataframe read in by the pandas package to capture the age groups in the column `Age` by the first digit of the age. For example, any age group starts with 3 ("30-34" or "35-39") in column `Age` will be translated to just 30 in column `age_approx`. This transformation is to be in line with the categories asked in quiz question.

```
age_approx = []
for index, row in demo_df.iterrows():
    if row['Age'][0]=='~':
        age_approx.append('0')
    else:
        age_approx.append(row['Age'][0]+'0')
```

```
7
8 demo_df = demo_df.assign(age_approx=pd.Series(age_approx).values
      )
9 demo_df = demo_df.drop(demo_df[demo_df.age_approx == 'A0'].index
      )
```

Listing 1: Add age groups

Next, we considered all possible combinations of genders and age groups (e.g., `(M, 30)`, `(F, 20)`, `M,50`) and for each of those demographic combinations, we compute the mean of all search logs (numerical values in columns from `Week 14` to `Week 39`) for each car model. The result is stored in the variable `table`. A view of part of the printed `table` is provided below.

```
{(30, 'M'): [('HG', 26.9),
  ('W212', 21.4),
  ('A8', 17.72),
  ('2-Series', 16.94),
  ...
```

Listing 2: `table`. The model strings and mean search logs are fabricated for demonstration

The printing above tells us that for the group of 30-year-old males, their mean search logs for HG is 26.9, for W212 is 21.4, and so on. The higher mean search log is assumed to be higher preference for the corresponding model. We then transformed the means to integers ranging from 0 to 2 according to the following rule: the top 1/3 highest means become 2, the next 1/3

become 1, and the rest 0. Thus, 0 indicates that a model is least preferred by a group, and 2 indicates the most preferred models. An example of the view into the integer transformed `table` is provided below.

```
{(30, 'M'): {'2-Series': 0,
'A8': 1,
'W212': 1,
'HG': 2
...
```

Listing 3: `table` when mean search logs are converted to integers. Data are fabricated for demonstration

We also added an indicator for the luxury of each model (the column `luxury_weight`), as one of the question in the quiz has "luxury" as an option. Intuitively, the more expensive the car, the more luxury it is. We used the same one-third heuristic: the top 1/3 most expensive cars have a luxury weight of 2, the next 1/3 expensive cars 1, and the rest 0.

```
car_df['Price_rank'] = car_df['Price'].rank(ascending=True)
luxury_weights = []
for index, row in car_df.iterrows():
    if row['Price_rank'] < len(car_df.index)/3:
        luxury_weights.append(0)
    elif row['Price_rank'] < 2*len(car_df.index)/3:
        luxury_weights.append(1)
    else:
        luxury_weights.append(2)
car_df = car_df.assign(luxury_weight=pd.Series(luxury_weights).
```

```
    values)
```

Listing 4: Add luxury weights

4 Engine Description

4.1 K-means Clustering Overview

The idea behind the entire engine is finding car models that have the feature set most similar to that specified by the user. We do this by first grouping the given data on the car models into clusters, then suggesting cars belonging to the cluster closest to the data point determined by the users' answers. We first give an overview of the K-means clustering technique, which is predominantly applied to numerical data, and of the K-modes clustering, which is used for categorical data. Finally, as our datasets include both numerical and categorical data, we review the K-prototypes, a hybrid of K-means and K-modes, and demonstrate how we used K-prototypes in the engine.

We will use the mathematical notations and preliminaries described in Huang (1997) for consistency in the descriptions of the three algorithms. Let $X = \{X_1, X_2, ..., X_n\}$ be the set of n instances in our data, and $\mathbf{X_i} = (x_{i1}, x_{i2}, ..., x_{im})$ be a vector of attributes of instance X_i. Suppose our data contains solely numerical values, then x_{ij} is numerical. Let a positive integer K be the desired number of clusters. Let C_k be a set of indices of the instances in cluster k.

For example, if X_i is in cluster k, then $i \in C_k$. Cluster k's representative vector (or centroid, as used in James, Witten, Hastie, & Tibshirani (2013)), is vector $\mathbf{Q_k} = (q_{k1}, q_{k2}, ..., q_{km})$, where q_{ki} is the mean of all values of attribute i of instances in cluster k. The goal of clustering the data into k groups is to minimize the following cost function E:

$$E = \sum_{k=1}^{K} \sum_{i \in C_k} d(\mathbf{X_i}, \mathbf{Q_k}) \tag{1}$$

where d is most commonly chosen to be the squared Euclidean distance (James et al., 2013):

$$d(\mathbf{X_i}, \mathbf{Q_k}) = \sum_{r=1}^{m} (x_{ir} - q_{kr})^2 \tag{2}$$

The K-means technique is a greedy approach that finds a local optimum to the K-means optimization problem defined by the cost function E. The algorithm follows the following steps, as suggested by Huang's paper:

(1) Randomly select k centroids

(2) For each instance in the data, assign it to the cluster corresponding to the nearest centroid. The distance is computed according to equation (2). Update the centroid after the assignment

(3) For each instance in the data, if the nearest centroid to it is not the one corresponding to its current cluster, assign it to that nearest centroid's cluster and update both of the two clusters' centroids

(4) Repeat step (3) until no further new assignment takes place.

4.2 K-modes Clustering Overview

K-modes clustering is an extension to the K-means technique to deal with clustering categorical data. In K-modes clustering, the representative vector $\mathbf{Q_k}$ is defined to be $(q_{k1}, q_{k2}, ..., q_{km})$ where q_{ki} is the mode of all values of attribute i of instances in cluster k. The cost function E defined in (1) is still the same, with the function d being altered to be:

$$d(\mathbf{X_i}, \mathbf{Q_k}) = \sum_{r=1}^{m} \delta(x_{ir}, q_{kr}) \tag{3}$$

where $\delta(a, b) = 0$ if $a = b$ and $\delta(a, b) = 1$ if $a \neq b$.

The algorithm for K-modes is exactly the same as that for the K-means technique, with the notion of "centroid" being replaced by that of "the representative vector."

4.3 K-prototypes Clustering Overview

K-prototypes technique is a combination of both K-means and K-modes algorithms and clusters data with both numerical and categorical variables. The cost function E stays the same, and the function d becomes

$$d(\mathbf{X_i}, \mathbf{Q_k}) = \sum_{r=1}^{m_l} (x_{ir}^l - q_{kr}^l)^2 + \gamma_k \sum_{r=1}^{m_p} \delta(x_{ir}^p, q_{kr}^p) \tag{4}$$

m_l and m_p being the numbers of numerical and categorical attributes of any instance, respectively. x_{ir}^l and q_{kr}^l are values of numerical attributes. x_{ir}^p

and q_{kr}^{p} are values of categorical attributes. γ_k is weight of the categorical attributes for cluster k.

The elements of the representative vector of any cluster are either the modes or the means of the attributes of all instances in that cluster. The algorithm for the K-modes technique is exactly the same as that of the K-modes one.

4.4 K-prototypes Used in the Engine

We did not use all the attributes of a car asked in the quiz in clustering data. This is because we distinguished two groups of attributes. The first includes age, gender, and preferred brand, while the second group contains wage, car type, car size, and environmentally friendly indicator. Attributes in the secondary group were used to cluster data.

We had to divide into these two groups because, unlike attributes in the second group, which each take one unique value, those in the first group take multiple ordered values. For example, if a user specify his age group and gender as '30s' and 'M', there is no car model with attribute 'age group' taking the value '30s' and attribute 'gender' taking the value 'M', but rather, each car has an attribute 'age group-gender' with multiple values (ranging from 0 to 2) depending on the specified pair age group - gender. Similarly, users may specify their preferred brand as "luxury", in which case, depending on a car's price, the luxury weights can take any integer from 0 to 2. Note that this is different from the attributes in the second group. Take the attribute car size for example. If a user says she wants a small car, then there are

many instances of car models which have the attribute 'car type' taking the value 'SMALL/COMPACT".

As such, the car filtering process is divided into two stages. The first stage uses the first group of attributes, and the second stage uses the second group.

4.4.1 The First Stage

The first stage uses three answers from users: their age groups, their genders, and their preferred brand.

The variable `table` contains the level of preference expressed by every age-gender group toward every car model, as shown in listing 3. According to listing 3, for the demographic group 30-M (aged 30s and male), the model "2-Series" has level 0 (least preferred), "A8" and "W212" get 1 (moderately preferred), and "HG" gets 2 (most preferred). Therefore, if the answer from the users indicated that they belong to the 30-M group, the model "2-Series" is awarded 0 score, the "A8" and "W212" each are awarded 1, and the "HG" is awarded 2. The code to implement this is printed below.

```
1 if row['Model'] in table[(int(age),gender)]:
2             score += table[(int(age),gender)][row['Model']]
```

Listing 5: Match gender and age group

For the question about preferred brand, if the user answers "Luxury", then each car model will be awarded a score equivalent to the luxury weight. This score is accumulated toward the age-gender score above. If the user answers a country of origin instead (e.g., "Korea"), then cars of that origin is awarded

1 point.

```
if brand != 'Luxury':
  if row['Country'] == brand:
    score += 1
else:
  score += row['luxury_weight']
```

Listing 6: Match preferred brand

4.4.2 The Second Stage

The second stage makes use of the answers to questions about wage, car type, car size, and environmentally friendly indicator. For the wage question, four options are available, corresponding to four groups of wage: A) 15 million and below, B) 15-30 million, C) 30-60 million, and D) above 60 million. We translated these four options to the corresponding car prices, using some statistical analysis. Specifically, option A corresponds to car price less than 15,000,000, option B 15,000,000-40,000,000, option C 40,000,000-70,000,000, and option D above 70,000,000. Finally, as we treat answers to this question as numerical ones, rather than categorical, we take the medium value in each category and store it in the variable `budget`.

```
if wage == '15':
        budget = 1500/2
elif wage == '15-30':
        budget = (1500+4000)/2
elif wage == '30-60':
```

```
        budget = (4000+7000)/2
else:
        budget = (7000+16931)/2
```

Listing 7: Match preferred brand

Question about car type has three options: A) Normal, B) SUV/RV, and C) Sport. Question about car size has three options: A) Small/Compact, B) Middle, and C) Big. In the file `car_data.csv`, we only have one column (column `car type`) that incorporates all these options (`SUV/RV`, `SPORT`, `SMALL/COMPACT`, `MIDDLE`, and `BIG`). To combat this, we make the following rules and assumptions:

1) If the user picks "A) Normal" for the question about car type, we assume they do not care about whether the car is of type "B) SUV/RV" or type "C) Sport" as much as its size, so response to the question about car size is considered

2) Alternatively, if the user picks either "B) SUV/RV" or type "C) Sport" for the question about car type, the question about car size is disregarded altogether. The value for the column `Car type` in `car_data.csv` is taken to be either "SUV/RV" or "SPORT" depending on whether the user picks B or C, respectively. This value is stored in the variable `net_car_type`

3) If the question about car size is considered, the value for the column `Car type` in `car_data.csv` is taken to be either "SMALL/COMPACT", "MIDDLE", or "BIG" depending on his/her answer. This value is stored in the variable `net_car_type`

```
if car_type != 'NORMAL':
  net_car_type = car_type
else:
  net_car_type = small_car
```

Listing 8: Match car type

For the K-prototypes algorithm, we set the number of clusters to be 5, treating the column `Price` in `car_data.csv` as numerical data, and the columns `Car type` and `EV/Hybrid available` as categorical. After training to get the clusters and their members according to the algorithm in Huang's paper using the package `kmodes`, the set of (converted) answers from users (`budget`, `net_car_type`, and `hybrid`) are used to find the closest cluster. As the cluster may contain more than five cars and we only want five cars in our recommendation list, we choose five cars with the highest scores earned in the first stage to recommend.

```
X_train = df[['Price','Car type', 'EV/Hybrid available' ]].
    values
kproto = KPrototypes(n_clusters=5, init='Huang', verbose=0)
clusters = kproto.fit(X_train, categorical=[1,2]).labels_

X_test = np.array([[budget, net_car_type, hybrid]])
X_df =  pd.DataFrame(X_test)
X_df[[0]]=X_df[[0]].astype('float')
X_test = X_df.values
labels = kproto.predict(X_test, categorical=[1,2])
```

```
10 fil_mask = df.assign(cluster=clusters)
11
12 fil_mask = fil_mask[fil_mask.cluster==labels[0]].drop(axis=1,
       columns='cluster')
13
14 return fil_mask.nlargest(num_rec, 'scores', 'first')
```

Listing 9: K-prototypes in use

5 Discussions

One strength of the algorithm is it incorporates rich data provided by SK Encar on demographic information in inferring car preferences of certain population groups. It also ensures that users are always recommended five cars that are similar to each other (similarity is measured by the distance from the prototypes, following K-prototypes technique) and have the highest demographic matching scores.

The only way to understand the performance of the engine in practice is to take feedback from users. Improving the engine by incorporating the feedback can be done by looking for the discrepancies between users' dissatisfactions and how the engine is implemented. Potential features of the engine that can be fine-tuned and experimented with include, but is not limited to:

- The scores assigned for the match in demographic information and car brand ("Germany", "Korea", "Luxury", etc.). For a crude example, if

it is evident that many more users complain about the suggested cars' country of origin than about the level of "luxury", then we can consider decreasing the luxury weights or increasing the demographic scores.

- The number of clusters in the K-prototypes algorithm
- The map from the indicated wage to the cars' price
- The extent to which the original data set is reduced after the first stage of filtering. Currently, it is reduced to one half, but this figure could be changed.
- Considering moving some attributes in the second stage to the first stage with a regular score accumulating scheme

The engine also comes with its own limitations, with the major one lying with how the engine can be improved.

- The only way to assess how well the algorithm matches the users with the cars in the database is to conduct a survey (as opposed to having a ground-truth data set to make comparisons)
- There is no way to automate incorporating feedback from users to improve the engine
- The score for every match in brand is arbitrarily set to 1 and requires further experiments to confirm its validity.

In conclusion, the only way to understand the engine's performance is to get feedback from users and incorporating this information from users to improve the engine can be cumbersome, not be done in a systematic way, and require potentially many experiments. We suggest taking a machine learning spin on this matching problem in the long run by first collecting data, which is certainly non-trivial but facilitates automation in the future.

Reference

[1] Huang, Z. (1997, February). Clustering large data sets with mixed numeric and categorical values. In Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD) (pp. 21-34).

[2] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). New York: springer.