Market Basket Clustering using Cuisines Recipes Dataset

Marian Alexandru Diaconu
203316
UniTN, 2nd year CS master's degree
Do not know yet
marian.diconu@studenti.unitn.it

Valentino Armani
203290
UniTN, 2nd year CS master's degree
Industry intention
valentino.armani@studenti.unitn.it

ABSTRACT

Nowadays, identifying different types of customers has become a very important task for companies. Many data mining methods have been developed and adapted to solve this task. This paper presents an approach to deal with this problem in the context of food market purchases modeled as market baskets. A dataset of recipes is used to preprocess the baskets and compute score features representing the basket membership level in each cuisines. The goal is to identify customers types based on the food they eat. In order to do that, the baskets are clustered based on their scores instead of on the values of the items. The method is analyzed and compared with a baseline in order to highlight differences, strengths and weaknesses. Finally, scalability analysis is performed and presented.

CCS CONCEPTS

• **Information systems → Clustering**; • **Mathematics of computing → Cluster analysis**; • **Computing methodologies → Model verification and validation**.

KEYWORDS

market basket dataset, unsupervised learning, clustering, tf-idf, dbscan, k-means

ACM Reference Format:
Marian Alexandru Diaconu and Valentino Armani. 2018. Market Basket Clustering using Cuisines Recipes Dataset. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Data mining has become a fundamental science in the 21st century because allows companies to search for recurring patterns about their customers in the data which may give insights about business strategies. However, collecting data about customers is not always an easy task. Based on the type of company and on the type of relationship that is created between the company and the customer, the digital assets that interconnect the customer with the company may be able to collect different amounts of information. One of the easiest ways for a retailer to collect data is to keep track of customers' purchases. Suppose a company is able to collect such data, the information would be stored as a table, where each row represents a purchase and each column a product of the purchase. This type of input format is also called Market Basket model. Under this model, a purchase is represented by a basket and a product by an item. Note that the tabular representation of a tabular market basket may be very sparse, that is many baskets may contain few items, hence this may not be ideal for some data mining techniques. In this paper we will focus on food items, therefore a basket may contain elements such as *milk, bread, beer.* Under these assumptions, a retail seller would like to use data mining techniques to identify types of customers based on the food they eat. This may be challenging for many reasons. First because the same ingredient may be used in different recipes or for the same recipe different ingredients may be used. Furthermore, a customer may buy only some of the elements he/she needs (because he/she may have the rest at home) which means that it is not clear what recipe a person wants to make when buying a specific item. This study analyzes a food market basket dataset considering also a cuisines recipes dataset. The Cuisines Recipes dataset is composed by many clusters, each representing a type of cuisine such as *Italian, British, Mexican.* Each cluster can be modelled as a market basket itself, where each basket represents a recipe and each item an ingredient of the recipe. Basically the cuisines recipes dataset can be used to understand the likelihood of a purchase to be related to the cuisines. Finally, the main goal of this paper is to cluster the purchases in such a way that different types of customers emerge. Section 4 will introduce one possible approach to this problem using a combination of DBSCAN and Term Frequency-Inverse Document Frequency (TF-IDF).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/1122445.1122456

2 RELATED WORK

This chapter will describe some techniques that may be used to preprocess and further cluster a text-based market basket dataset such as the one provided by the retailer. This is done also to better understand the utility of the cuisines recipes dataset.

2.1 Jaccard Similarity Coefficient and N-Shingles

The Jaccard Similarity Coefficient and N-Shingles are two techniques that can be used to measure similarity between arbitrary long strings. The Jaccard Similarity Coefficient or Jaccard Index is a statistic used for gauging the similarity and diversity of sample sets. The Jaccard coefficient measures the similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets [8].

$$JacccardIndex(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

On the other hand N-Shingles or N-Grams is a natural language processing technique which decomposes a string in a set of contiguous unique sub strings of length N. Below an example with $N = 3$:

$$3Shingles("Hello") = \{"Hel", "ell", "llo"\} \tag{2}$$

Combining the two techniques allows an easy computation of similarities between strings. Using this methods is possible to define the Jaccard similarity procedure described by Algorithm 1.

Algorithm 1: jacc(x,y)

```
$s1 \leftarrow 3Shingles(x)$;
$s2 \leftarrow 3Shingles(y)$;
**return** *JaccardIndex(s1,s2)*
```

2.2 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is a numerical statistic that shows the relevance of words in a document [6], hence it can be used to find the keywords of a document, that are the words that better characterize the document. This statistic is the product of two distinct statistics the Term Frequency (TF) and the Inverse Document Frequency (IDF), hence the name TF-IDF. The TF-IDF of each word is computed as follows:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) \tag{3}$$

Where t denotes the terms; d denotes each document; D denotes the collection of documents and TF and IDF are computed as follows:

$$tf(t, d) = \frac{|\{x \in d : x = t\}|}{|t|} \tag{4}$$

$$idf(t, D) = log(\frac{|D|}{1 + |d \in D : t \in d|}) \tag{5}$$

The first part of the formula, $tf(t, d)$ is simply to calculate the number of times each word appeared in each document and dividing it by the total number of terms in that document. The $idf(t, D)$ part assigns a lower weight to frequent words and assigns a greater weight for the words that are infrequent.

This technique can be used to preprocess the market basket and came up with a more suitable dataset. Basically, each purchase may be considered as a document and using TF-IDF is possible to find the most important terms for each document. That way, the sparse table mentioned in Section 1 can be reduced to a fixed number of terms decided a priory. It can be used to preprocess also the cuisines recipe dataset and obtain its key ingredients.

2.3 K-Means

This algorithm was first defined by J. MacQueen (1967) in [4] and represents one of the simplest and efficient unsupervised learning solutions to the well-known clustering problem. Given an initial parameter K and a set of points in R^d it tries to create K clusters minimizing some distance metric between points of each cluster. It does so by picking K initial points as far as possible from each other, which act as cluster representatives and are called centroids. Then all the points are inspected and assigned to the closest cluster. The centroids are recomputed as barycenters of the clusters and the process is repeated until the centroids do not change anymore.

K-Means can be used as a clustering algorithm for the problem mentioned in section 1. Moreover, K-Means is very efficient in terms of time and memory consumption for relatively high dimensional data. However, it also has some disadvantages such as the need of a priory knowledge of K. This method will also be used as baseline to compare the method that will be proposed in section 4. More about the evaluation method and interpretation in section 5.

2.4 Density-Based Spatial Clustering of Applications with Noise

Density-based spatial clustering of applications with noise (DBSCAN) is an algorithm that rises from the need to cluster spatial databases which allows the discovery of clusters of arbitrary shape [2], while the classical solutions to clustering problem do not. Moreover, it is efficient on large databases and can be solved in O(nlogn) computational time and O(n) memory usage. DBSCAN is able to recognize and identify clusters which have a typical density of points which is considerably higher than outside the clusters (noise). The key idea is that for each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points, i.e. the density in the neighborhood has to exceed some threshold [2]. The shape of a neighborhood is determined by the choice of a distance function for two points.

In the case of Market Baskets Clustering, this study use case, DBSCAN can be used if exists some kind of density concept within the data, that is clusters are enough dense while noise is sparse. If the market basket dataset is taken and clustered as it is, using DBSCAN, the result may not be so clear, because the items are strings and the most frequent items may be very different/distant for two baskets of the same cluster. Therefore in order to apply this technique to the dataset, some other preprocessing is needed to ensure that clusters are dense points while noise is not.

3 PROBLEM STATEMENT

This section will formalize the problem introduced in Section 1. Given a set of market baskets that contain food items and a set of recipes characterized by their ingredients and cuisine typology, the goal is to identify types of customers based on the food they have bought and most probably eaten. It is assumed that when a customer purchases some food items, with high probability he will use these ingredients to prepare some recipe. The market basket is defined by a list of baskets $M = [b_1, b_2, ..., b_n]$ and can be represented as a $n \times m$ matrix, where n is the number of baskets and m the number of items. Let call such matrix M, therefore:

$$M[b][i] = \begin{cases} 1, & \text{if } i \in b \\ 0, & \text{otherwise} \end{cases}$$

Where b is a basket, hence a customer purchase and i is an item. The cuisine recipes dataset has a similar format like the market basket dataset, only that it contains a dedicated column indicating the type of cuisine to which the recipe belongs. This second dataset is called R and can be defined as a list of clusters $R = [C_1, C_2, ..., C_l]$ where C_c represents the set of cuisine recipes belonging to cuisine type c (e.g. *Italian, Brazilan, Brittish*) and l is the total number of cuisines. Therefore a cuisine recipe can be defined as a matrix:

$$C_c[r][i] = \begin{cases} 1, & \text{if } i \in r \\ 0, & \text{otherwise} \end{cases}$$

Where r represents a recipe and i an ingredient. For memory optimization the cuisine recipes cluster can also be defined as a $C_c = \{r_1, r_2, ..., r_p\}$, where p is the total number of recipes belonging to cluster C_c and a recipe r_i is a set of ingredients/items. Given this input, the goal is to implement a procedure able to cluster all baskets in such a way that baskets of the same cluster belong to the same customer typology. In order to do this, some rough definition of customer type is needed. In this study, with customer type is intended a category of customers to which belong a set of baskets that most likely contain items that with high probability belong to a subset of the cuisines. This last concept will be better explained in the next section.

4 SOLUTION

In this section, a new method that combines text mining and clustering techniques is proposed as a solution to the defined problem. Initially, the method is presented and then some applied optimizations are described.

4.1 Proposed Method

Considering that the cuisines recipes R is represented by a set of strings, all $C_c \in R$ are transformed to documents by simply concatenating all recipes in an unique string. Let call the just created set of documents $D = \{D_1, D_2, ..., D_l\}$. After that, TF-IDF of each $t \in D_i$ is computed and all terms with $tfidf(t, D_i) > 0.1$ are selected in order to find the terms which best characterize the type of cuisine. Once this has been done, each cuisine C_c is represented by a list of tuples $K_{C_c} = [(i_1, s_1), (i_2, s_2)..., (i_h, s_h)]$ where i_i is an ingredient (also keyword of document D_c) and s_i is its score, which indicates the importance of the ingredient in cuisine C_c.

The next step is to estimate, for each basket the membership level in each cuisine. To do this, all baskets are iterated and for each item, the Jaccard similarity between the item and all cuisines top terms are computed. If the $jacc(i, t) > 0.2$ (i is a basket item and t is a cuisine top term) then the item is considered similar enough to contribute to the score of this basket belonging to the considered cuisine. Such score is calculated for each basket and each cuisine and stored in the $n \times m$ matrix N using the following formula:

$$N[b][d] = \sum_{i \in b} \sum_{(r,s) \in K[d]:jacc(i,r)>0.2} jacc(i, r) \times s \qquad (6)$$

Where b is a basket and $d \in [0, l]$ is one of the considered cuisines. This scores matrix N, unlike the original dataset M is more compact and once normalized represents in a good way the relationship between basket items and cuisine ingredients. In addition to reducing the space, applying this transformation leads two baskets that have common ingredients within the same cuisine to be closer in the euclidean space. The next step is to cluster just the obtained basket scores matrix N in order to find similar baskets. The chosen clustering algorithm is DBSCAN because it was observed that the points belonging to N are either very close each other or enough distant. Moreover, we want to allow the formation of clusters with arbitrary shapes. DBSCAN is applied to obtain clusters representing customer types as stated in section 3. The resulting clusters contain a set of baskets. An output cluster can be formalized as $OutputCluster = \{b_1, b_2, ..., b_g\}$ hence, using N is possible to compute the cluster average score for each cuisine obtaining a vector R^d where d is the number of cuisines. In that way is possible to control the average level of membership in each cuisine for each of the obtained clusters. Moreover, this allows to compare our method to some baseline method. The next section will be described how the method was evaluated and how it differs from a simple applicability of K-Means which represents also our baseline method with which we compare the proposed method. The method described in this section is can also be compacted with the following pseudocode, which returns in output a vector $r \in R^n$ such that $r[i]$ contains cluster of basket i:

Algorithm 2: Proposed Method

```
$M \leftarrow [b_1, b_2, ..., b_n]$# market basket dataset;
$R \leftarrow [c_1, c_2, ..., c_l]$# cuisines recipes dataset;
$K \leftarrow []$# cuisines keywords;
$D \leftarrow []$# cuisines as documents;
# transform cuisines recipes to documents;
foreach $c \in R$ do $D.append(c.toDocument())$ ;
# compute cuisines top terms ;
for $i \leftarrow 0$ to $R.size() - 1$ do
    $K_d \leftarrow []$;
    $d \leftarrow D[i]$;
    $c \leftarrow R[i]$;
    foreach $t \in Set(c)$ do
        $s = tfidf(t, d, D)$;
        if $s > 0.1$ then $K_d.append((t, s))$ ;
    end
    $K.append(K_d)$;
end
# compute baskets membership level;
$N \leftarrow new\ int[n][l]$;
for $b \leftarrow 0$ to $M.size() - 1$ do
    for $c \leftarrow 0$ to $R.size() - 1$ do
        $N[b][c] \leftarrow$
        $\sum_{i \in b} \sum_{(r,s) \in K[c]:jacc(i,r)>0.2} jacc(i, r) \times s$;
    end
end
# cluster baskets;
$\varepsilon \leftarrow 0.001$;
$minPoints = 100$;
$dbscan \leftarrow new\ DBSCAN(\varepsilon, minPoints)$;
$clusters \leftarrow dbscan.fit(N)$;
return $clusters.labels$
```

4.2 Optimizations

In Algorithm 2, it can be observed that the similarity between two items is computed each time is needed. Basically, the procedure ignores that the total number of distinct items is much smaller than the total number of items and therefore most of the time such similarity has already been computed. In order to avoid to compute twice the similarity for the same pair of objects a hash table which stores such similarities can be used. Such hash table should be able to hash 2 keys to the same bucket. In that way, once $jacc(x, y)$ has been computed, it is stored in in the bucket with key (x, y) and if $jacc(y, x)$ is needed its value can be retrieved from $hash(y, x)$ because it points to the already computed $hash(x, y)$. Another operation that is intensively executed ignoring the fact that it may have already been done is the computation of 3-Shingles within the $jacc(x, y)$ procedure. Another hash table can be used to store 3-Grams of all items and then use them each time they are needed. In order to do this, the precomputation described by Algorithm 3 is needed.

Algorithm 3: Precomputation of Similarities and 3-Grams

```
$HNGrams \leftarrow \{\}$ # hash table for 3-Grams;
$HSims \leftarrow \{\}$ # hash table for similarities;
foreach $b \in M$ do
    foreach $i \in Set(b)$ do
        if $HNGrams[i] == Null$ then
            $HNGrams[i] \leftarrow 3Shingles(i)$;
    end
end
foreach $k \in K$ do
    foreach $(i, s) \in k$ do
        if $HNGrams[i] == Null$ then
            $HNGrams[i] \leftarrow 3Shingles(i)$;
    end
end
foreach $i_1 \in HNGrams.keysSet()$ do
    foreach $i_2 \in HNGrams.keysSet()$ do
        if $HSims[(i_1, i_2)] == Null$ then
            $HSims[(i_1, i_2)] \leftarrow$
              $JaccardIndex(HNGrams[i_1], HNGrams[i_2])$;
        end
    end
end
```

Notice that in the pseudocode, the method call $HSims[(i_1, i_2)]$ is equivalent to $HSims[(i_2, i_1)]$ because of the specific hash table implementation. Once this precomputation is done formula 6 can be rewritten in the following way:

$$N[b][d] = \sum_{i \in b} \sum_{(r,s) \in K[d]:HSims[(i,r)]>0.2} HSims[(i, r)] \times s \quad (7)$$

This allows a significant reduction in computation time while increases memory consumption. However the number of unique items is significantly smaller than the total number of items, hence memory consumption should not increase too much.

5 EXPERIMENTAL EVALUATION

The main goal of this paper is to identify groups of customers based on the products that they purchase. This chapter will explain the data used for this scope and also the analysis done in order to evaluate the implementation described in 4. Moreover, scalability and baseline evaluations will be presented.

5.1 Evaluation with real data

To verify that the methodology presented extracts useful information, executions over real datasets were performed. The first dataset utilized is the Groceries Market Basket Dataset [5]. The data contained inside it is made of 9835 transactions done by customers shopping for groceries and there are 169 unique items. The average number of elements per baskets is 4.41. The second dataset used is the Recipe Ingredients Dataset [3]. This publicly available resource contains 39.774 recipes, with the relative list of ingredients, divided into 20 possible types of cuisine. The number of ingredients per recipe has variable length. The two datasets originate from different sources and there are no guarantees about matches between elements inside the two sources of data.

As described in Section 4 the implementation computes for each basket its cuisines' concept strength as a score. Once this preliminary phase is completed a clustering over the obtained scores is done. This step is done using the DBSCAN algorithm available in the Python library scikit-learn [7]. The implementation has memory complexity $O(n * d)$, where d is the average number of neighbors, while the original DBSCAN had memory complexity $O(n)$. This allows a faster computation subject to a greater usage of memory. The DBSCAN algorithm necessitates two parameters: a radius and a minimum number of points. In the following analysis is chosen 0.001 for the former while 100 for the latter. These coefficients allow the algorithm to cluster together only points that are definitely close each other and only in the case that there is a significant number of elements within the specified radius.

The goal of the clustering is to identify costumers that with high probability belong to a subset of the possible cuisines. Basket scores which represent the cuisines' concept strength are normalized around zero and used as input for the clustering procedure. Further for each obtained cluster, the average score for each cuisine is computed. The result can be analyzed in table 1. Analyzing the results it can be noticed that:

- Cluster -1 contains 5538 baskets. The scores for every cuisine in this cluster are similar and they are all positive. This can be interpreted as that there are many baskets in which a cuisine does not prevail over the others and they have a little of every concept.
- Cluster 0 contains 2544 baskets. The scores in this cluster are similar and they are all negative. This indicates that many baskets can no be clearly assigned to a specific cuisine because all the values are significantly under the mean.
- Cluster 1 contains 212 baskets. The score for the 'greek' cuisine here is the only one positive (+0.7) and with a large difference to the second top cuisine (-0.3). This suggests that this cluster contains items that with high probability can be associated to recipes of type 'greek'.
- Cluster 2 contains 505 baskets. In this cluster there only one term with score above the mean. Although this value is not so high and has a small difference to the second top cuisine, it is the only one positive. This is an indication that there

Table 1: The DBSCAN-algorithm results

	C -1	C 0	C 1	C 2	C 3	C 4	C 5	C 6
brazilian	0.357	-0.567	-0.567	0.171	-0.567	-0.173	-0.567	-0.567
british	0.401	-0.568	-0.568	-0.229	-0.568	-0.339	-0.568	-0.568
cajun_creole	0.354	-0.497	-0.497	-0.497	-0.497	-0.497	0.603	-0.497
chinese	0.302	-0.41	-0.41	-0.41	-0.41	-0.013	-0.41	-0.41
filipino	0.381	-0.55	-0.55	-0.281	-0.55	-0.015	-0.55	-0.55
french	0.414	-0.549	-0.549	-0.549	-0.549	-0.239	-0.549	-0.549
greek	0.346	-0.507	0.733	-0.507	-0.507	-0.507	-0.507	-0.507
indian	0.301	-0.412	-0.412	-0.412	-0.412	0.054	-0.412	-0.412
irish	0.399	-0.65	-0.65	-0.363	0.139	-0.446	-0.65	-0.65
italian	0.334	-0.43	-0.43	-0.43	-0.43	-0.43	-0.43	-0.43
jamaican	0.362	-0.532	-0.532	-0.2	-0.532	-0.01	-0.532	-0.532
japanese	0.279	-0.387	-0.387	-0.387	-0.387	0.139	-0.387	-0.387
korean	0.272	-0.368	-0.368	-0.368	-0.368	-0.016	-0.368	-0.368
mexican	0.362	-0.466	-0.466	-0.466	-0.466	-0.466	-0.466	-0.466
moroccan	0.263	-0.339	-0.339	-0.339	-0.339	-0.339	-0.339	-0.339
russian	0.378	-0.532	-0.532	-0.319	-0.532	-0.145	-0.532	-0.532
south._us	0.404	-0.56	-0.56	-0.315	-0.56	-0.346	-0.56	-0.56
spanish	0.275	-0.355	-0.355	-0.355	-0.355	-0.355	-0.355	-0.355
thai	0.383	-0.563	-0.563	-0.162	-0.563	-0.563	-0.563	0.015
vietnamese	0.312	-0.423	-0.423	-0.423	-0.423	-0.002	-0.423	-0.423

Table 2: The baseline results

	C 0	C 1	C 2	C 3	C 4	C 5	C 6	C 7
brazilian	-0.169	0.14	0.309	-0.008	0.41	-0.499	-0.339	-0.217
british	-0.13	0.195	0.343	-0.065	0.192	-0.509	-0.392	-0.153
cajun_creole	-0.069	0.121	0.369	0.08	-0.153	-0.409	-0.365	-0.208
chinese	-0.037	0.104	0.277	0.01	-0.169	-0.356	-0.223	-0.183
filipino	-0.108	0.112	0.377	-0.003	0.026	-0.465	-0.309	-0.215
french	-0.066	0.202	0.315	0.003	-0.116	-0.481	-0.353	-0.139
greek	-0.033	0.221	0.276	0.029	-0.188	-0.445	-0.376	-0.155
indian	-0.034	0.082	0.285	0.008	-0.176	-0.352	-0.216	-0.167
irish	-0.167	0.206	0.273	0.006	0.093	0.192	-0.412	-0.114
italian	-0.018	0.137	0.261	0.056	-0.2	-0.397	-0.311	-0.138
jamaican	-0.109	0.125	0.336	0.006	0.084	-0.461	-0.306	-0.206
japanese	-0.03	0.12	0.212	0.006	-0.15	-0.324	-0.178	-0.15
korean	-0.032	0.104	0.236	0.011	-0.153	-0.321	-0.199	-0.154
mexican	-0.031	0.116	0.349	0.045	-0.214	-0.436	-0.363	-0.177
moroccan	-0.018	0.044	0.285	0.024	-0.177	-0.321	-0.238	-0.175
russian	-0.101	0.197	0.296	-0.036	0.069	-0.463	-0.338	-0.125
south._us	-0.118	0.213	0.324	-0.045	0.132	-0.501	-0.388	-0.142
spanish	-0.004	0.055	0.25	0.046	-0.186	-0.334	-0.233	-0.179
thai	-0.113	0.132	0.353	0.021	0.168	-0.472	-0.402	-0.268
vietnamese	-0.041	0.122	0.262	0.019	-0.151	-0.367	-0.228	-0.18

are users that buy ingredients that likely can be traced back to 'brazilian' recipes.

- Cluster 3 contains 499 baskets. The score for the 'irish' cuisine here is the only one positive and with a quite large difference to the second top cuisine. This is evidence that there are users that buy ingredients that most likely belong to 'irish' recipes.
- Cluster 4 contains 217 baskets. In this cluster, there are only two cuisines above the mean but only 'japanese' has a considerable high score. It follows from here that there are users that buy ingredients that likely can be traced back to 'japanese' recipes. However, in this cluster also 'indian', 'vietnamese', 'jamaican', 'chinese', 'filipino' and 'korean' cuisines have scores around zero. The interpretation that may be done is that these cuisines have many ingredients in common. In fact all of them, apart from 'jamaican', may be considered belonging to Asian cuisines, hence they could have many features in common.
- Cluster 5 contains 157 baskets. The average score for the 'cajun_creole' cuisine here is the only one positive and with a very large difference to the second top cuisine. In consequence, can be stated that there are customers that buy ingredients that most likely belong to 'cajun_creole' recipes.
- Cluster 6 contains 163 baskets. The score for the 'thai' cuisine here is the only one positive and with a quite large difference to the second top cuisine. Thereby some users buy ingredients that most likely correspond to 'thai' recipes.

As can be observed many elements are inside the 'Cluster -1' or the 'Cluster 0', but this is not surprising. The average dimension size of the baskets is 4.41, thus it is quite small. Moreover, customer during a purchase may buy only one element for a specific recipe since they already have the others or spread among different purchases the ingredients necessary for it. These are the motivation behind the fact that many baskets can not be clearly attributed to any cuisine or they have a little of each concept. Finally, can be also noticed that there are clusters that clearly match customer to specific cuisine.

5.2 Baseline Evaluation

To prove that the presented methodology is effective and extracts useful information a comparison to a baseline method is done. This evaluation will try to show the significance of the cuisines recipes dataset. Instead of clustering basket scores as the proposed method, the input for the clustering procedure is represented by market baskets dataset converted from they text document representation to a matrix of token counts. The chosen clustering algorithm for this scope is the K-Means, one of the most famous and fastest clustering algorithm. It was not chosen DBSCAN as in the proposed method because it was observed that points of this different input are preatty sparse, hence DBSCAN would require big values for its parameters ε and $minPoints$ resulting however most of the time in 2 big clusters with little space for interpretation of mining results. The implementation used for K-means is the one available in the Python library scikit-learn [7]. K-Means requires one parameter: the number of clusters that are desired in output. In order to create a solution comparable with the previous methodology, this parameter was set to the same number of clusters produced by the DBSCAN algorithm, namely 8. The evaluation was also performed with other reasonable and different values for K, resulting in similar results. For this reason it was decided to present in the paper the result for the basline method with $K = 8$. In order to better interpret the obtained clusters using K-Means and compare them with the proposed method, the baskets scores matrix N is used again to compute the average score of each cluster for each cuisine. This is done because some semantic interpretation of the data has to be done and because baskets scores table represent in a good way the relationship between each basket and a specific cuisine. Moreover this can give insights about which type of customers the baseline method may have found.Analyzing the obtained results of the baseline method in Table 2, it can be observed that:

- Cluster 0, Cluster 6 and Cluster 7, that fits respectively 3217, 954 and 336 items, they all contain values that are under the mean and quite similar among them. This can be interpreted as that there are many baskets, in different clusters, which can no be clearly assigned to a specific cuisine.

- Cluster 1 and Cluster 2, that fits respectively 1379 and 1915 baskets, they all contain values that are above the mean and quite similar among them. This indicates that as that there are many baskets, in different clusters, in which a cuisine does not prevail over the others and they have a little of every concept.
- Cluster 3 contains 768 baskets. In this cluster, there are both positive and negative values but all of them are ~ 0. Thereby, since all the values are close to their mean, these baskets can not be assigned to a specific cuisine and they have a little of every concept.
- Cluster 4 contains 873 baskets. In this cluster, there are both values under and above the mean. Among the latter, there is one value that is slightly high than the others. This can be an indication that there are users that buy ingredients that likely belong to 'brazilian' recipes.
- Cluster 5 contains 393 baskets. The value for the 'irish' cuisine here is the only one positive and with a quite large difference to the second top cuisine. In consequence, can be stated that there are customers that buy ingredients that most likely belong to 'irish' recipes.

As can be observed this methodology is not very effective; six out of eight clusters do not provide useful information since there is not a cuisine that prevails over the others. An interpretation may be that clustering the dataset selecting as features the count of the words for each basket do not reflect well the relationship between a basket and a cuisine. Moreover, most of the time a basket contains only one ingredient per type, resulting in considering a basket which contains only one common ingredient from a cuisine as important as another basket which contains a very characteristic ingredient. This may lead in clustering elements together disregarding the importance of each ingredient and therefore baskets seem to be more similar each other than they are. Given that the proposed method clearly outputs clusters with high scores for specific subsets of cuisines, it can be concluded that for this specific use case, selecting features is a critical process and that the proposed features act quite well and give a large space for interpretation of mining results.

5.3 Scalability Evaluation with Synthetic Data

In order to test the scalability of the proposed method, some analysis using synthetic data is done. The main goal of this last analysis is to stress the procedure with large amounts of data in order to understand which are memory consumption and time complexity worst cases. This evaluation will consider the two main intensive operations of the proposed procedure, which are the computation of baskets membership levels and clustering. For the former, also the optimizations will be considered because they are part of the final implemented and tested solution. The simulation is done generating a bigger and synthetic Groceries Market Basket datasets; the aim is to test that the proposed methodology will work also for Groceries Markets that have larger number of purchases. Thus, since the Recipe Ingredients Dataset is involved only for the computation of the Basket scores, and that it is assumed that the number of recipes will not increase as the number of possible baskets, the cuisines recipes dataset is kept as it is.

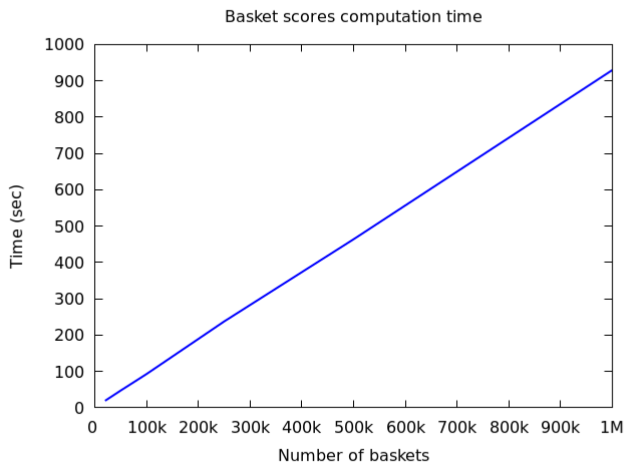

Figure 1: Time complexity for the computation of baskets membership levels

To generate a large amount of data that follows a realistic model, some statistics on the available Groceries Market Basket Dataset [5] are computed. The first step is the calculation of the average and the standard deviation of the number of items inside the baskets. Then, given the amount of baskets that are desired in output, the number of the items that will be inside each of them follows the pdf characterized by the previously computed statistics. The possible items inside a basket are the same that are present in the Groceries Market Basket Dataset. This procedure generated six new datasets of different sizes: 20k, 50k, 100k, 250k, 500k and 1M baskets.

Given a synthetic dataset one of the monitored procedures is the computation of the basket scores. This step is done as described in Section 4 utilizing the optimization presented. Then, the second tracked computation is the clustering. As previously stated, the implementation of the DBSCAN algorithm available in the Python library scikit-learn has memory complexity $O(n * d)$, while the original DBSCAN had memory complexity $O(n)$. This variation in the memory complexity represents an obstacle for the mining of 'big' synthetic datasets with high density. For this reason, in order to correctly manage such data and maintaining this clustering algorithm, the ELKI Data Mining Framework [1] DBSCAN is used for the evaluation. This Java implementation requires memory complexity of $O(n)$ and uses an R*-tree index to accelerate the computation.

The time complexity obtained for the two monitored procedures can be inspected in Figure 1 and Figure 2. As can be seen, the time complexity resulting for the baskets cuisines' concept strength computation is $O(n)$, thus is linear in function of the number of elements. For the clustering procedure it is slightly worse, and can be estimated in $O(n * logn)$.

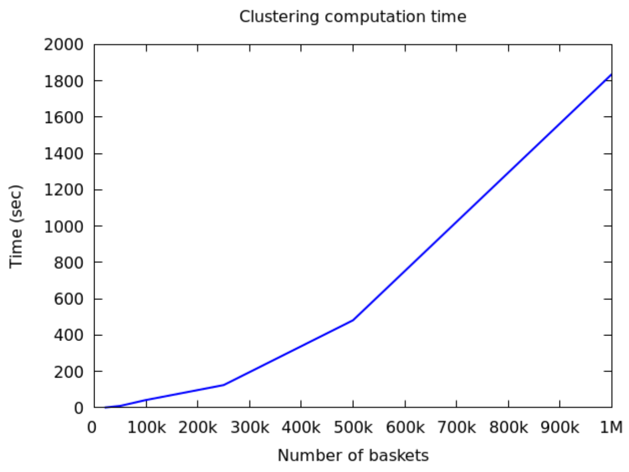

Figure 2: Time complexity for clustering

6 CONCLUSION

In this paper, an effective method able to face the problem of identifying types of customers based on the food they have bought is presented and evaluated. This work represents a suitable tool able to extract useful information that companies may use for insights about their business strategies. Given a dataset of market baskets and a dataset of recipes, using text mining and clustering techniques, the proposed method is able to detect useful relations between the two datasets in order to reach the final goal. This is done by performing several steps. Firstly, TF-IDF is used to extract the best terms that characterize the available cuisines. Then, all the basket items are compared to such terms in order to assign to each basket a score of its belonging to each of the possible cuisine. Finally, over these scores, the DBACAN algorithm is applied to cluster the baskets.

The proposed work is evaluated against a baseline solution and against its ability to scale up to large amounts of data. The good results obtained in both the two tests indicate that the presented methodology exceeds a trivial and naive usage of clustering technique and that it can handle data coming from markets that have a large number of purchases.

Future research should consider the potential effects of greater Recipe Ingredients Dataset, such also the ability to distribute among different commodity hardware the overall computation.

REFERENCES

[1] elki project. 2020. ELKI Data Mining Framework. Retrieved February 6, 2020 from https://elki-project.github.io/
[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.
[3] kaggle.com. 2019. Recipe Ingredients Dataset. Retrieved February 6, 2020 from https://www.kaggle.com/kaggle/recipe-ingredients-dataset
[4] James B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations.
[5] Irfan Nasrullah. 2019. Groceries Market Basket Dataset. Retrieved February 6, 2020 from https://www.kaggle.com/irfanasrullah/groceries
[6] Shahzad Qaiser and Ramsha Ali. 2018. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International Journal of Computer Applications* 181 (07 2018). https://doi.org/10.5120/ijca2018917395
[7] scikit learn.org. 2020. scikit-learn Machine Learning in Python. Retrieved February 6, 2020 from https://scikit-learn.org/stable/index.html
[8] wikipedia.org. 2019. Jaccard Index. Retrieved February 6, 2020 from https://en.wikipedia.org/wiki/Jaccard_index