Mühendislik Bilimleri ve Tasarım Dergisi
7(3), 608 – 618, 2019
e-ISSN: 1308-6693

Journal of Engineering Sciences and Design
DOI: 10.21923/jesd.467036

Araştırma Makalesi Research Article

# COMPARISON OF THE DATA MATCHING PERFORMANCES OF STRING SIMILARITY ALGORITHMS IN BIG DATA

**Bekir AKSOY[1], Sinan UĞUZ[2], Okan ORAL[3*]**

[1] Applied Sciences University of Isparta, Technology Faculty, Mechatronics Engineering, Isparta, Turkey
[2] Applied Sciences University of Isparta, Technology Faculty, Computer Engineering, Isparta, Turkey
[3]Akdeniz University, Engineering Faculty, Mechatronics Engineering, Antalya, Turkey

**Keywords**
*Algorithms, Text analysis, Natural language processing, Data analysis, Databases.*

**Abstract**
The great mobility in the world tourism in recent years has also enabled this sector to be included among the study areas of big data. In this study, a solution proposal was put forward by using the big data and string similarity algorithms (SSA) for the problems arising from the entry of the hotel data coming from different providers into databases with different names and addresses. Therefore, 2599 hotels of a tourism agency with a wide hotel network located in London were selected as the sample, and the Map-Reduce process was performed by using the Soundex algorithm to match these hotels with approximately three million hotel data coming from seventy different providers. Matching with Map-Reduce ensured a significant reduction in process count and process time. Furthermore, the Dice coefficient, Levenshtein and Longest common subsequence (LCS) algorithms were compared in terms of the data that they correctly matched, and process time. In this stage, the words decreasing the score of the algorithms in the database were detected and removed before the algorithms were implemented. The Dice coefficient algorithm yielded better results in terms of correct matching, and the Levenshtein algorithm yielded better results in terms of process time.

# BÜYÜK VERİDE METİN BENZERLİK ALGORİTMALARININ VERİ EŞLEME PERFORMANSLARININ KARŞILAŞTIRILMASI

**Anahtar Kelimeler**
*Algoritmalar, Metin analizi, Doğal dil işleme, Veri analizi, Veri tabanları.*

**Öz**
Son yıllarda dünya turizmindeki büyük hareketlilik, bu sektörün büyük verinin çalışma alanları arasına girmesini sağlamıştır. Bu çalışmada farklı sağlayıcılardan gelen otel bilgilerinin, veritabanlarına farklı isim ve adreslerle girilmesi sonucu oluşan problemler için, büyük veri ve string similarity algoritmaları (SSA) kullanarak bir çözüm önerisi ortaya konulmuştur. Bunun için geniş bir otel ağına sahip bir turizm acentasının Londra'da bulunan 2599 oteli örneklem olarak seçilmiş ve bu oteller ile yetmiş farklı sağlayıcıdan gelen yaklaşık üç milyon otel bilgisinin eşleştirilmesi için, soundex algoritmasından faydalanılarak Map-Reduce işlemi gerçekleştirilmiştir. Map-Reduce ile eşleme işlem sayısı ve işlem süresinde önemli ölçüde azalma sağlanmıştır. Çalışmanın diğer aşamasında ise Dice coefficient, Levenshtein ve Longest common subsequence (LCS) algoritmaları, doğru eşleyebildikleri veri ve işlem süresi açısından kıyaslanmıştır. Bu aşamada algoritmalar uygulanmadan önce veri tabanında algoritmaların skorunu düşüren kelimeler tespit edilerek çıkartılmıştır. Doğru eşleme bakımından Dice coefficient algoritması, işlem süresi açısından ise Levenshtein algoritması daha iyi sonuçlar üretmiştir.

**Alıntı / Cite**
Aksoy, B., Uğuz, S., Oral, O., (2019). Comparison of the Data Matching Performances of String Similarity Algorithms in Big Data, Journal of Engineering Sciences and Design, 7(3), 608-618.

| Yazar Kimliği / Author ID (ORCID Number) |
|---|
| B. Aksoy, 0000-0001-8052-9411 |
| S. Uğuz, 0000-0003-4397-6196 |
| O. Oral, 0000-0003-4256-0930 |

| Makale Süreci / Article Process | |
|---|---|
| **Başvuru Tarihi / Submission Date** | 03.10.2018 |
| **Kabul Tarihi / Accepted Date** | 04.04.2019 |
| **Yayım Tarihi / Published Date** | 15.09.2019 |

* İlgili yazar / Corresponding author: okan@akdeniz.edu.tr, +90-242-310-6377

## 1. Introduction

While the total contribution of the tourism sector to the world economy between the years 2006 and 2016 was 7.61 trillion dollars, the number of international tourists traveling around the world only in 2015 was 1 billion 186 million (Smith, 2016). One of the parameters that make a significant contribution to the great economic mobility in the sector is information and communication technologies that have developed in recent years. For a tourist, information and communication technologies are used intensively in every stage of mobility, before travelling, on the road, at the destination and on the way back. Nowadays, only corporate web pages or advertisements in different communication fields are not enough for hotels and tourism agencies to provide their services to tourists in the best way. Social media, forums, web blogs and all web environments in which accommodation comments can be made appear as important decision-making factors in tourists' preferences. Therefore, all actors of the tourism sector who are commented in the social media should follow these platforms in the virtual environment.

The number of data generated in the virtual environment for the tourism industry, which is a large sector, is also large. Therefore, the actors of this sector can make this fast data flow meaningful for them with big data analyses. In the world, tourism industry and big data-based academic studies have focused on different objectives. The studies on the estimation of tourist demands for later periods (Li vd., 2017; Toole vd., 2015) are considered to be important for the future planning of the industry. Studies (Liu vd., 2017;Gupta and Upadhyay, 2015; Xiang vd., 2015) have been carried out to determine the hotel satisfaction levels from the comments produced from social media sites such as Twitter, Facebook, Flickr, etc., forums and popular hotel search sites such as tripadvisor.com and booking.com. Some studies (Önder, 2017; Miah vd., 2016; Chen and Zhou, 2015; Peng and Huang, 2012) are related to the creation of tourist travel maps with semantic web applications by analyzing the tourist behaviors through the systems in which tourist behaviors can be tracked by location such as GPS, city travel cards and credit cards. The development of systems suggesting hotel proposals with big data analysis is also among the studies carried out (Shrote and Deorankar, 2016).

It was envisaged that the amount of data generated in the world would be doubled every two years and would reach approximately 8 Zettabytes by 2015 (Sagiroglu and Sinanc, 2013). Some auxiliary science fields such as natural language processing (NLP) are needed in big data analyses to analyze meaningful data from such a big data universe. NLP is a field of science that refers to the process of building, analyzing and interpreting the model of human-specific languages that is appropriate to be processed by the computer (Bird vd., 2009). NLP is used in numerous areas. NLP mechanisms are utilized in many areas such as the detection of incorrectly written words, alternative word suggestions, analysis of queries written in a form similar to the natural language, sentence translations from language to language, sentence or text derivation in the natural language, detection of junk mails (spam) by email providers, computer-aided language teaching, and detection of text plagiarism. One of the main problems of NLP is the correction of incorrectly written word phrases. Therefore, it is aimed to correct incorrectly written word phrases with the highest accuracy percentage using the advanced Soundex, Dice coefficient, Levenshtein and LCS algorithms.

The comments produced on the internet should be addressed in terms of the NLP while performing the big data analysis in the tourism sector because it is necessary to take into account that the comments have an official language or a nonofficial language (which may contain abbreviations and slang) as well as the unique grammatical structure of the language used in the comments made. One of the scenarios that can be encountered for tourism-related NLP can be experienced in the service sector, which is the other stakeholder of the tourism industry. Nowadays, tourism agencies play an important role between the hotel and the customer in terms of marketing for holiday sales. Each tourism agency has many hotels marketed by it within itself. The data related to these hotels are constantly updated by being stored in big databases. However, the incorrect entry of the hotel data by users into the system causes multiple different records of the same hotel to be kept in the database. A chain of hotels may have hotels in different cities, or the hotel name may contain words such as resort, SPA and luxury. For example, the fact that a hotel is recorded as “Miracle Hotel” or “Miracle Hotel & SPA” into the database poses a problem.

There are two main purposes of this study. Firstly, to increase the quality of the representation of hotel names in databases by correcting these types of incorrectly written word phrases with the highest accuracy percentage through the SSA by first detecting non-standard abbreviations and nonsense characters in names. Secondly, to determine the effect for the process time on the sample dataset of map reduce.
For this purpose, the Map-Reduce process was performed by using the Soundex algorithm to match the data of the hotels of a tourism agency that were selected as the sample with the data of the hotels coming from different providers. In the other stage of the study, the Dice coefficient, Levenshtein and LCS algorithms were compared in terms of the data that they correctly matched, and process time. In the literature studies carried out, it has been observed that the studies on this subject, especially those carried out using data sets belonging to the tourism sector, are limited. The fact that higher performance was obtained from the SSA by removing the common

words in the dataset before the SSA was used is the most important factor that distinguishes this study from other studies.

This article is organised as follows. SSAs are classified, and some algorithms are introduced in Section II. The concept of big data is described in Section III. The method of this paper is introduced in Section IV. The findings of this paper are presented in Section V. The paper ends with a conclusion and a description of future work.

## 2. String Similarity Algorithms

In information technologies, text similarity takes an important place among the methods that are used to analyze text data. The SSA allows for the numerical expression of similarity ratios between texts. Similarity ratios may consist of texts, words or long sentences to be calculated (Dursun and Sonmez, 2008). The SSA, which is the subject of the NLP study field, includes many subjects of scientific study such as word recognition (Fuentes vd., 2016), information retrieval (Kisla vd., 2015), text summarization (Bakar vd., 2000), word learning (Kurdziel and Spencer, 2016), translation (Xiang vd., 2014), text classification (Lodhi vd., 2002), development of question and answer systems (Ilhan vd., 2008), and detection of plagiarism Baruah and Mahanta, 2013).

String similarity measures are presented in Figure 1. Edit-based similarity measures indicate after how many moves the distance of two texts will be equal. These moves are based on the differentiation, alteration, reduction or increasing of the characters found in the same or nearby locations in the texts (Deng vd., 2013; Jiang vd., 2013). The total similarity is calculated by dividing the texts to be found in Token based similarity measures into words or word groups. In hybrid similarity measures, edit-based similarity measures and token-based similarity measures are used together. Different combinations of letters in many language structures on the Earth may have the same pronunciations. For this reason, it is common for users to write words incorrectly while writing since the definition of words may become complicated.

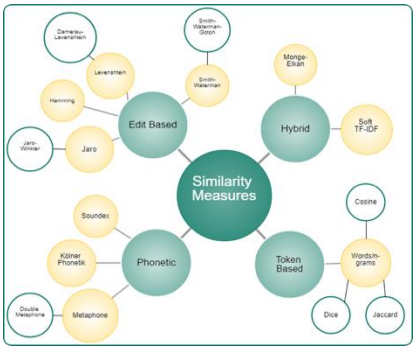



**Figure 1.** String Similarity Measures (Naumann and Herschel, 2013).

Furthermore, homophony between words makes it difficult to classify typographical errors correctly or to index these names correctly. Phonetic algorithms are among the SSAs used to find solutions to these problems (Mutalib and Noah, 2011; Parmar and Kumbharana, 2014).

### 2.1. Soundex Algorithm

Different combinations of letters in a word may show similarity in the pronunciation of that word. This may lead to the misspelling of words. The Soundex algorithm, one of the first algorithms developed to find a solution to this problem, was developed by Robert Russell and Margaret Odell in 1918 (Odell and Russell, 1918). Since the Soundex algorithm is an algorithm developed only for the English language, various studies have been carried out regarding the adaptations of this algorithm in different languages (Yahia vd., 2006; Baruah and Mahanta, 2015; Bhatti vd., 2014; Jaisunder, 2017; Shedeed and Abdel, 2011; Freeman vd., 2006). The Soundex algorithm generates codes depending on pronunciation in detecting the similarity of words. Table 1 shows the processing steps of the Soundex algorithm.

**Table 1.** Soundex Algorithm

| | |
|---|---|
| **1** | All letters in the word are capitalized, and all punctuation marks are removed. |
| **2** | The first letter remains in the word. |
| **3** | The letters 'A', E', 'I', 'O', 'U', 'H', 'W', 'Y' among the letters except the first letter are removed from the word. |
| **4** | Each letter is replaced with the appropriate number that corresponds to it. Except for these, any numeric, alpha-numeric or character existing in the word is removed and replaced with a space. |
| **5** | The words except for the first letter are coded according to Table 2. |
| **6** | Only one of the same adjacent letters remains, and the other one is removed from the word. |
| **7** | The spaces are deleted, and zero is added to the end as many times as the missing number to complete the expression to 4 digits. |

The numeric equivalents of the letters in the algorithm are presented in Table 2.

**Table 2.** Numeric Equivalents of The Letters

| Numeric Equivalent | Letter Equivalent |
|---|---|
| 1 | 'B', 'F', 'P', 'V' |
| 2 | 'C', 'G', 'J', 'K', 'Q', 'S', 'X', 'Z' |
| 3 | 'D','T' |
| 4 | 'L' |
| 5 | 'M','N' |
| 6 | 'R' |

The code equivalents of some words with similar pronunciation generated by the algorithm in the Soundex algorithm are presented in the examples in Table 3.

**Table 3.** Soundex Code Equivalents of Some Words

| Word | Soundex Code |
|---|---|
| SCHMID, SCHMIDT, SCHMIT | S530 |
| REAL, RAIL, REILLY, RULE | R400 |
| JONES, JONAS, JOHANNAS | J520 |
| HOTEL, HOTTEL, HUDDLE | H340 |
| SURFACE, SERVOS | S612 |
| TURKEY, TOWERS, THRUSH | T620 |
| BEEMAN,BEAMAN, BAUMANN | B550 |

### 2.2. Dice Coefficient Algorithm

The Dice coefficient measure is used in determining the similarities and differences in datasets. This method is calculated by dividing two times of the data intersected in two data sets by the sum of individual data elements (Dice, 1945). The statistical expression of the Dice coefficient similarity coefficient of the X and Y word set is presented in Equation (1). It can be interpreted that the closer the similarity coefficient found to 1 is, the more similar the two texts are.

$$D(X,Y) = \frac{2.|X \cap Y|}{|X| \cup |Y|} \quad (1)$$

where X is the first word, Y is the second word.

In Figure 2, it is observed that the Dice coefficient similarity coefficient of the X and Y word set is calculated as an example.

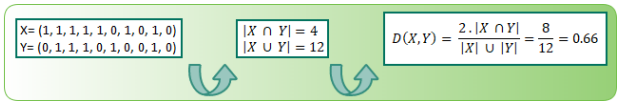



**Figure 2.** Exemplary Calculation of The Dice Coefficient Similarity

### 2.3. Levenshtein Algorithm

The Levenshtein algorithm is used in finding the process count required to calculate the similarity of two words by converting one of the two words given to the other one (Levenshtein, 1966). The purpose of this algorithm is to calculate the amount of change in the letter between two words. This algorithm results in the least cost to convert one text to another by adding, deleting and displacing. The fact that the calculated Levenshtein distance value is zero indicates that the two words compared are the same (Kruskal and Sankoff, 1999; Heeringa, 2004; Ugon vd., 2015; Chaudhary vd., 2016; Kurdziel and Spencer, 2016). In Equation (2), the Levenshtein similarity coefficient is calculated.

$$D(X,Y) = 1 - \frac{Z}{\max(|X|,|Y|)} \quad (2)$$

where X is the first word, Y is the second word, and Z is the Levenshtein distance.

In order to calculate the Levenshtein distance of two words with N and M word lengths, it is necessary to create a matrix in the form of [N+1] x [M+1] (Su vd., 2008). For example, the calculation of the Levenshtein similarity of the words "rixos" and "pinas" can be examined in Figure 3. The matrix values for the words pinas and rixos given in Figure 3.a are N=5 and M=5. During the calculation of the similarity value, if the letters are not equal, 1 is added and written to the smallest value to the left, top and cross left of the cell (Figure 3.b). If the letters compared on the matrix are equal, the value on its cross left is copied (Figure 3.c).

| | | r | i | x | o | s |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| p | 1 | | | | | |
| i | 2 | | | | | |
| n | 3 | | | | | |
| a | 4 | | | | | |
| s | 5 | | | | | |

**(a)** Levenshtein Matrix

| | | r | i | x | o | s |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| p | 1 | 1 | 2 | 3 | 4 | 5 |
| i | 2 | | | | | |
| n | 3 | | | | | |
| a | 4 | | | | | |
| s | 5 | | | | | |

**(b)** Letters are Different

| | | r | i | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| p | 1 | 1 | 2 | 3 | 4 | 5 |
| i | 2 | 2 | 1 | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

**(c)** Letters are The Same

| | | r | i | x | o | s |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| p | 1 | 1 | 2 | 3 | 4 | 5 |
| i | 2 | 2 | 1 | 2 | 3 | 4 |
| n | 3 | 3 | 2 | 2 | 3 | 4 |
| a | 4 | 4 | 3 | 3 | 3 | 4 |
| s | 5 | 5 | 4 | 4 | 4 | 3 |

**(d)** Result Matrix

**Figure 3.** Identification of the levenshtein matrix

The value in the last cell of the matrix found after all letters are equalized gives the Levenshtein distance (Figure 3.d). The lower the distance value obtained is, the less the cost is. When the closeness of these words

is graduated, fewer changes indicate that the two words are more similar to each other.

### 2.4. Longest Common Subsequence Algorithm

The LCS algorithm is a similarity measure that is successfully used for sequence matching (Nyirarugira and Kim, 2015). The LCS algorithm aims to find a common subsequence with the longest possible length in two sequences (Chowdhury vd., 2014; Tabataba and Mousavi, 2012). As it is seen in Equation (3), the algorithm needs to be repeated until all characters are matched.

$$LCS(X_i,Y_j)\begin{cases} 0 & if\ i=0\ or\ j=0 \\ 1+LCS(X_{i-1},Y_{j-1}) & if\ X_i=Y_j \\ \max(LCS(X_{i-1},Y_j),LCS(X_i,Y_{j-1}) & if\ X_i\neq Y_j \end{cases} \quad (3)$$

where X is the first word, and Y is the second word.

An exemplary problem for the LCS algorithm is presented in Figure 4. Here, it is observed that the commonality of the word "abcf" was achieved sequentially by comparing the words "abcdaf" and "acbcf". The first word to be compared is placed in the first line of the, and the second word is placed in the first column. All members of the second line and the second column are set as zero. The first letter of the word in the first column is compared individually with all letters of the word in the first line. If the same letter is matched, one more than the number in the upper cross left is entered into the cell into which the number will be entered. If the letters are not matched, the number larger than those in the first cells on its top and left is written in the cell into which the number will be entered. After the values are entered into all cells, from which cell the value in the cell comes is marked with arrows until the first cell by starting from the last cell backwardly.

| | | **a** | **b** | **c** | **d** | **a** | **f** |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **a** | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| **c** | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| **b** | 0 | 1 | 2 | 2 | 2 | 2 | 2 |
| **c** | 0 | 1 | 2 | 3 | 3 | 3 | 3 |
| **f** | 0 | 1 | 2 | 3 | 3 | 3 | 4 |

**Figure 4.** An Exemplary LCS Algorithm

While the path marked with arrows is followed, the cells with increment in number are circled. When the matches in the circled cells are written, the longest commonality of two elements in a sequential manner is achieved.

## 3. Big Data

In the age of technology, very big data stacks are created with e-mails, videos, sound files, images, click flows, logs, messages, search queries, social network interactions, science data, sensors and mobile phones. It is necessary to develop new methods since it is getting more difficult to capture, format, store, manage, share, analyze and visualize these data with each passing day (Zikopoulos and Eaton, 2011). The concept of big data was defined by Cavoukian and Jonas (2012), as "datasets the size of which is beyond the ability of typical database software tools to capture, store, manage, and analyze". Furthermore, it can also be defined as a data stack that makes traditional data processing methods insufficient. There are five main features (Volume, Velocity, Variety, Verification, Value) that define big data (Figure 5).


**Figure 5.** Big data 5V features

Volume refers to the high volume of data. For example, there are millions of sensors in the engine and other parts of an airplane. These sensors record each state in the airplane and create very large volume data in a single flight. Velocity refers to the rate of data. The data obtained by the sensors in the airplane can be collected quickly as a result of the high levels which today's microprocessor speeds have reached. Variety refers to different types of data that can be obtained such as image, sound and text file. Verification is the feature that is used in cases when it is necessary to check whether the incoming data are safe during data flow. The feature of Value refers to obtaining significant results from the big data analysis that is performed using the first four features.

To perform analyses using the features of big data, it is necessary to split the data into pieces that can be processed and to bring the results back together. This process is called Map-Reduce. The Map-Reduce process consists of four stages as it is seen in Figure 6.

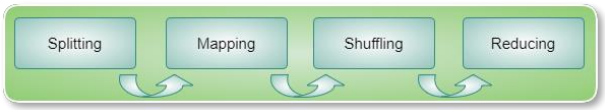



**Figure 6.** Stages of Map-Reduce

In the splitting stage, the data are divided into 64 MB or 128 MB blocks. With the mapping process, the expressions in data blocks are divided into words. In the Shuffling stage, the results found for the correct matching of the expressions formed by the mapping process are directed to the Reducer. The Reducer performs the most correct matching process by roaming on the records it has received. The reducing stage, which is the final stage, refers to the process of printing the results to the source desired (database, stream, etc.).

## 4. Method

In the sales of holiday packages, tourism agencies play an important role between the hotel and the customer. Tourism agencies make sales by offering appropriate bids to their customers through various technological applications. Tourism agencies can keep the data of the hotels with which they are contracted in their databases and also provide users with the data they provide through the web services of different agencies. For this reason, data come to the travel agency from multiple providers. The hotel records that are identical to each other or that have been previously recorded in the agency's database can be included in hotel addresses incoming from different providers. However, since the address spellings of these data expressing the same hotel and coming from different providers are different from each other, the matching of these hotels with each other in databases is possible after a certain process time. The shortening of this process as much as possible is important for agencies in the sector. This study aiming to contribute to this problem consists of three stages including data pre-processing, Map-reducing process and the implementation of different algorithms, observed in Figure 7.

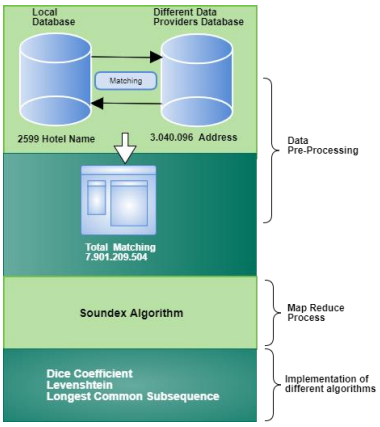


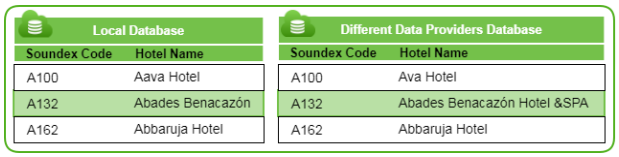

**Figure 7.** Stages of the study

### 4.1. Data Pre-Processing

In the first stage of the study, the data containing the names of 2599 hotels belonging to the city of London were sampled from the local database of the agency discussed in the study. These records needed to be individually matched with 3.040.096 address data coming to the agency's local database from seventy different providers. This matching process requires 2599 x 3.040.096 = 7.901.209.504 controlling processes. This process places a significant workload. The Map-Reduce process was implemented in the next stage to reduce this workload.

### 4.2. Map-Reduce Process

Group codes were generated for hotel names using the Soundex algorithm to decrease the number of records to be matched in the data pre-processing process. Due to a large number of hotels in the local database, the grouping process was firstly performed among the hotels to be matched. Figure 8 includes some examples of the Soundex codes that were generated for the hotel names in the local database and the hotel names coming from different providers. After all Soundex codes were generated, the hotel in the local database and the hotel names coming from different providers were matched. The number of controlling processes in the data pre-processing stage was significantly reduced by this matching process.



Local Database

| Soundex Code | Hotel Name |
|---|---|
| A100 | Aava Hotel |
| A132 | Abades Benacazón |
| A162 | Abbaruja Hotel |

Different Data Providers Database

| Soundex Code | Hotel Name |
|---|---|
| A100 | Ava Hotel |
| A132 | Abades Benacazón Hotel &SPA |
| A162 | Abbaruja Hotel |

**Figure 8.** Soundex Codes Generated

### 4.3. Implementation of Different Algorithms

Common words in data sets decrease the correct matching ratio in the SSA. For this reason, the hotel names and addresses were separated word by word, the words with a frequency of more than 1000 were sorted from many to less, and these are presented in Table 4.

**Table 4.** General Words Set

| Word | Freq. | Word | Freq. | Word | Freq. |
|---|---|---|---|---|---|
| hotel | 41621 | city | 3253 | at | 1566 |
| inn | 26703 | comfort | 3242 | del | 1473 |
| suites | 13569 | villa | 3123 | casa | 1441 |
| resort | 9281 | park | 3048 | village | 1284 |
| & | 8572 | airport | 2756 | apartment | 1265 |
| and | 6934 | residence | 2720 | b&b | 1216 |
| the | 5912 | le | 2099 | plus | 1195 |
| - | 4807 | garden | 2063 | villas | 1179 |
| beach | 4796 | palace | 1880 | centre | 1134 |
| la | 4294 | days | 1867 | motel | 1120 |
| de | 4292 | boutique | 1778 | guest | 1113 |
| apartments | 4157 | san | 1755 | el | 1084 |
| spa | 4021 | center | 1604 | north | 1062 |
| by | 3773 | hostel | 1593 | apartamentos | 1055 |

The words given in Table 4 were not taken into consideration in the algorithm calculations for the SSA to give results with higher performance. The uses of the Dice coefficient, Levenshtein and LCS algorithms, respectively, updated by removing the common words are observed in Equations (4), (5) and (6). The correct matching (score) conditions of the name and address data of the updated SSA are observed in Equation (7). The correct matching percentage was obtained as a result of centuplicating the condition expression stated in Equation 7 by the results obtained.

$$D(X,Y) = \frac{2.|X \cap Y| - (Common\ Words)}{|X| \cup |Y| - (Common\ Words)} \quad (4)$$

$$D(X,Y) = 1 - \frac{Z}{\max(|X|,|Y|) - (Common\ Words)} \quad (5)$$

$$LCS(X_i,Y_i)\begin{cases} 0 & if\ i=0\ or\ j=0 \\ 1+LCS(X_{i-1},Y_{j-1}) - (Common\ Words) & if\ X_i = Y_j \\ \max(LCS(X_{i-1},Y_j), LCS(X_i,Y_{j-1}) - (Common\ Words) & if\ X_i \neq Y_j \end{cases} \quad (6)$$

$$(score_{name} \geq 0.9 \ \ and \ score_{adress} \geq 0.9) \ or \ (score_{name} \geq 0.99 \ \ and \ score_{adress} \geq 0.7) \quad (7)$$

where X is the first word, Y is the second word, Z is the Levenshtein distance, scorename is the matching hotel name score, and scoreadress is the matching hotel address score.

### 4.4. Developed Application Software

A window of the interface of the software developed for this study is presented in Figure 9. The software was developed using the C# programming language. In the program, the user was first offered options regarding whether he would optionally perform the Map-Reduce process and whether he would use the set of common words. In addition, the selection of the algorithms used in the study was enabled. Furthermore, the areas the user could log in for the name and address scores expressed in Equation 7 were created. The process time, process count, automatic matching count and incorrect matching count are presented in the results section.

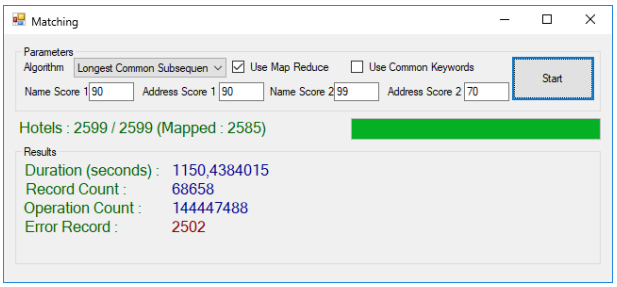



**Figure 9.** Developed Application Software

## 5. Findings

Table 5 presents a part of the Soundex code table created for 2599 hotel names belonging to the city of London obtained from the database of the agency discussed in the study.

**Table 5.** Soundex Code Table For The Hotel Names Obtained From The Local Database

| Sample Code | Soundex Code | Hotel Name |
|---|---|---|
| O1 | A000 | A Home to Rent - The Belgravia Apartment |
| O2 | A100 | Abbey |
| .O2599 | .--- | .--- |

Table 6 presents a part of the Soundex code table created for 3.040.096 hotel names coming from different data providers.

**Table 6.** Soundex Code Table For Hotel Names Coming From Different Data Providers

| Code | Soundex Code | Hotel Name |
|---|---|---|
| K1 | A000 | A |
| K2 | A000 | A - Austerlitz Hotel*** |
| K3 | A000 | A - Haven Townhouse |
| K4 | A000 | A & A Plaza Hotel |
| K5 | A000 | A & Be |
| K6 | A000 | A & EM 19 Dong Du Hotel |
| K7 | A000 | A & Em 46 Hai Ba Trung Hotel |
| K8 | A000 | A & Em Dong Du |
| K9 | A000 | A & Em Hotel - 19 Dong Du |
| K10 | A000 | A & H Suite Madrid |
| K11 | A100 | Aap Hotel & Hostel |
| K12 | A100 | Aava Hotel |
| K13 | A100 | Aava Hotel Whistler |
| K14 | A100 | Aava Whistler - Deluxe |
| K15 | A100 | Aava Whistler - Superior King (1 Bed) |
| K16 | A100 | Aava Whistler Hotel |
| K17 | A100 | Aava Whistler Hotel |
| K18 | A100 | Ab Arganda |
| K19 | A100 | Ab Hotel Arganda |
| K20 | A100 | Ab Pension Granada |
| K3.040.096 | --- | --- |

When the hotel names given in Table 5 and Table 6 are matched with each other, it is necessary to perform a matching process as a part of which is observed in Table 7. As a result of this process, the Total Process Count=7.901.209.504 and the Total Process Time (second)=94.104 were obtained.

**Table 7.** The Data Matrix Before Map-Reduce

| Local Database Sample Code | | Different Providers Sample Code | | | | |
|---|---|---|---|---|---|---|
| O1 | K1 | O2 | K1 | .. | O2599 | K1 |
| O1 | K2 | O2 | K2 | .. | O2599 | K2 |
| O1 | K3 | O2 | K3 | .. | O2599 | K3 |
| O1 | K4 | O2 | K4 | .. | O2599 | K4 |
| O1 | K5 | O2 | K5 | .. | O2599 | K5 |
| O1 | K6 | O2 | K6 | .. | O2599 | K6 |
| O1 | K7 | O2 | K7 | .. | O2599 | K7 |
| O1 | K8 | O2 | K8 | .. | O2599 | K8 |
| O1 | K9 | O2 | K9 | .. | O2599 | K9 |
| O1 | K10 | O2 | K10 | .. | O2599 | K10 |
| O1 | K11 | O2 | K11 | .. | O2599 | K11 |
| O1 | K12 | O2 | K12 | .. | O2599 | K12 |
| O1 | K13 | O2 | K13 | .. | O2599 | K13 |
| O1 | K14 | O2 | K14 | .. | O2599 | K14 |
| O1 | K15 | O2 | K15 | .. | O2599 | K15 |
| O1 | K16 | O2 | K16 | .. | O2599 | K16 |
| O1 | K17 | O2 | K17 | .. | O2599 | K17 |
| O1 | K18 | O2 | K18 | .. | O2599 | K18 |
| O1 | K19 | O2 | K19 | .. | O2599 | K19 |
| O1 | K20 | O2 | K20 | .. | O2599 | K20 |
| . | . | . | . | | . | . |
| O1 | K3.040.096 | O2 | K3.040.096 | .. | O2599 | K3.040.096 |
| Total Process Count=7.901.209.504 | | | | | | |
| Total Process Time (second)=94.104 | | | | | | |

As it is seen in Table 8, after the data were reduced by performing the Map-Reduce process, the Total Process Count=100.190.117 and the Total Process Time (second)= 1193 were obtained. Table 9 includes the performances of the Dice coefficient, Levenshtein and LCS algorithms. The incorrect data column is the column that indicates how many of 2599 records were matched incorrectly by the algorithms before common words were removed.

**Table 8.** The Data Matrix After Map-Reduce

| Local Database Sample Code | | Different Providers Sample Code | | | | |
|---|---|---|---|---|---|---|
| O1 | K1 | O2 | K1 | .. | O2599 | K100.190.108 |
| O1 | K2 | O2 | K2 | .. | O2599 | K100.190.109 |
| O1 | K3 | O2 | K3 | .. | O2599 | K100.190.110 |
| O1 | K4 | O2 | K4 | .. | O2599 | K100.190.111 |
| O1 | K5 | O2 | K5 | .. | O2599 | K100.190.112 |
| O1 | K6 | O2 | K6 | .. | O2599 | K100.190.113 |
| O1 | K7 | O2 | K7 | .. | O2599 | K100.190.114 |
| O1 | K8 | O2 | K8 | .. | O2599 | K100.190.115 |
| O1 | K9 | O2 | K9 | .. | O2599 | K100.190.116 |
| O1 | K10 | O2 | K10 | .. | O2599 | K100.190.117 |
| Total Process Count=100.190.117 | | | | | | |
| Total Process Time (second)= 1193 | | | | | | |

The updated incorrect data column gives the number of incorrect matching obtained when the common words were run without being included in the algorithm. At this point, the Dice coefficient algorithm achieved success by 99.23% while the Levenshtein algorithm achieved success by 81.45%. The LCS algorithm showed a relatively poor performance by 3.19% compared to the others.

**Table 9.** Comparison of The Performances of The Algorithms

| Sample Code | Data Count | Incorrect Data | Updated Incorrect Data | Success (%) |
|---|---|---|---|---|
| Dice coefficient | 2599 | 79 | 4 | 99.23 |
| Levenshtein | 2599 | 356 | 360 | 81.45 |
| LCS | 2599 | 2502 | 2497 | 3.19 |

The processing times of the SSAs before the common words were removed and the processing times after the common words were removed are presented in the chart in Figure 10. When the chart is examined, it is observed that there is no significant difference between the two groups in terms of duration.

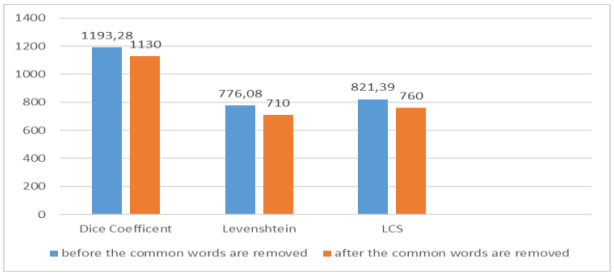



**Figure 10.** The processing times of the SSAs according to common words

In Figure 11, the time analysis that before and after the map reduce process of the names of the hotels selected as a sample in Istanbul, Berlin, Amsterdam, Bucharest and Dubai is seen. According to in Figure 11, with the map reduce operation, a gain of approximately 20% was obtained in terms of the process time (sec).

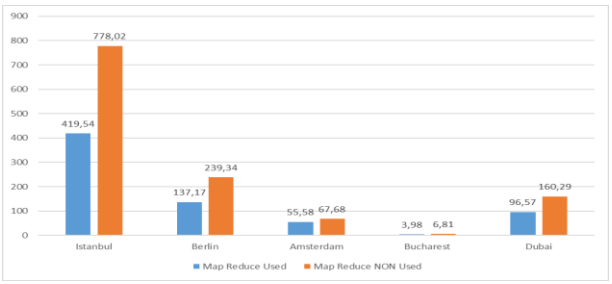



**Figure 11.** Durations by map reduced used or non used

After the map reduction process, the performances of SSAs according to the five cities given as a sample are seen in Table 10. Accordingly, it is seen that the best results are obtained with the dice coefficient algorithm. When the averages of the results of the algorithms for these five sample cities are taken, it can be concluded that the dice coefficient algorithm performs 5 times more than the levenshtein algorithm and 1.3 times higher than the LCS algorithm.

**Table 10.** The Comparison of SSAs After Map-Reduce

| City | Dice Coefficient | Levenshtein | LCS |
|---|---|---|---|
| Istanbul | 95.24% | 16.99% | 64.48% |
| Berlin | 88.95% | 12.82% | 62.57% |
| Amsterdam | 94.86% | 19.86% | 70.32% |
| Bucharest | 93.57% | 27.14% | 74.13% |
| Dubai | 92.42% | 16.48% | 68.49% |
| **Mean** | **93.00%** | **18.60%** | **68.00%** |

In figure 12, it is seen that the number of hotels matched of the dice coefficient algorithm before and after common words are used. Accordingly, with the use of common words, the accuracy of hotel matches is increased by about 50%

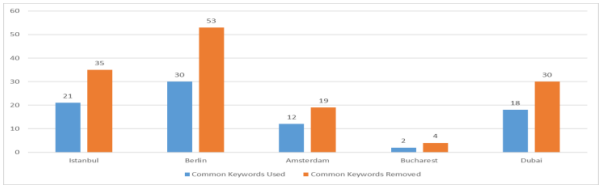



**Figure 12.** Mis-matched hotel count by common keywords used or removed.

## 5. Conclusion

The fact that the data stored in databases by being obtained from different data providers are not consistent with each other leads to many problems, especially data redundancy. When the obtained data

are of big data sizes, the matching of data becomes more complicated and takes longer. In this study, solutions were searched for the problems arising from the entry of the hotel data coming from different providers into databases with different names and addresses, using a data set of the tourism sector. For this purpose, the Map-Reduce process was performed by using the Soundex algorithm to match the data of the hotels of a tourism agency that were selected as the sample with the data of the hotels coming from different providers. Thus, a significant amount of time was saved in terms of the time required for data matching. In the final stage of the study, some SSAs were compared in terms of the data that they correctly matched, and process time. It was observed that the Dice coefficient algorithm yielded a better result in terms of correct matching. It was observed that there was no significant difference between the algorithms in terms of process time. Since each SSA will exhibit different performance on the data set used, different SSAs can be used in the following studies. The data set in this study was obtained from the tourism sector. The methods used in this study can also be applied to the data sets belonging to different sectors. The Soundex algorithm supporting the English language was used for the data set used in this study. Another one of the suggestions is the adaptation of Soundex or other algorithms to these languages in the matching of words from different languages.

**Conflict of Interest**

No conflict of interest was declared by the authors.

**References**

Bakar, Z. A., Sembok, T. M. T., and Yusoff, M., 2000. An evaluation of retrieval effectiveness using spelling-correction and string-similarity matching methods on Malay texts, Journal of the Association for Information Science and Technology, vol. 51, no. 8, pp. 691-706, doi: 10.1002/(SICI)1097-4571(2000)51:8<691: :AID-ASI20>3.0.CO;2-U

Baruah, D., and Mahanta, A. K., 2013. A new similarity measure with length factor for plagiarism detection, International Journal of Computer Applications, vol. 72, no. 14, pp. 14-17.

Baruah, D., and Mahanta, A. K., 2015. Design and development of soundex for assamese language, International Journal of Computer Applications, vol. 117, no. 9, pp. 9-12, doi: 10.5120/20581-3000

Bhatti, Z., Waqas, A., Ismaili, I. A., Hakro, D. N., and Soomro, W. J., 2014. Phonetic based soundex and shapeex algorithm for Sindhi spell checker system, Advances in Environmental Biology, vol. 8, no. 4, pp. 1147-1155.

Bird, S., Klein, E., and Loper, E., 2009. Natural Language Processing with Python. O'Reilly Press, pp. 463.

Cavoukian, A., and Jonas, J., 2012. Privacy by design in the age of big data. Information and Privacy Commissioner of Ontario, Canada, pp. 3.

Chaudhary, A., Wakchoure, N., Gotarne, N., Nath, P., and B., Dhakulkar, 2016. A comparative study on name matching algorithms, International Journal of Research in Advent Technology, vol. 4, no. 5, pp. 127-129.

Chen, X., and Zhou, L., 2015. Design and implementation of an intelligent system for tourist routes recommendation based on Hadoop, 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, pp. 774–778. doi: 10.1109/ICSESS.2015.7339171

Chowdhury, S. R., Hasan, M. M., Iqbal, S., and Rahman, M. S., 2014. Computing a longest common palindromic subsequence, Fundamenta Informaticae, vol. 129, no. 4, pp. 329-340, doi: 10.3233/FI-2014-974

Dice, L. R., 1945. Measures of the amount of ecologic association between species, Ecology, vol. 26, no. 3, pp. 297-302.

Dursun, B., and Sonmez, A. C., 2008. A new method for computing the similarity of Turkish texts, IEEE 16th Signal Processing, Communication and Applications Conference, Aydın, pp. 76. doi: 10.1109/SIU.2008.4632581

Freeman, A. T., Condon, S. L., and Ackerman, C. M., 2006. Cross linguistic name matching in English and Arabic: a one to many mapping extension of the Levenshtein edit distance algorithm, in proc. Main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. Association for Computational Linguistics, pp. 471-478, doi:10.3115/1220835.1220895

Fuentes, A. A. G., Parra, I. P., Quevedo-Torrero, J. U., and Perez, R. D., 2016. Comparative analysis of phonetic algorithms applied to Spanish," International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, pp. 1180-1185, doi: 10.1109/CSCI.2016.0223

Gupta P., and Upadhyay, A., 2015. Sentiment and predictive analysis of big data for hotel reviews, International Journal of Software & Hardware Research in Engineering, vol. 3, no. 5, pp. 78–86.

Heeringa, W. J. 2004. Measuring dialect pronunciation differences using Levenshtein distance, Groningen: s.n, pp.323.

Ilhan, S., Duru, N., Karagoz, S., and Sagir, M., 2008. Metin madenciligi ile soru cevaplama sistemi, Electrical – Electronics - Computer Engineering Symposium, Bursa, pp. 356-359.

Jaisunder, G. C, Ahmed, I., and Mishra, R. K., 2017. Need for customized soundex based algorithm on indian names for phonetic matching, Global Journal of Enterprise Information System, vol. 8, no. 2, pp. 30-35, doi: 10.18311/gjeis/2016/7658

Jiang, Y., Deng, D., Wang, J., and Li, G., 2013. Efficient parallel partition based algorithms for similarity search and join with edit distance constraints, in Proc. Joint EDBT/ICDT 2013 Workshops, Genoa. doi: 10.1145/2457317.2457382

Kisla, T., Karaoglan, B., and Metin, S. K., 2015. Extracting the Features of Similarity in Short Texts. IEEE 23th Signal Processing And Communications Applications Conference, Malatya, pp. 180-183, doi: 10.1109/SIU.2015.7130443

Kruskal, J. B., and Sankoff, D., 1999. Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison. Stanford, CA: CSLI Publications.

Kurdziel, L. B. F., and Spencer, R. M. C., 2016. Consolidation of novel word learning in native English-speaking adults, Memory, vol. 24, no. 4, pp. 471-481, doi: 10.1080/09658211.2015.1019889

Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Doklady, vol. 10, no. 8. pp. 707-710.

Li, G., Deng, D., and Feng, J., 2013. A partition-based method for string similarity joins with edit-distance constraints, ACM Transactions on Database Systems (TODS), vol. 38, no. 2, pp. 1–33, doi: 10.1145/2487259.2487261

Li, X., Pan, B., Law, R., and Huang, X., 2017. Forecasting tourism demand with composite search index, Tourism Management, vol. 59, pp. 57-66, 2017. doi: 10.1016/j.tourman.2016.07.005

Liu, Y., Teichert, T., Rossi, M., Li, H., and Hu, F., 2017. Big data for big insights: Investigating language-specific drivers of hotel satisfaction with 412,784 user-generated reviews, Tourism Management, vol. 59, pp. 554–563.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C., 2002. Text classification using string kernels, Journal of Machine Learning Research, vol. 2, pp. 419-444.

Miah, S. J., Vu, H. Q., Gammack, J.,. and McGrath, M., 2017. A big data analytics method for tourist behaviour analysis, Information & Management, vol. 54, no. 6, pp. 771-785, doi: 10.1016/j.im.2016.11.011

Mutalib N. S. A., and Noah, S. A., 2011. Phonetic coding methods for Malay names retrieval," International Conference on Semantic Technology and Information Retrieval, Putrajaya, pp. 125-129. doi: 10.1109/STAIR.2011.5995776

Naumann, F., and Herschel, M., 2010. An introduction to duplicate detection," Synthesis Lectures on Data Management, vol. 2, no.1, pp. 1-87, doi: 10.2200/S00262ED1V01Y201003DTM003

Nyirarugira, C., and Kim, T., 2015. Stratified gesture recognition using the normalized longest common subsequence with rough sets, Signal Processing: Image Communication, vol. 30, pp. 178-189, doi: 10.1016/j.image.2014.10.00844.

Odell, M., and Russell, R., 1918. The soundex coding system, US Patents 1261167.

Onder, I., 2017. Classifying multi-destination trips in Austria with big data, Tourism Management Perspectives, vol. 21, pp. 54-58, doi: 10.1016/j.tmp.2016.11.002

Parmar, V. P., and Kumbharana, C. K., 2014. Study existing various phonetic algorithms and designing and development of a working model for the new developed algorithm and comparison by implementing it with existing algorithm (s), International Journal of Computer Applications, vol. 98, no. 19, pp. 45-49.

Peng, X., and Huang, Z., 2012. Enabling semantic queries against the spatial database, Advances in Electrical and Computer Engineering, vol. 12, no.1, pp. 45-50, doi: 10.4316/AECE.2012.01008

Sagiroglu, S., and Sinanc, D., 2013. Big data: A review, International Conference on Collaboration Technologies and Systems (CTS), San Diego, pp 42-47. doi: 10.1109/CTS.2013.6567202

Shedeed, H. A., and Abdel, H., 2011. A new intelligent methodology for computer based assessment of short answer question based on a new enhanced soundex phonetic algorithm for Arabic language, International Journal of Computer Applications, vol. 34, no. 10, pp. 40-47.

Shrote, K. R., and Deorankar, A. V., 2016 Hotel recommendation system using hadoop and

mapreduce for big data, International Journal of Computer Science, Information Technology, and Security, vol. 6, no. 2, pp. 137–141.

Stein-Smith, K., 2016. The US Foreign Language Deficit: Strategies for Maintaining a Competitive Edge in a Globalized World. Palgrave Macmillan, pp. 21, doi: 10.1007/978-3-319-34159-0

Su, Z., Ahn, B. R., Eom, K. Y., Kang, M. K., Kim, J. P., and Kim, M. K., 2008. Plagiarism detection using the Levenshtein distance and Smith-Waterman algorithm, 3rd International Conference on Innovative Computing Information and Control, Dalian, Liaoning, pp. 0-3. doi: 10.1109/ICICIC.2008.422

Tabataba F. S., and Mousavi, S. R., 2012. A hyper-heuristic for the longest common subsequence problem, Computational Biology and Chemistry, vol. 36, pp. 42–54, doi: 10.1016/j.compbiolchem.2011.12.004

Toole, J. L., Colak, S., Sturt, B., Alexander, L. P., Evsukoff, A., and González, M. C., The path most traveled: Travel demand estimation using big data resources, Transportation Research Part C: Emerging Technologies, vol. 58, pp. 162-177, 2015. doi: 10.1016/j.trc.2015.04.022

Ugon, A., T. 2015. Nicolas, M. Richard, P. Guerin, P. Chansard, C. Demoor, and L. Toubiana, "A new approach for cleansing geographical dataset using Levenshtein distance, prior knowledge and contextual information, Medical Informatics Europe, Madrid, pp. 227-229. doi: 10.3233/978-1-61499-512-8-227

Xiang, L. , Jiang, N., Ya-ting, Y., Xi, Z., and Cheng-gang, M., 2014. Application of generalization language model in Chinese-Uyghur machine translation, Application Research of Computers, vol. 31, no. 10, pp. 2994-2997, doi: 10.3969/j.issn.1001-3695.2014.10.026.

Xiang, Z., Schwartz, Z., Gerdes, J. H., and Uysal, M., 2015. What can big data and text analytics tell us about hotel guest experience and satisfaction? International Journal of Hospitality Management, vol. 44, pp. 120-130, doi: 10.1016/j.ijhm.2014.10.013

Yahia, M. E., Saeed, M. E., and Salih, A. M., 2006. An intelligent algorithm for Arabic soundex function using intuitionistic fuzzy logic, 3rd International IEEE Conference Intelligent Systems, London, pp. 711-715. doi: 10.1109/IS.2006.348506

Zikopoulos, P., and Eaton, C., 2011. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. Mcgraw-Hill Osborne Media Press, pp. 176.