



Duplicate product record detection engine for e-commerce platforms

Osman Semih Albayrak^{a,b,*}, Tefvik Aytekin^c, Tolga Ahmet Kalaycı^{a,d}

^a Hepsiburada, İstanbul, Turkey

^b Department of Computer Engineering, Marmara University, İstanbul, Turkey

^c Department of Computer Engineering, Bahcesehir University, İstanbul, Turkey

^d Department of Industrial Engineering, İstanbul Technical University, İstanbul, Turkey

ARTICLE INFO

Keywords:

Duplicate record detection

Feature engineering

Text similarity

Classification

ABSTRACT

Having a clean product catalog and keeping it complying with the standards of the industry is one of the primary concerns of e-commerce companies. Integrating product data from multiple providers confronts the companies with a challenging issue: duplicate product records. Since it is possible to describe a product with a variety of different words, images and attributes, detecting duplicate product records is a difficult task to overcome. In this work, a novel duplicate record detection engine is proposed for an e-commerce company, Hepsiburada. The engine is developed based on a real-world dataset. In order to build a training set we use text similarity and domain-specific distance metrics for generating candidate duplicate product pairs which are then labeled by human experts. We performed extensive feature engineering and state-of-the-art classification models to determine whether any two products are duplicated or not. The experimental results show that our engine is able to detect duplicate product records with high precision and outperforms the accuracy of non-adaptive methodologies.

1. Introduction

In today's highly competitive markets, clean data is an important necessity for all kinds of industries. As an integral part of almost all business fields of commerce, e-commerce requires closer inspection of data. A well-structured and clean product catalog is a crucial necessity for all operations of e-commerce companies. Regardless of how many products an e-commerce company has in its catalog, keeping the data clean provides better operations, better analytics, better marketing, and improvement in sales.

20 years ago, e-commerce was a newborn market. It has made tremendous growth in the past 20 years and has become one of the most important players of the trade industry today. Fig. 1 shows the results of a recent OECD report which suggests that more than half of all individuals in OECD countries have made online purchases in 2018 (OECD, 2019).

When it comes to e-commerce, market growth often means catalog growth and catalog growth means new product entries to the catalogs which leads to more complex catalog structures. As an example, it can be seen from Fig. 2 that the product catalog of Hepsiburada had a splendid growth from the year 2015 to 2020. Every single year the catalog grew up with millions of products. During 2019, Hepsiburada expanded its catalog with nearly 8 million new products. Numbers in the year 2020 show the impact of the COVID-19 pandemic on e-commerce.

Such catalog growths, of course, come with their own difficulties. Even sub-sectors are born to overcome the management difficulties of colossal catalogs. To manage the growth and keep the product catalog clean and well structured, e-commerce companies invest in Product Information Management solutions. A Product Information Management solution is a central platform to gather, manage and enrich the product information and create a product catalog. However, in daily business, keeping the catalog clean is sometimes an impossible task. Hundreds or even thousands of new products may be uploaded to the catalog from providers in a single day. Since there is no automated content control mechanism and different providers may upload the same products with different attributes (name, image, description, etc.), catalogs are getting dirtier day by day at each upload. As a consequence of deficient control mechanisms, while the catalogs are expanding, duplicate product records occur. A duplicate product record means that, for some reason, two or more product records exist in the catalog for the same product. An example where the duplicates have the same product name and image is given in Fig. 3.

On the other hand, a duplicate product pair may not have identical product names or images. Fig. 4 illustrates a duplicate product pair with similar product names. Textually, product names are not identical, however they are pointing out to the same product.

* Corresponding author at: Department of Computer Engineering, Marmara University, İstanbul, Turkey.

E-mail addresses: semihalbayrak@marun.edu.tr (O.S. Albayrak), tevfik.aytekin@eng.bau.edu.tr (T. Aytekin), kalaycit@itu.edu.tr (T.A. Kalaycı).

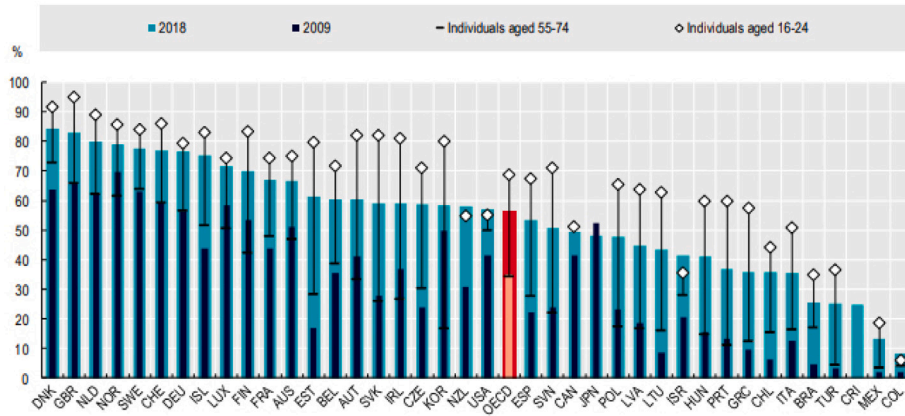


Fig. 1. Percentage of individuals who have purchased online in 2018.
Source: OECD, ICT Access and Usage by Households and Individuals (database), <http://oe.cd/hhind> (accessed February 2019).

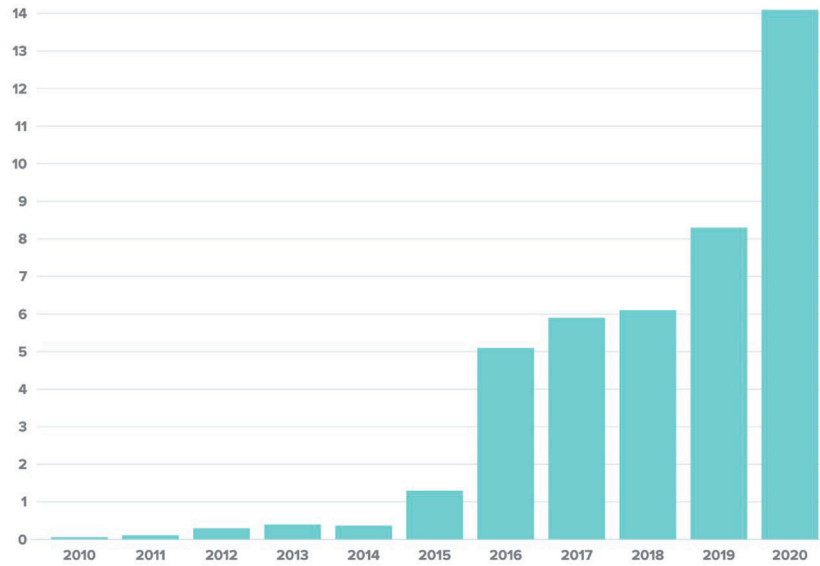


Fig. 2. Annual product entry counts of Hepsiburada. (Entry counts in the y-axis are in millions.).



Fig. 3. An example of a duplicate product record with identical text and image.

Fig. 5 illustrates another scenario of duplicate product records. Two products in Fig. 5 have similar but not identical images. However, they are again referring to the same product.

Another reason for the presence of duplicate product records is that sometimes vendors knowingly create duplicate product records. Products with a large number of vendors make competition difficult



Fig. 4. An example of a duplicate product record with similar text.

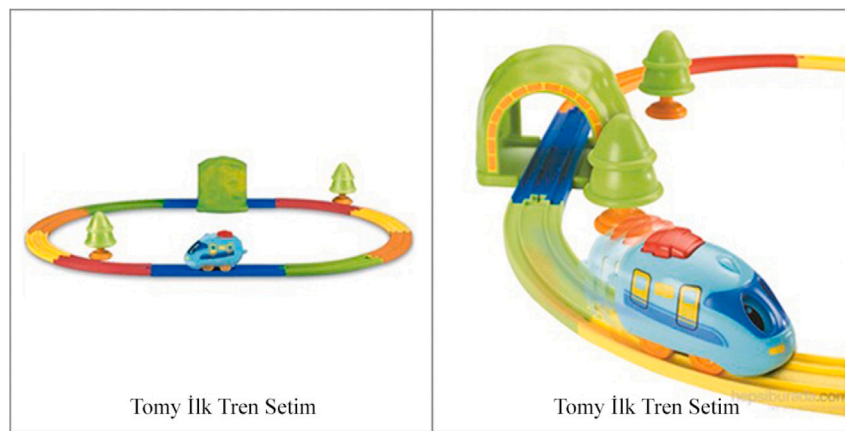


Fig. 5. An example of a duplicate product record with similar image.

for vendors. For example, for a product with 20 vendors, a significant portion of the vendors are not visible to customers on the page, as they are in lower ranks. Vendors try to overcome this problem by opening duplicate product records. When they open a new record for the same product, they become the only seller of that product.

Since Hepsiburada users accustomed to find all the information about the product they are looking for on a single page, they also accustomed to find all the vendors and all the variants of the product on a single page. Therefore the journey of the users is retrograding by the duplicates.

Also, a wide range of other complications surrounds the duplicate product record problem. Duplicate product records affect the performance of product recommendation engines negatively. Basically, the product recommendation engines recommend the products according to the condition of being viewed and/or sold together. The presence of duplicate product records makes it difficult for product recommendation engines to learn this relationship between products accurately. On the other hand, product listing pages and search engines are also negatively affected by duplicate product records. Product listing pages and search results pages are showcases for e-commerce companies. Listing the same product twice or more on these pages means being deprived of showing other products to the customer. Duplicate product records also negatively affect the market performance measurements and stock management of the products. For these reasons, it is crucial to study the problem of identifying duplicate products.

This paper proposes a detailed, industrially tested and currently in use methodology to identify duplicate product records. Our major contributions are as follows: first, we make an extensive feature engineering to address the problem from various perspectives. Second,

our model can adapt the product naming convention differences of brands and categories. Third, we apply a text processing special to the e-commerce domain. Finally, the results show that our design is able to detect duplicate records with high precision.

The rest of the paper is organized into seven sections. Section 2 discusses related work. Section 3 gives an explanatory analysis of the data we use in this study. Section 4 describes the methodology for constructing a duplicate product records dataset for building machine learning models. Section 5 describes a classification model which uses text-based features. Section 6 describes the enriched model which uses additional features to further improve model performance. Section 7 gives the experimental results. Finally, Section 8 concludes the paper and discusses future work.

2. Related work

An important amount of previous work has been made on the field of duplicate product record detection. Although in this work we prefer to entitle the issue as duplicate product record detection, several other terms such as duplicate detection (Monge, & Elkan, 1997; Sarawagi, & Bhamidipaty, 2002), product matching (Li, Dou, Zhu, Zuo, & Wen, 2020), reference matching (McCallum, Nigam, & Ungar, 2000), name matching (Cohen, Ravikumar, Fienberg et al., 2003; Raeesi, Asadpour, & Shakeri, 2020), entity resolution (Fisher, Christen, & Wang, 2016), detecting plagiarism (Hoad, & Zobel, 2003; Roostaei, Fakhrhmad, & Sadreddini, 2020), record linkage (Winkler, 1999), and near-duplicate detection (Xiao, Wang, Lin, Yu, & Wang, 2011) are used in previous researches.

Previous studies can be grouped into two as text-based and adaptive methods. Also blocking methods are used to speed up the computations. Below we will look at each in turn.

2.1. Text based methods

Traditional text-based methods are mainly categorized into three groups. The first group of methods is based on character-based techniques such as deletions, insertions, sequence, and subsequence comparisons (Cohen et al., 2003; Levenshtein, 1966). They can roughly be named as edit distance-like techniques. The goal of these methods is to calculate the minimum number of edit operations (character deletion, insertion, substitution) that converts one of the given two strings to the other. The idea is, the count of edit operations would simply give the distance among two given strings. Although the edit distance-like techniques have the advantage of simplicity, they are only good for short strings. When it comes to longer strings edit distance calculation will be computationally expensive. The other disadvantage of edit distance-like techniques is that they do not take the semantic meanings into account.

The second group of methods is based on comparing two strings by looking at the tokens of the strings (Arasu, Chaudhuri, & Kaushik, 2008; Cohen et al., 2003). Jaccard similarity is a typical example of token-based methods. For given two strings A and B , to calculate Jaccard similarity, one simply converts the given strings into two sets of tokens A' and B' then divides the size of the intersection of the sets A' and B' by the size of the union of the sets A' and B' .

$$Jaccard(A', B') = \frac{A' \cap B'}{A' \cup B'} \quad (1)$$

Unlike the edit distance-like techniques, token-based similarity techniques have the advantage of computational efficiency. They are applicable for relatively long texts and they can take semantic meanings into account. However, they have their own disadvantages as well. First, they are not efficient for single words or short phrases of several words. Second, they are not able to detect recurrent words in the given documents. Third, token-based methods simply give equal importance to each word in a given document. Therefore, the question “How important a word in a given document?” is unanswered.

The final group of methods is based on converting the given two documents into vectors and applying a similarity measure (such as cosine) on these vectors (Hassanzadeh, & Consens, 2009; Koudas, Marathe, & Srivastava, 2004; Tata, & Patel, 2007). The most common way to do this is to use a bag of words approach with TF-IDF (‘Term Frequency - Inverse Document Frequency’). TF-IDF measures the importance of each word in a given document based on how often a word appears in that document and also in the given collection of documents. The idea behind the TF-IDF is, if a word appears frequently in a document, it means the word is important and should have a high importance score. However, if a word appears frequently in the given collection of documents, it means the word is not a unique identifier, therefore a lower importance score should be given to the word. The formula of TF-IDF is:

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D) \quad (2)$$

Where the t stands for a term, d stands for a document, and D stands for the collection of documents. Although there are variations of the exact calculations, our calculation is as follows. The first part of the formula, $TF(t, d)$, calculates the number of times the term t appears in document d divided by the total number of terms in the document d . ‘Term Frequency’ basically scores the importance of the term t in the document d . The second part of the formula, $IDF(t, D)$ calculates the log of the number of documents in the collection D divided by the number of documents that contain the term t . ‘Inverse Document Frequency’ basically scores the term according to how rare it is in the collection. Finally, the TF-IDF score of a term would be equal to the multiplication of the TF score and the IDF score.

In our experiments, the TF-IDF approach gives the best performance among text similarity algorithms for duplicate product record detection problem. Therefore we used the TF-IDF approach as the primary text similarity metric when constructing the training dataset. Cohen et al. (2003) also confirms that TF-IDF generally gives the best results on the problem of matching strings.

2.2. Adaptive methods

Despite the relative effectiveness of traditional similarity methods to solve the duplicate detection problem (Köpcke, Thor, & Rahm, 2010) compares several entity matching methodologies and proposes that the learning-based match strategies outperform the non-learning approaches. Bilenko, and Mooney (2003) also states that rather than tuning traditional string similarity metrics, using the adaptive approaches have clear advantages to identify duplicate records. Their work shows that adaptive methods are capable to learn domain-specific naming conventions. de Carvalho, Gonçalves, Laender, and da Silva (2006), Cohen, and Richman (2002), Thor, and Rahm (2007) also proposes an adaptive method to detect duplicates by combining multiple similarity metrics.

The idea behind adaptive methods is to overcome the adaptation problem by combining various traditional similarity methods into a single dataset as independent features and training a classification model to decide whether two given documents are duplicated or not Elmagarmid, Ipeirotis, and Verykios (2006).

Similar to our problem, Ghani, Probst, Liu, Krema, and Fano (2006) describes the work they have done as extracting attribute and value pairs from textual product descriptions to represent products as a set of attributes and values. Similar to our solution design, using the traditional text similarity algorithms, they first create a dataset representing the products and treated the problem as a learning problem. Ristoski, Petrovski, Mika, and Paulheim (2018) uses image similarities with text similarities to match products from different e-commerce websites. Hadi Kiapour, Han, Lazebnik, Berg, and Berg (2015) on the other hand, studies only the image similarities to match products of the apparel domain.

Our study has shown that since the text, image, and price features are representations of a product in different spaces, using them together as independent features increase the performance of the classification models and gives better results. Also, as explained in Section 3, the representation of the brands in the dataset with features that will characterize their product naming habits also increases the performance of the model. To the best of our knowledge, characterizing the documents according to the groups that they belong to is not studied before.

2.3. Blocking methods

Since the catalogs of the e-commerce companies consist of millions of product records it is impractical to calculate similarity among all possible pairs of products which has $O(n^2)$ complexity. Due to this quadratic complexity of matching algorithms, a technique called blocking is developed. The goal of the blocking techniques is simply to reduce the number of comparisons. While using the blocking techniques, it is important to keep missed match count as small as possible (Papadakis, Skoutas, Thanos, & Palpanas, 2020).

As stated by O'Hare, Jurek-Loughrey, and de Campos (2019), blocking rules can be developed manually or can be found heuristically or can be learned automatically, yet each technique has its difficulties: Developing blocking rules manually requires some domain expertise, learning the rules requires a labeled dataset, and heuristic approaches require manual optimizations. In this work, since we have domain expertise, we divided the dataset into small pieces by a disjunctive variable. The details of the blocking method we use will be explained in Section 3.1.

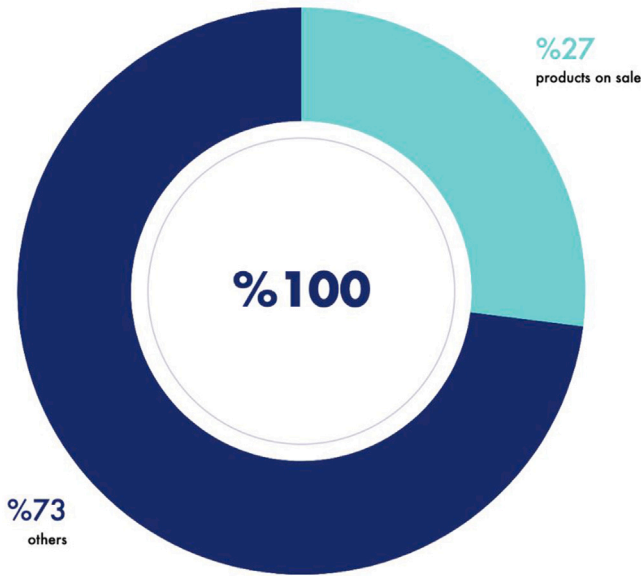


Fig. 6. The ratio of products that are on sale.

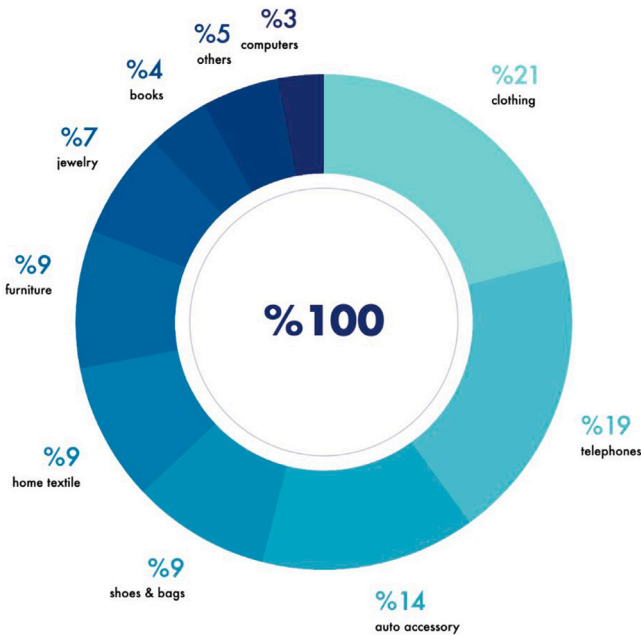


Fig. 7. The distribution of the products by categories.

3. Explanatory analysis

In this section, we will summarize the results of our investigations of the product catalog. Investigations made on the product catalog have shown that the catalog consists of approximately 43 million products. As shown in Fig. 6, approximately 14 million of these products are currently on sale.

The distribution of products on sale by categories is shown in Fig. 7. There are 30 main categories in the catalog. As can be seen from Fig. 7, categories such as clothing, telephones, auto accessory, shoes & bags, furniture, home textile, jewelry, books, and computers constitute ninety-five percent of the total number of products on sale.

The products, which are divided into 30 main categories as above are defined under approximately 75,000 different brands. While a significant portion of the brands contains products from a single category,

some brands have products from multiple categories. For example, Samsung branded products are distributed in computers, telephones, photos & cameras, and consumer electronics categories.

3.1. Blocking methodology

To decrease the number of pairwise comparisons (which is quadratic with respect to the number of products) we divide the dataset into meaningful chunks. Since searching for duplicated products in 43 millions of individual products would need trillions of comparisons, to be able to develop a scalable solution, ways to divide the dataset into smaller pieces are investigated. During the investigation, one of the primary findings is that 29 million products in the catalog are inactive, which means that they are not on sale. This finding reduced our dataset into 14 million individual products.

However, the problem of trillions of comparisons is still on the stage. In further investigations, a catalog-specific rule is uncovered. We find that if two products are duplicated, they are almost always taking part under the same brand and the same main category. If two products belong to different brands and/or different main categories, it is extremely unlikely that the two are duplicated. Products of different brands and/or different main categories cannot be identical. For example, an Adidas shoe record cannot be identical to a Puma shoe record. An Adidas shoe record can only be identical to another Adidas shoe record. As another example, a mobile phone that belongs to Samsung cannot be identical to a computer that belongs to Samsung. A Samsung mobile phone can only be identical to another Samsung mobile phone.

Therefore the catalog is divided into segments by brands and main categories of the products to reduce the comparison counts and to create a scalable solution.

3.2. Stop words

In the product catalog, almost all product names contain the word that corresponds to their brands. As an example, the product name of an iPhone begins with the word 'Apple'. On the other hand, as explained in Section 3.1, two products can be duplicate versions of each other only if they belong to the same brand. Therefore the duplicate detection engine compares any product of Apple only with another product of Apple. Since the product names of Apple contain the word 'Apple', the word 'Apple' can be considered as a stop word. For these reasons, in this work, words that correspond to brands are considered as stop words and removed from the product names.

Contrary to the classical text mining studies, classical stop word removal approaches do not fit into this specific problem of the e-commerce domain. Common stop words are actually important words for e-commerce product catalogs. An example to illustrate this scenario is given in Fig. 8.

Two products are shown in Fig. 8. The name of the first product is 'Bruder Cat Mini Excavator and Construction Worker' and the name of the second product is 'Bruder Cat Mini Excavator'. For any string similarity algorithm, the product names of these two products would be calculated as relatively similar. As an example, the Jaccard similarity of these two products is equal to 0.57. Since the word 'and' is a classical stop word, if we remove this stop word from the product name, then the Jaccard similarity will be 0.66. Removing the stop word would increase the similarity of given product names. However, in this example, it can be seen that the word 'and' is important to differentiate the two products. The word 'and' actually adds a new item to the 'Bruder Cat Mini Excavator' and makes products different. Therefore, instead of removing the stop word 'and', a rule is developed as follows: If one of the product names contains the word 'and' and the other does not, then the two products are not identical.



Fig. 8. An example of stopwords. The word 've' means 'and' in Turkish.x.

3.3. Hero products

Vendors tend to create duplicates of bestsellers or most viewed products more than other products. To prevent the closure of bestsellers or most viewed products, a hero product detection algorithm is developed. The definition of a hero product is as follows: products with high metrics such as sales, views, number of unique customers are defined to be hero products.

- Traffic and sales data of the products are collected for the last two years. Following statistics are created using the collected data: the number of unique customers viewing the product page, the number of views of the product page, the number of unique customers who bought the product, the number of unique orders where the product is sold, the total net order amount of the product, the total number of sales of the product.
- The collected statistics are scored between 0 and 100 by performing percentile analysis. Products with a score of 95 and above in any statistic are considered as hero products. For example, view heroes are products with a score of 95 or above in any of the view metrics. Sales heroes are also products with a score of 95 and above in any of the sales metrics.
- If there are products defined as a hero in the duplicate product pairs, these products are not closed.

3.4. Naming conventions

Different brands have different conventions when it comes to naming their products. Traditional similarity metrics fail to adapt to such differences and for this reason, they fail to detect duplicate product records. As an example, in the dataset, there are such fashion brands that give the same name to hundreds of different products: 'Pregnant Woman's Blouse'. Although the names of these products are the same, they are not duplicate products. On the other hand, there are such brands that give reasonably different names to the products such as 'Woman's Short Sleeve T-Shirt' and 'Woman's Shoulder Low-Cut Printed Short Sleeve T-Shirt'. For this type of brands, a relatively low text similarity score can mean that products might be duplicated. Since it is impossible to tune different thresholds for thousands of different brands, traditional string similarity metrics have their disadvantages to decide whether the two given products are duplicated or not. To overcome the naming differences of the brands, we decided to combine traditional string similarity metrics in adaptive learning methodologies (Bilenko & Mooney, 2003; Lin, Liao, & Lee, 2013).

3.5. Solution design

As a result, a three-phase method is created for the solution of the problem. The first phase consists of using traditional string similarity algorithms to find product pairs that are candidates of being identical and sending the pairs to human referees to be labeled as 'duplicate' or 'not duplicate'. In this phase, a trainable dataset labeled by human referees is obtained. This phase will be detailed in Section 4.

The second phase is to train a classification model that is able to differentiate duplicate product pairs from not duplicate ones. The model is trained based on the dataset created in the first phase. This phase will be detailed in Section 5.

The third and final phase is to enrich the classification model trained at the second phase with the image similarity scores of the product pairs. This phase will be detailed in Section 6.

4. Construction of the training set

This section will summarize the methods of finding product pairs that are a candidate for being duplicate of each other by using traditional string similarity algorithms.

4.1. Data cleaning and standardization

To standardize the textual attributes of the products such as product name and product description, several operations are executed as follows:

- All textual attributes are converted to lower case.
- Numbers written in the form of letters are converted into numerical equivalents.
- Punctuations and special characters are removed from attributes.
- Adjacent numerical and textual expressions are separated. As an example, expressions like '1 cm' or '1 kg' are separated as '1 cm' and '1 kg'.
- All textual attributes are tokenized into a list of words. As an example the product name 'iphone 6 64 GB' is tokenized into a list of words as ['iphone','6','64','GB'].

4.2. Catalog specific rules

To be able to differentiate product pairs regardless of their text similarity scores, catalog-specific rules are developed as follows:

- **First word difference:** After the removal of the brand words, the leading words of the remaining text of the product name should be the same. As an example, for a product whose name is 'iphone 6 32 GB', the leading word 'iphone' stands for the type of product.

Therefore, the leading word of a duplicate product's name should be 'iphone' as well. If it is not, if the first word of a candidate duplicate's name is 'ipad' then, regardless of their text similarity score it can be decided that these two products are not duplicate.

- **Category word difference:** A word list is produced from the hierarchy names of the product catalog and named as category word list. The following rule is developed: If two products are duplicate and both of them contain a category word then the category words they have should be the same word in order to define them as duplicate. As an example, if product *A* contains the word 'boot' and product *B* contains the word 'slipper' then regardless of their text similarity score it can be decided that these two products are not duplicate. Although comparisons are made only between products under the same main category and under the same brand, the reason for the development of such a rule is that products with misidentified hierarchies can also be found in the catalog.
- **Numerical difference:** The numerical entities of the product names must be identical if the two product names have the same number of numeric entities. The reason is that numerical entities are explaining either the measure of the products or the version of the products. As an example, if product *A* and product *B* is duplicate and the product name of product *A* is 'iphone 6 32 GB' then the product name of product *B* can be neither 'iphone 7 32 GB' nor 'iphone 6 64 GB'. The product name of product *B* should have the same numerical entities as product *A* in order for them to be duplicate. On the other hand, if the product name of product *A* is 'iphone 6 32 GB' and product name of product *B* is 'iphone 32 GB' products remain as candidate duplicate products. Also if the product name of product *A* contains the words '4 GHz 6 MB Cache Processor' and the name of product *B* contains the words '6 GHz 4 MB Cache Processor', since products have the same numeric entities in different orders, they remain as candidate duplicate products for the decision of the model.
- **First three letter rule:** Due to the agglutinative nature of Turkish, most of the typos (made intentionally or unintentionally) occur after the stem form of the words. Therefore, to be able to detect such product pairs that have relatively low textual similarity scores because of the typos, we developed a rule that takes the first three letters of every word of the product names and constructs secondary product names. As an example, by this rule, the secondary name of the 'iphone 7 32 GB' would be equal to 'iph 7 32 GB'. If the Jaccard similarity of the secondary names of the products is greater than the Jaccard similarity of the primary names then it would be a signal for the possibility of typos. For this reason, the product pair will remain as a candidate.

4.3. Candidate duplicate product records generation

Since initially we do not have labeled product pairs, using the traditional string similarity methods and catalog-specific rules we generate a candidate duplicate product pair list with the pairs that have similarity scores above light thresholds and pass catalog-specific rules. Then the candidate duplicate product pair list is sent to the human referees for being labeled as 'duplicate' or 'not duplicate'.

In the following we summarize the steps of the candidate duplicate product records list generation process:

1. Standardize the dataset.
2. Divide the dataset into clusters by brands and main categories.
3. For each cluster do the following steps:
 - (a) Take the product names that belong to a cluster and create their TF-IDF vectors.
 - (b) Take the attributes of the first product of the cluster, name it as anchor product and remove the anchor product from the cluster.

(c) For each of the remaining products of the cluster do the following:

- i. Check if the first words of the anchor product and the candidate product are identical. If they are not identical, then the anchor product and the candidate product are not duplicate. Skip to the next candidate product.
- ii. Check if the category words of the anchor product and the candidate product are identical. If they are not identical, then the anchor product and the candidate product are not duplicate. Skip to the next candidate product.
- iii. Check if the numerical entities of the anchor product and the candidate product are identical. If they are not identical, then the anchor product and the candidate product are not duplicate. Skip to the next candidate product.
- iv. Check the TF-IDF cosine similarity score of the anchor product's name and the candidate product's name.

- A. If the similarity score is smaller than 0.3 then the anchor product and the candidate product are not duplicate. Skip to the next candidate product.
- B. If the similarity score is between 0.3 and 0.5 then check if the first three letters Jaccard similarity score of the product names is greater than their Jaccard similarity score. If the first three letters Jaccard similarity score is greater, it means one of the products may have a spelling mistake, products may be duplicate. Add the anchor product and the candidate product to the candidate duplicate product records list else skip to the next candidate product.
- C. If the similarity score is greater than 0.5 products may be duplicate. Add the anchor product and the candidate product to the candidate duplicate product records list else skip to the next candidate product.

(d) If the remaining product count of the cluster greater than 1, turn back to step 3b. Else, skip to the step 4.

4. Send the generated candidate duplicate product records list to human referees to get each pair labeled as 'duplicate' or 'not duplicate'. Table 1 illustrates a sample output of the candidate duplicate product records list generation process. The term 'P1' in Table 1 refers to the 'anchor product'. The term 'P2' refers to the 'candidate product'. The term 'Hero' simply refers to which one of the anchor product and the candidate product is a hero product. If none of them is a hero product, then it is denoted with 'None'. The term 'Similarity' corresponds to the TF-IDF cosine similarity score of the anchor product and the candidate product.

4.4. Duplicate product record detection in production

To generate candidate duplicates which will be given to the trained model for classification, we apply the same rule-set as explained in Section 4.3. The reasons for doing so are as follows:

1. We aim to equalize the candidate generation logic used in building the training set and used for submitting candidates to the trained model in order to rule out the biases that might be inherited from the sampling method we applied.

#	Benzer SKU	SKU Adı	Marka	Ürün Tipi	Kategori	Katalog	Mal Grubu	Hero	Resim
<input type="radio"/>	HBV000001EQTF	Miglior Gatto Av Hayvanlı Kedi Konservesi 405 Gr	Miglior Gatto	Kedi Maması	Yetişkin Kedi Konserveleri	PetShop	6824	Evet	
<input type="radio"/>	HBV0000056E84	Miglior Gatto Av Hayvanlı Kedi Konservesi 405 Gr	Miglior Gatto	Kedi Maması	Yetişkin Kedi Konserveleri	PetShop	6824		

[Karşılaştır](#)

Fig. 9. An example of a candidate duplicate product pair listed on the deduplication platform.

Table 1

A sample output of candidate duplicate product records list.

Product names	Hero	Similarity
P1: Wee Baby P.P Kulplu Biberon	None	0.64
P2: Wee Baby Pp 744 Kulplu Plastik Biberon 150 ml		
P1: Fixplant 3 Şişe 84 gr Saç Dolgunlaştırıcı	P1	0.77
P2: Fixplant 3 Şişe 84 gr Saç Tozu		
P1: ANS Kedi Göğüs Bel TasmasıYeşil	None	0.83
P2: Ans Kedi Göğüs Bel Tasması		
P1: Alemdar Karton Klasör Geniş	P2	1.00
P2: Alemdar Karton Geniş Klasör		
P1: Vıga Toys 2D Kaplumbağa Puzzle	None	0.54
P2: Vıga Toys 2D Ördek Puzzle		

- To achieve the goal of finding duplicate records on a scale, the candidate duplicate generation rule-set which is explained in Section 4.3 is crucial. Since the great majority of possible product pair combinations are not duplicated, calculating the features of all would be inefficient. Therefore we eliminate the pairs that are most likely not duplicated.
- We also aim to balance the data-set by eliminating the pairs that are most likely not duplicates. Otherwise, we would have an extremely unbalanced data set.
- Finally, it is a necessity to send human referees a minimal number of product pairs to be labeled for the training set since manual labeling by a human referee is a costly process.

4.5. The deduplication platform

An internal web page is designed and developed by the IT teams of the company to let the human referees view, compare, and label the detected candidate duplicate product pairs. Fig. 9 is an example of a candidate duplicate product pair that is listed on the deduplication platform.

Using the listing page, the referees are able to get the summarized information of the products. If they want to compare the pair in more detail, they simply click to the 'Karşılaştır (compare)' button. If the referee clicks the 'Karşılaştır' button, she will be able to compare the products in more detail via the product pair detail page that is illustrated in Fig. 10.

Then the referee compares the attributes of the products and if she thinks that the candidate products are actually duplicate she simply clicks one of the 'Suspend' buttons. The button in the left closes the product listed on the left side of the page and the product listed on the right-hand side would stay on sale. The button on the right, on the other hand, closes the product listed on the right-hand side of the page and the product listed on the left-hand side stays on sale. If the referee thinks that the products are not duplicates, she only needs to click on the 'Benzer Değil (not duplicate)' button then both of the products would stay on sale.

The deduplication platform logs the actions of the referees into an elasticsearch index so that the engine can collect labels of the product

pairs and creates the dataset that will be used to train classification models. Fig. 11 illustrates the end to end life cycle of the deduplication platform as follows:

- The Catalog Database stores all necessary product information (such as product name, description, image, etc.).
- The Deduplication Engine retrieves the product information from the Catalog Database and creates the Candidate Duplicate Product Records List.
- The Deduplication Engine puts the Candidate Duplicate Product Records List to FTP Server for labeling and to Similarity Metrics Database for creation/re-creation of the classification models.
- The Application Server retrieves the list from FTP Server, puts the list to Similar Products Database, and lists the products on screens for further operations such as labeling and suspending.
- The Application Server also sends the operations performed by human referees to the Service Server.
- Finally, the Service Server updates the Similar Products Database and Catalog Database according to the operations performed by human referees.

5. Text based model

In the second phase, classification models are trained and tested by candidate product pairs that are labeled at the first phase. The target value of the classification models are the labels that are produced by human referees as 'duplicate' or 'not duplicate'. The features used to build the model are explained below in detail. Using these elements binary classification models are built.

The purpose of this phase is, by training an adaptive model, to increase the precision of the candidate duplicate product records list sent to human referees. Although we eventually aim to design a system that will automatically find and close the duplicate products, in the short term the products detected by the engine have to be checked by human referees. For this reason, the target metric is chosen as precision in order to minimize manpower effort in the flow. Therefore we aim to increase the true duplicate percentage in product pairs sent to human referees as much as possible.

5.1. Features of the text based model

Let P_1 and P_2 be the product pair that is detected in first phase. We denote the number of characters in the product names of P_1 and P_2 as c_1 and c_2 , respectively, and denote the number of words in the product names of P_1 and P_2 as w_1 and w_2 , respectively. The extracted features of the text based model are as follows:

- Maximum product name length:** $\max(c_1, c_2)$
- Minimum product name length:** $\min(c_1, c_2)$
- Maximum word count:** $\max(w_1, w_2)$
- Minimum word count:** $\min(w_1, w_2)$
- Word count difference over maximum word count:** $\frac{\text{abs}(w_1 - w_2)}{\max(w_1, w_2)}$



ProductId	ProductId
HB000001EQTE	HB0000056E83
SKU	SKU
HBV000001EQTF	HBV0000056E84
Barkod	Barkod
HBV000001EQTF	0000000T13568
Ürün Adı	Ürün Adı
Miglior Gatto Av Hayvanli Kedi Konservesi 405 Gr	Miglior Gatto Av Hayvanli Kedi Konservesi 405 Gr
Resimler	Resimler
	
Varyant Özellikleri	Varyant Özellikleri
Marka	Marka
Miglior Gatto	Miglior Gatto
Ürün Tipi	Ürün Tipi
Kedi Maması	Kedi Maması
Kategori	Kategori
Yetişkin Kedi Konserveleri	Yetişkin Kedi Konserveleri
Satıcılar	Satıcılar
Satıcı : maskotpet Fiyat :18	Satıcı : LATMOS TİCARET Fiyat :17.89 Satıcı : maskotpet Fiyat :18
Yıldız	Yıldız
3.7	5
Yorum Sayısı	Yorum Sayısı
3	1
Açıklama	Açıklama
Miglior Gatto Av Hayvanli Kedi Konservesi 405 Gr MIGLIOR GATTO AV HAYVANLI KEDİ KONSERVESİ 405 GR. Kediler için eşsiz lezzeti ile dengeli bir beslenme sağlayan, içeriği yüksek kalitede ham maddeler ile oluşturulmuş sos içerisinde av hayvanı eti taneli konserve kedi mamasıdır.	Miglior Gatto Av Hayvanli Kedi Konservesi 405 Gr MIGLIOR GATTO AV HAYVANLI KEDİ KONSERVESİ 405 GR. Kediler için eşsiz lezzeti ile dengeli bir beslenme sağlayan, içeriği yüksek kalitede ham maddeler ile oluşturulmuş sos içerisinde av hayvanı eti taneli konserve kedi mamasıdır.
Ürün Özellikleri	Ürün Özellikleri
Gramaj : 405 gr Tat : Av Hayvanı Ürün Kilogram : 0 - 0,9 kg	Gramaj : 405 gr Tat : Av Hayvanı Ürün Kilogram : 0 - 0,9 kg
<input type="button" value="Suspend"/>	<input type="button" value="Benzer Değil"/>
<input type="button" value="Suspend"/>	<input type="button" value="Suspend"/>

Fig. 10. An example of a candidate duplicate product pair shown at the product pair detail page.

- **TF-IDF cosine similarity:** Cosine similarity between the TF-IDF word vectors of the names of the products P_1 and P_2 is defined as a feature.
- **Cosine similarity:** Instead of using TF-IDF weights, each word weighted equally as 1. Cosine similarity between the vectors of the names of the products P_1 and P_2 is defined as a feature.
- **Jaccard similarity:** Jaccard similarity of product names P_1 and P_2 is defined as feature.
- **Product description similarity:** Jaccard similarity of product descriptions of P_1 and P_2 is defined as a feature.
- **First three letters Jaccard similarity:** The method and the reason of producing this feature is explained in detail at Section 4.2.
- **Is first three letters Jaccard similarity greater:** The feature is generated to be able to answer the question: 'Do one of the names of P_1 or P_2 contain any typo?'
- **Jaccard numerical entity similarity:** The method and the reason of producing this feature is explained in detail at Section 4.2.
- **Jaccard textual entity similarity:** Jaccard similarity of non-numerical entities of P_1 and P_2 is defined as a feature.
- **Subset similarity:** If one of the product names of P_1 and P_2 is a subset of the other with respect to their words, then this feature is flagged as true else it is flagged as false.
- **Subset first three letters similarity:** If one of the product names of P_1 and P_2 is a subset of the other with respect to the first three

letters of their words, then this feature is flagged as true else it is flagged as false.

- **Edit distance:** Edit distance of products names is defined as a feature.
- **Product type similarity:** Jaccard similarity of product types such as 'Monts, Coats, Jackets' of P_1 and P_2 is defined as a feature.
- **Maximum occurrence count:** Let P_1 and P_2 be paired with m and n different products as a candidate, respectively. The maximum of m and n is defined as a feature. The idea of this feature is as follows: we find that some of the brands have hundreds of products with similar names, however, none of the products are duplicated. For instance, a textile brand has hundreds of products named 'pregnant blouse' however none of these blouses are duplicate. The feature is defined to get a signal from such brands.
- **Target:** The target value in the dataset is the label given by human referees for P_1 and P_2 which can be either 'duplicated' or 'not duplicated'.

5.2. Assessment of the text-based model

In this section, we will give a high-level overview of the text-based model. The text-based model outperforms the non-adaptive solution of the first phase in the following areas:

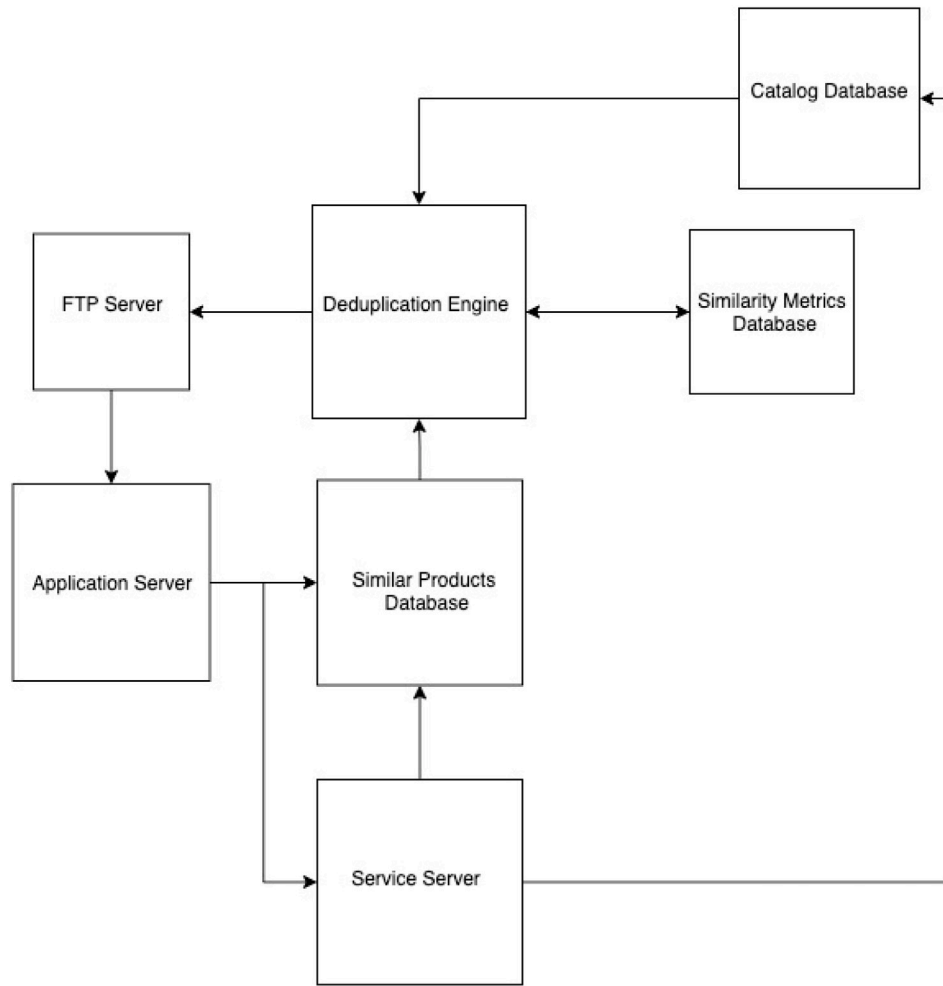


Fig. 11. End to end life cycle of the deduplication platform.

The text-based model can differentiate the similarity score of product names that contain too many words from the similarity score of product names that contain few words. For example, let P_1 and P_2 be the product pair with product names consisting of 6 words each. Let P_3 and P_4 be the product pair with product names consisting of 2 words each. Also, let P_1 and P_2 's Jaccard similarity score to be equal to the Jaccard similarity score of P_3 and P_4 . In this scenario, the low threshold based approach of the first phase would equalize the probability of being a duplicate of these pairs. The text-based model, on the other hand, discovers that the threshold can be kept lower as the word count of the product names increases. It finds that as the number of words in product names decreases, the products need to have a higher similarity score to be duplicated. This is because as the number of words in the product name increases, the pool of words that can be used to identify the product also increases. Fig. 12 is an example product pair for the explained scenario. The text-based model is able to classify the product pair illustrated at Fig. 12 as 'duplicate'.

The text-based model also can adapt to the naming differences of brands. The features named 'maximum occurrence count' and 'minimum occurrence count' give the model ability to differentiate the threshold of being duplicated according to the naming differences of the brands. Fig. 13 is an example product pair for this scenario. The brand illustrated in Fig. 13 has tens of products with identical product names: 'Pregnant Dress'. Using the features 'maximum occurrence count' and 'minimum occurrence count' the text-based model can classify the product pair as 'not duplicate'.

The text-based model, on the other hand, fails in certain scenarios. The features 'maximum occurrence count' and 'minimum occurrence

count' give the model the ability to adapt to naming differences of different brands. However, the outputs also show that the model still needs new related features to be able to detect such cases more accurately. The investigations made on the model's outputs uncovered the fact that for the best performing results, the model needs to be enriched with image similarities of the products. Another important point uncovered by the model's outputs is that some of the product pairs can be labeled by checking the price difference among them. These observations lead us to enrich the text-based model which we will explain next.

6. The enriched model

The purpose of this phase is to simplify and enrich the text-based model by studying the results and to produce new features to improve the failing scenarios. In this way, we aim to increase the model's ability to detect duplicate product pairs more precisely.

6.1. Features of the enriched model

To produce a computationally scalable final model, we conducted experiments to eliminate inefficient features of the text-based model. The features selected after the elimination of the inefficient features can be found in Table 7 of Section 7.6. In addition to the features selected from the text-based model we add the following features to enrich the model:



Fig. 12. An example product pair.



Fig. 13. An example product pair.

- **Image similarity:** Image similarity scores of the products are defined as a feature. We will explain how we calculate image similarities in Section 6.2.
- **Image filter:** The majority of the duplicated records have image similarities greater than 0.3. Therefore a new feature is defined as follows: true if the image similarity of P_1 and P_2 is greater than 0.3 otherwise it is false.
- **Jaccard filter:** The majority of the duplicated records have Jaccard similarities greater than 0.7. Therefore a new feature is defined as follows: true if Jaccard similarity of P_1 and P_2 is greater than 0.7 otherwise it is False.
- **Price over minimum price:** Let x be the 10th percentile of the prices that merchants of P_1 are selling at and y be the 10th percentile of the prices that merchants of P_2 are selling at. The feature is calculated as: $abs(x-y)/min(x, y)$. We created two more versions of this feature with the percentile set to 50th and 90th.
- **Position of the image similarity:** The feature is defined as whether the image similarity of P_1 and P_2 is greater than the 25th percentile of image similarities of other candidates under the brand. The idea behind this feature is to give an advantage to the product pairs that have a higher similarity score than other pairs of the brand. We created one more version of this feature with the percentile set to 75th.
- **Position of the Jaccard similarity:** This feature is defined using the same methodology explained in the feature 'Position of the image similarity'.
- **Position of the TF-IDF cosine similarity:** This feature is defined using the same methodology explained in the feature 'Position of the image similarity'.

6.2. Image similarity score computation for product pairs

Image similarity score computations for product pairs are conducted via a deep convolutional neural network model proposed by [Schroff, Kalenichenko, and Philbin \(2015\)](#). In their paper, the authors presented a CNN model that maps images to a 128-dimensional Euclidian space where distances between vectors correspond to a measure of dissimilarity between images. Their model incorporates a Siamese network with a triplet loss function to train on matching/non-matching image patches. These patches involve two matching images and a non-matching image where the model tries to separate the matching pair from the non-matching pair as much as possible. Unlike previous methods in the literature, the model is optimized directly on the output embedding itself rather than an intermediate bottleneck layer.

The similarity network consists of 155 hidden layers and 3,743,280 trainable weights in total. The input of the network is a $3 \times 96 \times 96$ array where the output is a 1×128 vector which represents 128-dimensional embedding of an image. $1 - FrobeniusNorm$ of the difference of two embeddings represents the similarity between related two images.

Due to the high volume of data and computational power requirements to train such a deep model, we have directly incorporated the weights trained by [Schroff et al. \(2015\)](#) into our study.

6.3. Assessments of the enriched model

The enriched model outperforms the non-adaptive solution of the first phase and the text-based model on the following areas:

Using brand-based features makes the enriched model more sensitive to naming conventions than other solutions. In this way, the



Fig. 14. An example product pair which is differentiated by image similarity.



Fig. 15. An example product pair which is differentiated by price. The left product is 174,28 Turkish Lira. The right product is 88,53 Turkish Lira.

enriched model gains the ability to classify product naming conventions of brands more precisely and to make more accurate classifications. As an example, the features 'position of the Jaccard similarity' and 'position of the TF-IDF cosine similarity' give the model the ability to position the text similarities of the product pairs based on other product pairs under the same brand. Also the feature 'position of the image similarity' gives the model the ability to position the image similarities of the product pairs based on other product pairs under the same brand. This method enables the model to learn which similarity score is normal and which similarity score is abnormal for each brand. Thus, the model is able to distinguish brands with product names that are very similar to each other from brands that generally have distant product names.

The use of image similarity scores gives the model the ability to classify product pairs that are very similar in textual terms, but visually different as 'not duplicate'. Fig. 14 is an example product pair of the explained scenario. The enriched model is able to classify the product pair illustrated in Fig. 14 as 'not duplicate' even though the product names are almost identical.

The use of price differences gives the model the ability to label the product pairs that are very similar both textually and visually as 'not duplicate'. Fig. 15 is an example product pair of the explained scenario. The enriched model is able to classify the product pair illustrated in Fig. 15 as 'not duplicate' even though the product names are identical and product images are similar.

7. Experimental results

7.1. An overview of the dataset

The dataset set contains 34007 pairs which consist of 12324 duplicate and 21683 non-duplicate pairs. Table 2 shows the proportional distribution of the categories of product pairs in the overall

Table 2

Product categories and rates in dataset.

Categories	Rates
Mother - baby	0.24
Auto accessory	0.19
Construction market	0.18
Supermarket	0.11
Petshop	0.07
Toys	0.05
Cosmetic	0.05
Stationery/office	0.04
Home decoration	0.03
Sport	0.02
Home textiles	0.02

dataset. Category selection and prioritization are made by the business units of Hepsiburada. As a result of the choices made according to the priorities of the business units, the distribution took place as in Table 2.

Table 3 shows distributions of basic similarity features of the dataset according to their mean, standard deviation, minimum value, and maximum value. Indicators of the table are calculated using all product pairs of the dataset. The Highness of the mean indicators of Jaccard similarity, TF-IDF cosine similarity, cosine similarity, product description similarity, and edit distance reveals the fact that our blocking method is performing well on the matter of dividing products into sub-sets that contains similar products.

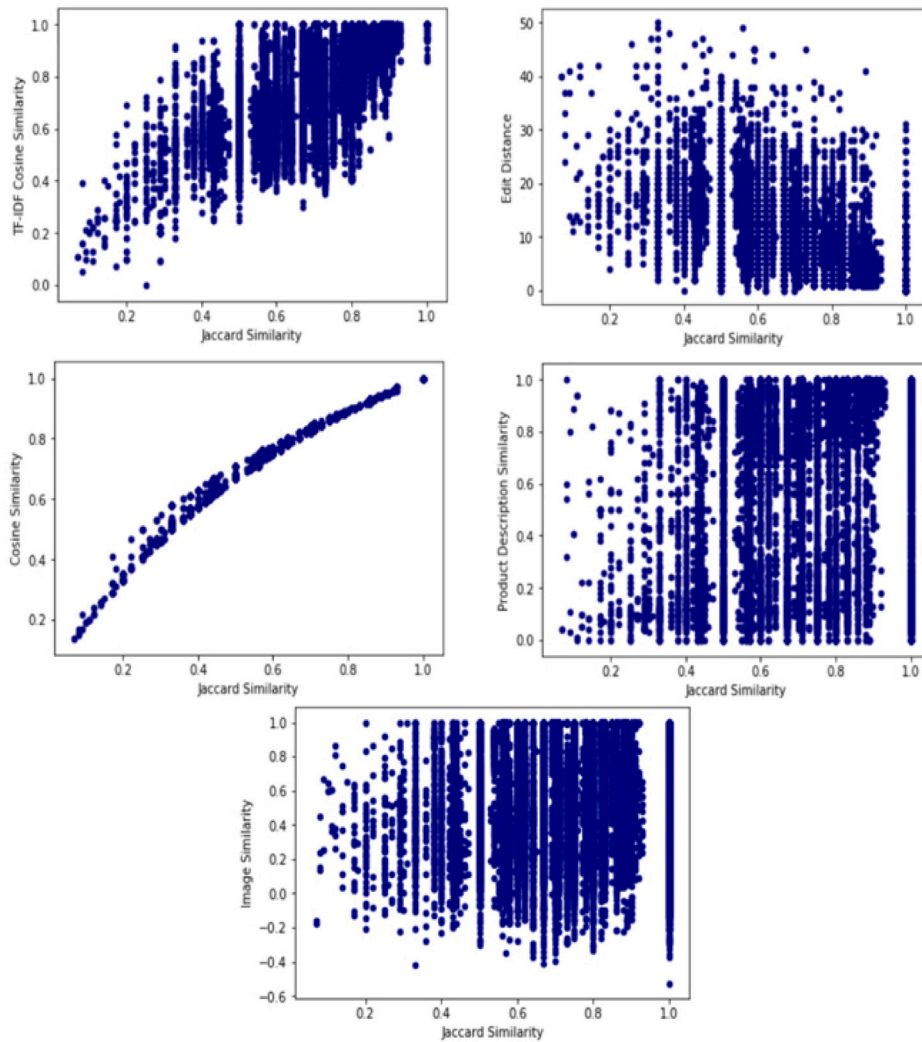


Fig. 16. Scatter plot comparisons of basic features based on Jaccard similarity.

Table 3

Distributions of basic similarity features.

Feature name	Mean	Std	Min	Max
Jaccard sim.	0.80	0.18	0.30	1.00
TF-IDF cosine sim.	0.85	0.18	0.37	1.00
Cosine sim.	0.88	0.12	0.45	1.00
Image sim.	0.58	0.31	-0.52	1.00
Jaccard textual entity sim.	0.89	0.30	0.20	1.00
Jaccard numerical entity sim.	0.44	0.49	0.00	1.00
Product description sim.	0.75	0.30	0.00	1.00
Edit distance	5.38	6.69	0.00	50.00

7.2. Scatter plot comparisons of basic features based on jaccard similarity

Fig. 16 illustrates a scatter plot comparison of Jaccard similarity and TF-IDF cosine similarity, cosine similarity, image similarity, product description similarity, edit distance of the product pairs.

The prominent correlations among Jaccard similarity and TF-IDF cosine similarity, Jaccard similarity, and cosine similarity are expected due to their token-based natures. Since both Jaccard, TF-IDF cosine, and cosine similarity methods calculate the similarities based on the word tokens of the product names, they naturally correlate with each other. On the other hand, since edit distance is a character-based and Jaccard similarity is a token-based techniques, they are naturally not correlated.

The relations among Jaccard similarity and image similarity, Jaccard similarity and product description similarity, however, point out some important information about the product catalog. First, in an ideal world, Jaccard similarity and product description similarity would be expected to be correlated. It would be ideal for two products with similar product names to have similar product descriptions. However, the scatter plot comparison of these two similarity metrics shows that these two metrics are not correlated with each other. This implies that different vendors can describe a product with different texts even though they give the same name to the product. Second, the relation between Jaccard similarity and image similarity indicates that products that have similar product names might have dissimilar images. It also indicates that products that have similar images might have dissimilar names.

7.3. The results obtained from the first phase - The rule-based solution

The first phase is to identify candidate pairs with the rule-based solution and a low threshold in order to create a labeled dataset. Since the rule-based solution recognizes product pairs that have a similarity score under the given threshold as not candidate pairs, human referees are not able to label such product pairs. For this reason, we do not have the labels of the product pairs that the engine implemented in the first phase does not recognize as a candidate. We only have labels of the products identified as candidate duplicate products from the first

Table 4

Performance metrics of the text-based model.

Classification model	Accuracy	F1 score	Precision	Recall
Random forest	0.833	0.814	0.798	0.807
SVM	0.795	0.768	0.757	0.759
K-NN	0.801	0.779	0.740	0.775
Decision tree	0.792	0.767	0.733	0.762
Logistic regression	0.760	0.740	0.657	0.743
AdaBoost	0.774	0.749	0.699	0.745

Table 5

Performance metrics of the text-based model.

Feature name	Importance
Maximum occurrence count	0.346
TF-IDF cosine similarity	0.127
Product description similarity	0.118
Edit distance	0.067
Jaccard numerical entity similarity	0.054
Maximum word count	0.037
Maximum product name length	0.035
Minimum product name length	0.033
Minimum word count	0.031
First three letter Jaccard similarity	0.028
Jaccard similarity	0.027
Cosine similarity	0.027
Word count difference over max. word count	0.023
Subset first three letter similarity	0.017
Subset similarity	0.011
Product type similarity	0.009
Jaccard textual entity similarity	0.005
Is first three letter Jaccard similarity greater	0.005

phase. Therefore, metrics such as accuracy, F1 score, and recall cannot be calculated for the results obtained from the first phase.

As stated in Section 5, the precision score is chosen as the target metric to reduce the workload of human referees. When the products labeled by human referees are examined, the precision score was calculated as **0.363** in the first phase, that is, 36% of the candidate product pairs that the engine says 'might be duplicate' is actually duplicate.

7.4. The results obtained from the second phase - The text-based model

Table 4 shows the results from the text-based model, using various classifiers. Results are compared based on accuracy, f1 score, precision, and recall metrics. All the classification results that are given below and the following sections are based on a 10-fold cross-validation experiment.

The results show that the random forest classifier outperforms various other classifiers and gives the best results. Compared to the rule-based solution, the text-based model has achieved a much higher precision score. This result confirms our argument that using adaptive methods to detect duplicate product records would yield better results than non-adaptive methods.

Table 5 shows impurity-based importance scores obtained from the random forest classifier model. The results show that 'Maximum Occurrence Count', 'TF-IDF Cosine Similarity', 'Product Description Similarity' are the three most important features.

7.5. The results obtained from the third phase - The enriched model

Table 6 shows obtained results from the enriched model, using various classifiers. The results showed that the random forest classifier outperforms various other classifiers and gives the best results.

Table 7 shows obtained impurity-based feature importance from the random forest classifier model. The results show that 'Image Similarity' dominates other features.

Table 8 shows precision scores of the enriched model using all features and top 5 highest importance features. The results show that for

Table 6

Performance metrics of the enriched model.

Classification model	Accuracy	F1 score	Precision	Recall
Random forest	0.860	0.842	0.853	0.832
SVM	0.828	0.804	0.810	0.794
Decision tree	0.823	0.799	0.800	0.789
AdaBoost	0.821	0.799	0.780	0.792
K-NN	0.813	0.791	0.771	0.783
Logistic regression	0.790	0.767	0.721	0.764

Table 7

Performance metrics of the enriched model.

Feature name	Importance
Image similarity	0.372
Price over minimum price	0.175
Product description similarity	0.085
Jaccard numerical entity similarity	0.082
TF-IDF cosine similarity	0.079
Jaccard similarity	0.049
Image filter	0.038
Position of the image similarity	0.033
Word count difference over max. word count	0.029
Position of the TF-IDF cosine similarity	0.027
Position of the Jaccard similarity	0.014
Product type similarity	0.009
Jaccard filter	0.008
Is first three letter jaccard similarity greater	0.002

Table 8

Precision score comparison of all features and top 5 features.

Classification model	All features	Top 5 features
Random forest	0.853	0.831
SVM	0.810	0.754
Decision tree	0.800	0.783
Adaboost	0.780	0.786
K-NN	0.771	0.753
Logistic regression	0.721	0.647

Table 9

Accuracy score comparisons of the Text-based model and the enriched model.

Classifier	Text-based model		Enriched model	
	Accuracy	Std. dev	Accuracy	Std. dev
Random forest	0.833	0.0043	0.860*	0.0045
SVM	0.795	0.0114	0.828	0.0121
K-NN	0.801	0.0050	0.813	0.0105
Decision tree	0.792	0.0136	0.823	0.0064
Logistic regression	0.760	0.0070	0.790	0.0063
AdaBoost	0.774	0.0080	0.821	0.0073

all classifiers except AdaBoost, seemingly insignificant features make a significant difference on the basis of precision score, which is the target metric of the engine. For all classifiers except AdaBoost classifier, p-values are calculated as less than 0.01. The results for the Adaboost classifier are insignificant as the precision scores for all features and for top 5 features are very close.

7.6. Result comparisons

Tables 9, 10, 11, 12 show the accuracy, F1, precision, recall scores of the text-based model and the enriched model respectively. To calculate these scores and standard deviations we applied 10-fold cross-validation. For all results, p-values are calculated as less than 0.05. Since accuracy, F1 and recall scores could not be calculated for the rule-based solution, the comparisons are made between the text-based model and the enriched model. As we discussed in Section 7.3 the precision score of the rule-based solution is calculated as **0.363**. Results show that the enriched model outperforms the text-based model.

Table 10

F1 score comparisons of the text based model and the enriched model.

Classifier	Text-based model		Enriched model	
	F1 score	Std. dev	F1 score	Std. dev
Random forest	0.814	0.0043	0.842*	0.0045
SVM	0.768	0.0114	0.804	0.0121
K-NN	0.779	0.0050	0.791	0.0105
Decision tree	0.767	0.0136	0.799	0.0064
Logistic regression	0.740	0.0070	0.767	0.0063
AdaBoost	0.749	0.0080	0.799	0.0073

Table 11

Precision score comparisons of the text based model and the enriched model.

Classifier	Text-based model		Enriched model	
	Precision	Std. dev	Precision	Std. dev
Random forest	0.798	0.0130	0.853*	0.0103
SVM	0.757	0.0132	0.810	0.0170
K-NN	0.740	0.0174	0.771	0.0077
Decision tree	0.733	0.0232	0.800	0.0105
Logistic regression	0.657	0.0151	0.721	0.0128
AdaBoost	0.699	0.0171	0.780	0.0124

Table 12

Recall score comparisons of the text based model and the enriched model.

Classifier	Text-based model		Enriched model	
	Recall	Std. dev	Recall	Std. dev
Random forest	0.807	0.004	0.832*	0.005
SVM	0.759	0.010	0.794	0.012
K-NN	0.775	0.004	0.783	0.010
Decision tree	0.762	0.016	0.789	0.007
Logistic regression	0.743	0.006	0.764	0.009
AdaBoost	0.745	0.008	0.792	0.008

Table 13

Aggregated importance of text-based model's features.

Feature group	Importance
Product name features	0.527
Brand features	0.346
Product description similarity	0.118
Product type similarity	0.009

Table 13 groups features of the text-based model under four main titles and shows aggregated feature importance scores of the main titles. Features are grouped under the following titles:

- **Product Name Features:** The features derived from product name similarities.
- **Brand Features:** The features derived from brand behaviors.
- **Product Description Features:** The features derived from product description similarities.
- **Product Type Features:** The features derived from product type similarities.

Table 14 groups features of the text-based model under six main titles and shows aggregated feature importance scores of the main titles. Features are grouped under the following titles:

- **Image Features:** The features derived from image similarities.
- **Price Features:** The features derived from image similarities.
- **Product Name Features:** The features derived from product name similarities.
- **Brand Features:** The features derived from brand behaviors.
- **Product Description Features:** The features derived from product description similarities.
- **Product Type Features:** The features derived from product type similarities.

Table 14

Aggregated importance of enriched model's features.

Feature group	Importance
Image features	0.410
Price features	0.175
Product name features	0.218
Brand features	0.103
Product description features	0.085
Product type features	0.009

Results show that, since the engine determines the candidate duplicate product pairs by applying filters to the text-similarity scores of the product pairs, in the training set image features have become more distinctive compared to others. If the training set were created by filtering product pairs via image similarity scores then the text-similarity features would become more distinctive than the image features. Therefore, when evaluating feature importance scores, it should not be overlooked that these scores are dependent on the rule sets used while creating the training set.

8. Conclusion and future work

This work presented the development of a duplicate record detection engine for the product catalog of Hepsiburada. The engine is implemented in three phases.

In the first phase, similar product pairs are detected using traditional text similarity methods. Product pairs that are above a certain threshold are recognized as candidate duplicate product pairs. The candidate duplicate product list generated in this way is sent to human references to be labeled as 'duplicate' or 'not duplicate'.

In the second phase, classification models are trained using labeled product pairs. The text similarity scores produced in the first phase are used as the features of the model. The labels produced by the referees are used as targets. The experimental results show that the text-based model established in the second phase drastically increases the precision score of the classification performance compared to the rule-based solution.

In the third phase, the text-based model is enriched using product images and prices. In the experiments, it is observed that the enriched model improved the performance of the text-based model.

The engine developed for Hepsiburada is put into use within the company. Thousands of duplicate product records detected by the engine are processed by human referees and removed from the catalog. The engine runs daily and scans the categories requested by business units to detect duplicate product records in those categories. On the other hand, new products are entered into the product catalog every day by vendors. While the catalog is cleaned by the duplicate product record detection engine, it is also contaminated with new duplicate records uploaded by vendors.

As future work, we aim to perform duplicate product record detection in real-time during product entries made by the vendors. Thus, the product catalog will be kept clean at all times.

CRediT authorship contribution statement

Osman Semih Albayrak: Conceptualization, Methodology, Software, Validation, Writing – original draft, Formal Analysis, Visualization, Investigation. **Tevfik Aytekin:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Tolga Ahmet Kalaycı:** Project Administration, Software, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully thank the development team of Hepsiburada for keeping this project alive by implementing user-friendly deduplication screens. Also special thanks to the catalog teams of Hepsiburada for sharing their knowledge which helped us to understand the nature of the problem better.

This research is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under project number 3180216.

References

- Arasu, A., Chaudhuri, S., & Kaushik, R. (2008). Transformation-based framework for record matching. In *2008 IEEE 24th international conference on data engineering* (pp. 40–49). IEEE.
- Bilenko, M., & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 39–48).
- de Carvalho, M. G., Gonçalves, M. A., Laender, A. H., & da Silva, A. S. (2006). Learning to deduplicate. In *Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries* (pp. 41–50).
- Cohen, W. W., Ravikumar, P., & Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *IIWeb, Vol. 2003* (pp. 73–78).
- Cohen, W. W., & Richman, J. (2002). Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 475–480).
- Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2006). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1), 1–16.
- Fisher, J., Christen, P., & Wang, Q. (2016). Active learning based entity resolution using Markov logic. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 338–349). Springer.
- Ghani, R., Probst, K., Liu, Y., Krema, M., & Fano, A. (2006). Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1), 41–48.
- Hadi Kiapour, M., Han, X., Lazebnik, S., Berg, A. C., & Berg, T. L. (2015). Where to buy it: Matching street clothing photos in online shops. In *The IEEE international conference on computer vision*.
- Hassanzadeh, O., & Consens, M. P. (2009). Linked movie data base. In *LDOW*.
- Hoad, T. C., & Zobel, J. (2003). Methods for identifying versioned and plagiarized documents. *Journal of the American Society for Information Science and Technology*, 54(3), 203–215.
- Köpcke, H., Thor, A., & Rahm, E. (2010). Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1–2), 484–493.
- Koudas, N., Marathe, A., & Srivastava, D. (2004). Flexible string matching against large databases in practice. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30* (pp. 1078–1086).
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8), 707–710.
- Li, J., Dou, Z., Zhu, Y., Zuo, X., & Wen, J.-R. (2020). Deep cross-platform product matching in e-commerce. *Information Retrieval Journal*, 23(2), 136–158.
- Lin, Y.-S., Liao, T.-Y., & Lee, S.-J. (2013). Detecting near-duplicate documents using sentence-level features and supervised learning. *Expert Systems with Applications*, 40(5), 1467–1476.
- McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 169–178).
- Monge, A., & Elkan, C. (1997). An efficient domain-independent algorithm for detecting approximately duplicate database records.
- OECD (2019). Unlocking the potential of e-commerce. OECD Going Digital Policy Note, Paris. <http://www.oecd.org/going-digital/unlocking-the-potential-of-e-commerce.pdf>.
- O'Hare, K., Jurek-Loughrey, A., & de Campos, C. (2019). An unsupervised blocking technique for more efficient record linkage. *Data & Knowledge Engineering*, 122, 181–195.
- Papadakis, G., Skoutas, D., Thanos, E., & Palpanas, T. (2020). Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys*, 53(2), 1–42.
- Raeesi, M., Asadpour, M., & Shakery, A. (2020). Swash: A collective personal name matching framework. *Expert Systems with Applications*, 147, Article 113115.
- Ristoski, P., Petrovski, P., Mika, P., & Paulheim, H. (2018). A machine learning approach for product matching and categorization. *Semantic Web*, 9(5), 707–728.
- Roostaei, M., Fakhrahmad, S. M., & Sadreddini, M. H. (2020). Cross-language text alignment: A proposed two-level matching scheme for plagiarism detection. *Expert Systems with Applications*, 160, Article 113718.
- Sarawagi, S., & Bhamidipaty, A. (2002). Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 269–278).
- Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 815–823).
- Tata, S., & Patel, J. M. (2007). Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record*, 36(2), 7–12.
- Thor, A., & Rahm, E. (2007). MOMA-a mapping-based object matching system. Citeseer.
- Winkler, W. E. (1999). *The state of record linkage and current research problems*. Statistical Research Division, US Census Bureau, Citeseer.
- Xiao, C., Wang, W., Lin, X., Yu, J. X., & Wang, G. (2011). Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems*, 36(3), 1–41.