MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

DUPLICATE PRODUCT RECORD DETECTION ENGINE FOR E-COMMERCE PLATFORMS

OSMAN SEMIH ALBAYRAK

MASTER THESIS
Program of Data Engineering

Thesis Supervisor
Assoc. Prof. Tevfik AYTEKIN

ISTANBUL, 2021

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

DUPLICATE PRODUCT RECORD DETECTION ENGINE FOR E-COMMERCE PLATFORMS

OSMAN SEMIH ALBAYRAK

(527619019)

MASTER THESIS
Program of Data Engineering

Thesis Supervisor
Assoc. Prof. Tevfik AYTEKIN

ISTANBUL, 2021

ACKNOWLEDGEMENTS

First, I owe my deepest gratitude to my advisor Assoc. Prof. Tevfik Aytekin for his continuous support, his patience and wisdom. Whenever I ran into a trouble spot or had a question, he supported me in such ways that I could not dare to hope.

I wish to express my sincere thanks to Prof. Dr. Haluk Rahmi Topçuoglu for his thoughtful guidance on these strange days. I am able to complete this thesis thanks to him.

Finally, special thanks to my colleagues Tolga and Fırat, for the times we spent together on this work while seeking ways of improvement and to my company, Hepsiburada.com for always supporting me to push the limits of my knowledge.

Jan, 2021 **Osman Semih ALBAYRAK**

TABLE OF CONTENTS

ÖZET

E-TİCARET PLATFORMLARI İÇİN MÜKERRER ÜRÜN KAYDI TESPİT MOTORU

Temiz ve sektör standartlarını karşılayabilen bir ürün kataloğuna sahip olabilmek ve eldeki ürün kataloğunu sektör standartlarının altına düşmeden yaşatabilmek e-ticaret şirketlerinin temel uğraşlarından biridir. Binlerce ürün sağlayıcısı tarafından sisteme girilen yeni ürün bilgileri şirketleri zorlu bir problemle karşı karşıya bırakır: Mükerrer ürün kayıtları. Herhangi bir ürünü birbirinden farklı kelimelerle, farklı resimlerle ve bileşenlerle tanımlamak mümkün olduğundan, mükerrer ürün kayıtlarını tespit edebilmek üstesinden gelmesi zor bir görevdir. Bu çalışmada, bir e-ticaret firması olan Hepsiburada.com için özgün bir mükerrer ürün kaydı tespit motoru önerilmiştir. Motor, Hepsiburada.com'un gerçek verileri temel alınarak geliştirilmiştir. Ham veriden, eğitilebilir bir veri seti oluşturabilmek için çeşitli metin benzerliği algoritmaları, e-ticarete özel kurallandırılmış metin benzerliği metrikleri ve görsel benzerlik metrikleri kullanılmıştır. Metin benzerliği hesaplamaları için Jaccar benzerliği, TF-IDF kosinüs benzerliği ve edit uzaklığı gibi geleneksel metin benzerliği yöntemlerine başvurulmuştur. Görsel benzerlik hesaplamaları için bir Siyam (İkiz) Sinir Ağı eğitilmiştir. Herhangi iki ürünün mükerrer olup olmadığını tespit edebilmek için oluşturulan veri seti kullanılarak iki sınıflı sınıflandırma modelleri eğitilmiştir. Deneysel sonuçlar, önerilen motorun, Hepsiburada.com içerisindeki mükerrer ürün kayıtlarını geleneksel yöntemlerden daha başarılı şekilde tespit edebildiğini göstermiştir.

ABSTRACT

DUPLICATE PRODUCT RECORD DETECTION ENGINE FOR E-COMMERCE PLATFORMS

Having a clean product catalog and keeping it complying with the standards of the industry is one of the essential concerns of e-commerce companies. Integrating the 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 with. In this thesis, a novel duplicate record detection engine is proposed for an e-commerce company, Hepsiburada.com. The engine is developed based on a real-world data set. A number of different text similarity algorithms, domain-specific distance metrics, image similarity metrics are used to form a training data set. Traditional text similarity algorithms such as Jaccard similarity, TF-IDF cosine similarity and edit distance are used for text similarity calculations. A Siamese (Twin) Neural Network is trained and used for image similarity calculations. Two-class classification models are trained using the data set created to determine whether any two products are duplicated or not. The experimental results show that our engine is able to use product information for duplicate record detection and outperforms the accuracy of non-adaptive methodologies.

ABREVIATIONS

ICT : Information, Communication Technology Sector
IDF : Inverse Document Frequency
OECD : The Organization for Economic Co-operation and Development
TF-IDF : Term Frequency - Inverse Document Frequency
TF : Term Frequency

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Current Situation

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 does an e-commerce company has in its catalog, keeping the data clean provides better operations, better analytics, better marketing, 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 in the trade industry today. Figure 1.1 visualizes the results of a recent OECD report [1] which suggests that more than half of all individuals in OECD countries have made online purchases in 2018.

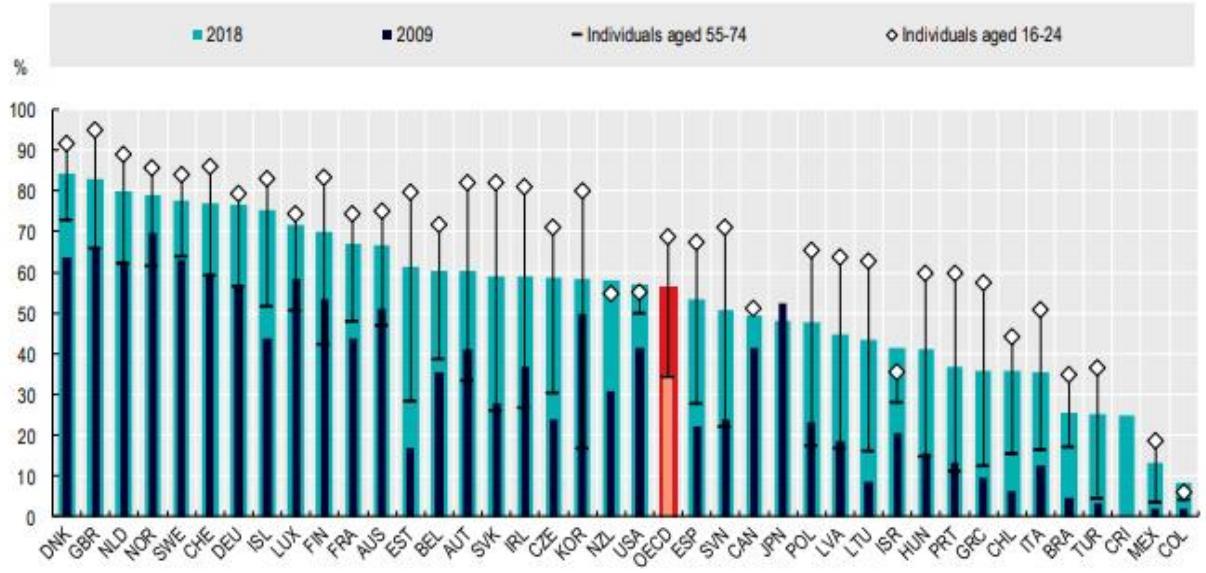

Figure 1.1: Percentage of individuals who have purchased online in 2018. Source: "Unlocking the potential of e-commerce", OECD Going Digital Policy Note, OECD, Paris.

When it comes to e-commerce, market growth often means catalog growth and catalog growth means new product entries to the catalogs which leads companies to more complex catalog structures. As an example, it can be seen from Figure 1.2, the product catalog of Hepsiburada.com had a splendid catalog growth from the year 2015 to 2020. Every single year, the catalog grew up with millions of products. During 2019, Hepsiburada.com expanded its catalog with nearly 8 million new products. The numbers of the year 2020 clearly explains the impact of the pandemic on e-commerce.

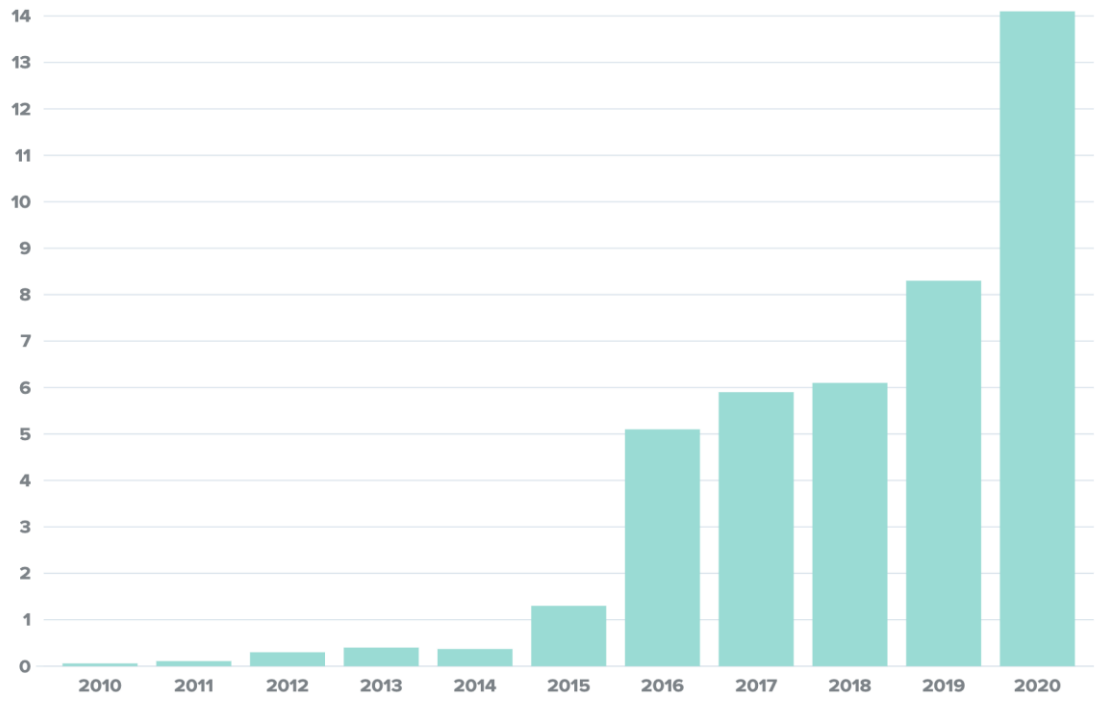

Figure 1.2: Annual product entry counts of Hepsiburada.com in terms of millions.

Such catalog growths, of course, comes with their own difficulties. Even sub-sectors were born to overcome the management difficulties of colossal catalogs. To manage the growths 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 and thousands of new products may have uploaded to the catalog from providers in a single day. Since there is not an automated content control mechanism, and different providers

may upload same products with different at- tributes (name, image, description, etc.) to the catalog, 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. Product record is a set of attributes that represent the product such as product name, product description, product brand, etc. A duplicate product record means, for some reason, two or more identical products exist in the catalog. An example, where the duplicates have exactly the same product name and image is given in Figure 1.3.

Figure 1.3: An example of a duplicate product record with identical text and image.

On the other hand, a duplicate product pair may not have identical product names or images. Figure 1.4 illustrates a duplicate product pair with similar product names. Textually, product names are not identical, however, they are pointing out to the exact same product.

Figure 1.4: An example of a duplicate product record with similar text.

Figure 1.5 illustrates another scenario of duplicate product records. Two products in Figure 1.5 have similar but not identical images. However, they are again referring to the exact same product.

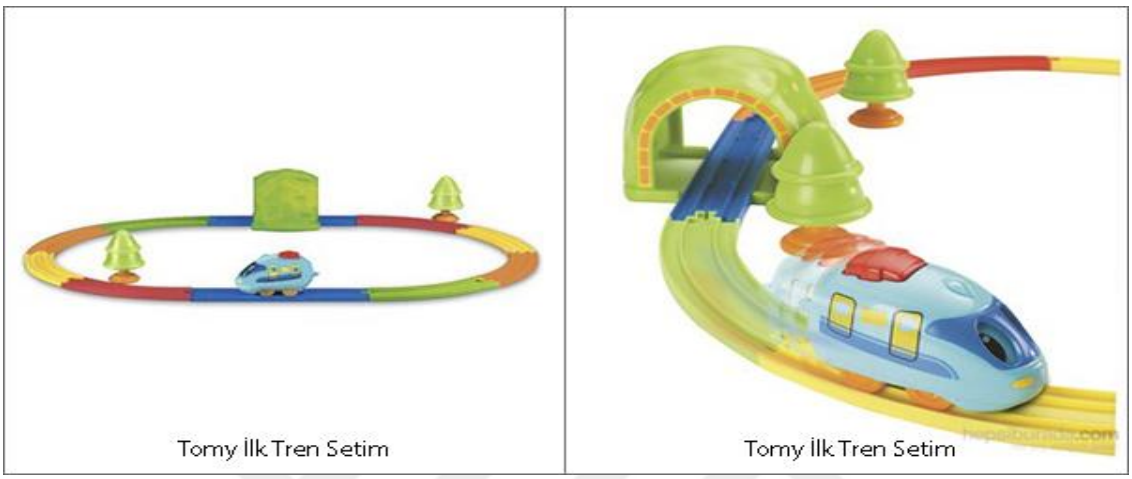

Figure 1.5: An example of a duplicate product record with similar image.

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 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. Because when they open a new record for the product, they become the only seller of that product.

Since Hepsiburada.com users are accustomed to finding all the information about the product they are looking for on a single page. They are accustomed to finding 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 on these pages mean being deprived of showing another product to the customer. Duplicate product records also negatively affect the market performance measurements and stock management of the products. For the reasons mentioned above, it is crucial to study the problem of identifying duplicate products.

1.2 Contribution

An important amount of previous work has been made on the field of duplicate product record detection. Although in this thesis, it is preferred to entitle the issue as duplicate product record detection, several other terms such as duplicate detection [2, 3], product matching [4], reference matching [5], name matching [6, 7], entity resolution [8], detecting plagiarism [9, 10], record linkage[11], and near-duplicate detection [12] are used in previous researches.

Most of the previous studies are basically text-based approaches where traditional string similarity metrics such as TFIDF [6] and Jaccard Similarity [12] are used. Since a product is typically defined with several text attributes and visual attributes such as product name, product description, the brand of the product, price of the product and image; in this work, to accomplish the goal of detecting duplicate products a hybrid methodology of standard traditional similarity metrics and image similarity metrics are used.

Also, when the product catalog is investigated it is discovered that 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 exactly the same name to hundreds of different products: ‘Pregnant Woman's Blouse’. Although the names of these products are exactly 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 naming differences of the brands, in this work traditional string similarity metrics are combined in adaptive learning methodologies [13, 14].

This thesis proposes a three-phased methodology is used to identify duplicate product records. At the outset, the goal of the first phase is to create a trainable golden dataset. Traditional string similarity algorithms are used to calculate text similarities of product pairs. Pairs that are over a relatively low threshold accepted as candidate duplicate product pairs. Candidate pairs are sent to human referees to be labeled as ‘duplicate’ or ‘not duplicate’.

Second, using the labeled product pairs, a variety of classification models are trained and tested to determine whether given two products are matched and identical. The target values of the models are the labels that are produced by the human referees. The features of the model, on the other hand, are the produced similarity scores using traditional string similarity algorithms and brand-specific features.

Third, we aim to enrich the text-based model by studying the results and producing new features according to failure scenarios. New features based on image similarity scores, price differences and brand-based differences of the product pairs are generated and added to the model. The experimental results showed that our enriched model further improves the text-based model.

2 RELATED WORK

2.1 String Similarity Methods

Traditional string similarity 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 [6, 15]. 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 [6, 16]. 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' to the size of the union of the sets A' and B'.

$$Jaccard(A', B') = \frac{A' \cap B'}{A' \cup B'} \qquad 2.1$$

In this work, the sets A and B consist of tokenized word lists of the given product names P_1 and P_2. If, for example, P_1 is 'iphone 6 64 GB' and P_2 is 'iphone 7 32 GB', when tokenization is performed, the set A' will be equal to ['iphone', '6', '64', 'GB'] and the set B' will be equal to ['iphone', '7', '32', 'GB']. In that case, the Jaccard Similarity of the sets A' and B' will be $\frac{2}{6}$. To detect duplicated products Jaccard similarities of the product pairs can be used as one of the primary similarity scores.

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 given two documents into vectors and applying a similarity measure (such as cosine) on these vectors [17–19]. The most common way to do this is to use a bag of words approach with TF-IDF. 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:

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D) \qquad 2.2$$

Where the t stands for a term, d stands for a document, and D stands for the collection of documents. Although there are variations the exact calculation that we use in this work 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 this work, D stands for the collection of product names which belong to the same brand. As an example, products that belong to the brand Apple comprises a collection. d stands for each product name (or document), and t stands for each word in a product name. TF-IDF

transformation of a product name creates an n-dimensional vector for each product name where n stands for the total number of distinct words in D. After each product name is vectorized by TF-IDF we use cosine similarity to calculate the similarity scores.

In our experiments, the TF-IDF cosine similarity approach gives the best performance among text similarity algorithms for duplicate product record detection problem. Therefore it is used as the primary text similarity metric. [6] also confirms that TF-IDF cosine similarity 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 [20] compares several entity matching methodologies and proposes that the learning-based match strategies outperform the non-learning approaches. [13] also states that rather than tuning traditional string similarity metrics, adaptive approaches have clear advantages to identify duplicate records. Their work shows that adaptive methods are capable to learn domain-specific naming conventions. [21–23] 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 [24].

Similar to our problem, [25] 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 treat the problem as a learning problem. [26] uses image similarities with text similarities to match products from different e-commerce websites. [27] on the other hand, studies only the image similarities to match products of the apparel domain.

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 [28].

As stated by [29] blocking rules can be developed manually or can be found heuristically or can be learned automatically, yet each technique has its own 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 the expertise of our domain, we walk with the idea of dividing the dataset into small pieces by a disjunctive variable. The details of the blocking method we use will be explained in Chapter 4.

3 PRELIMINARIES

In this chapter, 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 Figure 3.1, approximately 14 millions of these products are currently on sale.

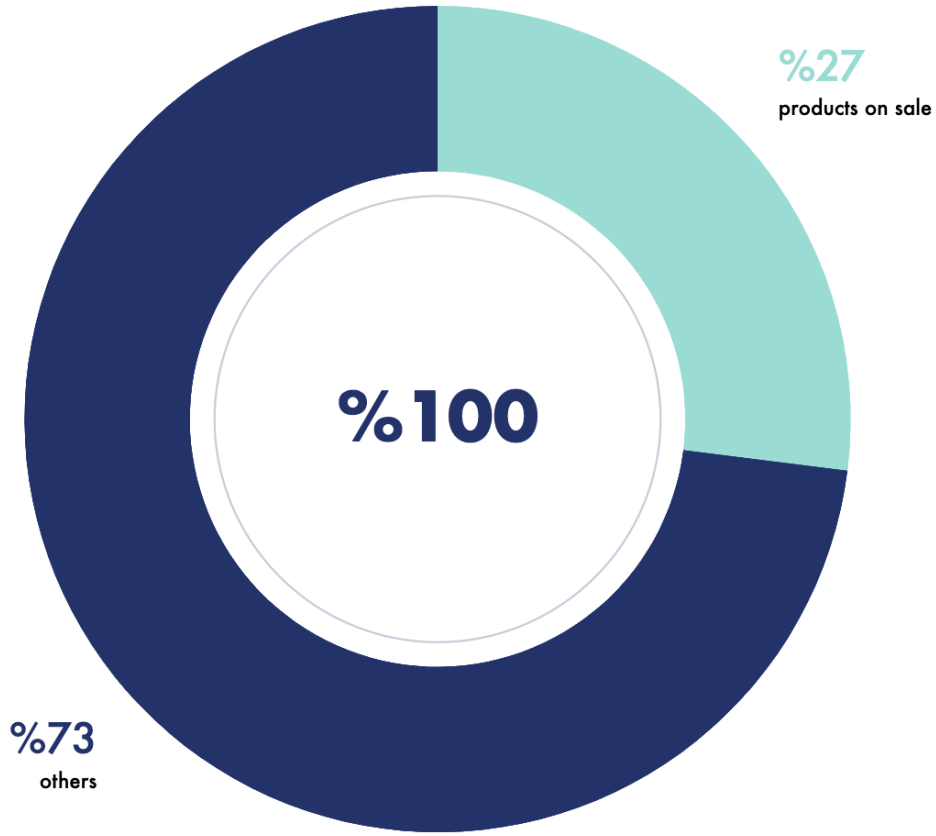

Figure 3.1: The ratio of products that are on sale.

The distribution of products on sale by categories is shown in Figure 3.2. There are 30 main categories in the catalog. As can be seen from Figure 3.2, 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.

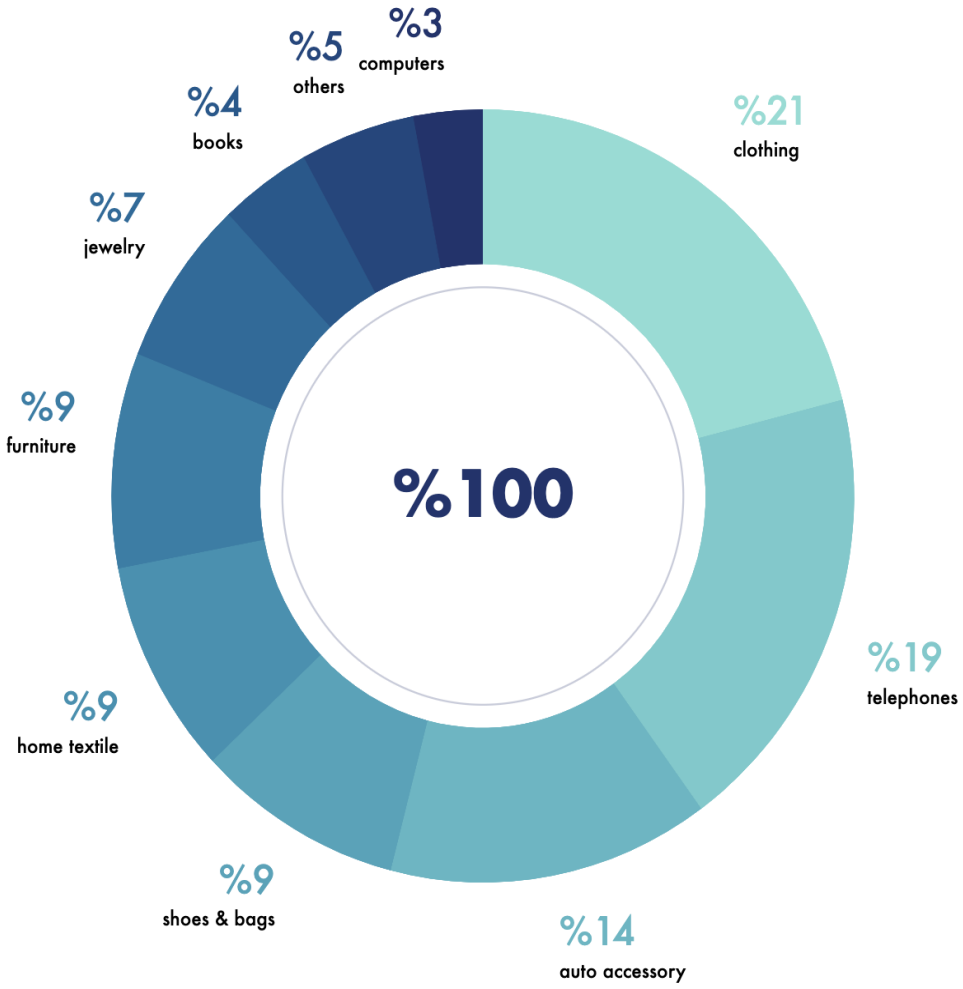

Figure 3.2: The distribution of the products by categories.

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.

Preliminary studies in the catalog show that, 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 records. 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.

As detailed in Chapter 4, it has been determined that searching for identical products in a cluster of 14 million products will not be an economical solution method under the conditions explained above. Instead, it is less costly to compare the products whose brands and the main categories are identical.

In addition, preliminary research has shown that vendors tend to create duplicates of bestsellers or most viewed products more than other products. This tendency of the vendors raises the question `Which of the two identical products should be closed and which should remain on sale?'. To prevent the closure of bestsellers or most viewed products, we aim to collect such products under the definition of `Hero Product'. For this purpose, we decide to develop the hero product detection algorithm. The details will be discussed in Chapter 4.

Finally, the preliminary results have shown that the same product registrations can have different names, different images, and different prices. For this reason, it is not possible to identify such products with rule-based approaches, instead, adaptive learning-based approaches is needed. Due to the nature of adaptive learning-based approaches, it is necessary to create a training dataset that is suitable for the problem.

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 Chapter 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 Chapter 5.

The third and the 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 Chapter 6.

4 CONSTRUCTION OF THE TRAINING SET

This chapter will summarize the methods of finding product pairs that are candidates 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 ‘1cm’ or ‘1kg’ 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’].
- Contrary to the classical text mining studies, when the product catalog is investigated, it is discovered that the 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 Figure 4.1. Two products are shown in Figure 4.1 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.

Figure 4.1: An example of stopwords. The word 've' means 'and' in Turkish.

- We discover that 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 Chapter 3, two products can be duplicate if they belong to the same brand. Therefore, the duplicate detection engine compares any product of Apple only with another product of Apple. Since product names of the 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.

4.2 Blocking

In order to decrease the number of pairwise comparisons (which is quadratic with respect to the number of products), we divide the data set into meaningful chunks. During the explanatory data analysis, it is detected that the company is keeping 43 millions individual products in its catalog. Since searching for duplicated products in such a large dataset 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 data set into 14 million individual products.

However, the problem of trillions of comparisons is still in the phase. In further investigations, a catalog-specific rule is uncovered. As explained in Chapter 3, we find that if two products are duplicated, they are almost always taking part under the same brand and the same main category. 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.

4.3 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, the hero product detection algorithm is developed. The definition of hero product is as follows: products with high metrics such as sales, views, number of unique customers are defined to be hero products. The steps of the hero product detection algorithm are as follows:

- 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.

4.4 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**: When the product catalog is investigated, it is uncovered that for identical products the following rule can be defined: 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 identical.

- **Category word difference**: A word list is produced from the hierarchy names of the product catalog and named as a category word list. The following rule is developed: If two products are identical 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 identical. 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 identical. 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:** When the product catalog is investigated, it is found that if two products are identical then the following rule can be defined: 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 identical 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’. Product name

of product B should have the same numerical entities with product A in order for them to be identical. On the other hand, if 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.

- **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.

4.5 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 category 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 identical. 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 identical. 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 identical. 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.

(1) If the similarity score is smaller than 0.3 then the anchor product and the candidate product are not identical. Skip to the next candidate product.

(2) If the similarity score is between 0.3 and 0.5 then check if the first three letter Jaccard similarity score of the product names is greater than their Jaccard similarity score. If the first three letter Jaccard similarity score is greater, it means one of the products may have a spelling mistake, products may be identical. Add the anchor product and the candidate product to the candidate duplicate product records list else skip to the next candidate product.

(3) If the similarity score is greater than 0.5 products may be identical. Add the anchor product and the candidate product to the candidate duplicate product records list else skip to the next candidate product.

4) Send the generated candidate duplicate product records list to human referees to get each pair labeled as ‘duplicate’ or ‘not duplicate’. Table 4.1 illustrates a sample output of the candidate duplicate product records list generation process. The term ‘Product’ in Table 4.1 refers to the ‘anchor product’. The term ‘SimilarProduct’ refers to the ‘candidate product’. The term ‘HeroProduct’ 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.

Table 4.1: A sample output of candidate duplicate product records list.

Product	SimilarProduct	HeroProduct	Similarity
Id_1	Id_2	None	0.633
Id_3	Id_4	Id_4	0.453
Id_5	Id_6	None	0.708
Id_7	Id_8	None	0.615
Id_9	Id_10	None	0.755
Id_11	Id_12	None	0.704
Id_13	Id_14	None	0.848
Id_15	Id_16	None	0.686
Id_17	Id_18	Id_18	0.457
Id_19	Id_20	None	0.489
Id_21	Id_22	None	0.659

4.6 The Deduplication Platform

An internal web page is designed and developed by the IT teams of the company to let the human referees to view, compare and label the detected candidate duplicate product pairs. Figure 4.2 is an example of a candidate duplicate product pair that is listed on the deduplication platform.

Figure 4.2: An example of a candidate duplicate product pair 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 Figure 4.3.

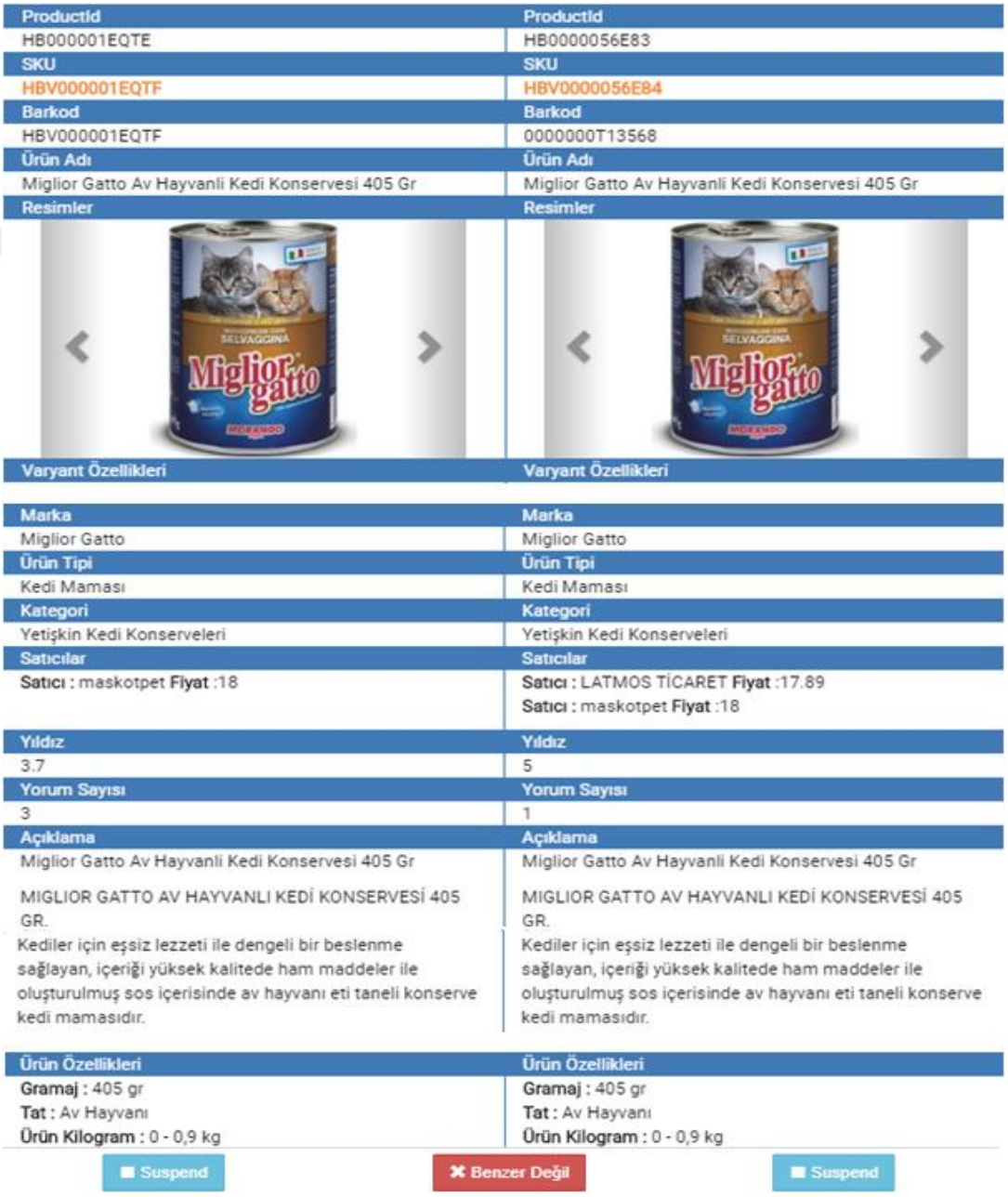

Figure 4.3: An example of a candidate duplicate product pair shown at the product pair detail page.

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. Figure 4.4 is an example log of a labeled product pair.


```
{
  "_index": "productsimilaritylogs",
  "_type": "processlog",
  "_id": "2893a4b4-5a6b-4a6d-92b7-cb5bdc8a66aa",
  "_score": 1,
  "_source": {
    "sku": "HBV00000682BT",
    "similarSku": "HBV0000058W97",
    "processSku": "HBV00000682BT",
    "percantage": 100,
    "process": "Suspend",
    "user": "oagca",
    "processDate": "2019-10-17T13:45:58.9120525+00:00",
    "processDetail": "Operation 'Suspend' applied by oagca.",
    "heroSku": "",
    "id": "2893a4b4-5a6b-4a6d-92b7-cb5bdc8a66aa"
  }
},
```


Figure 4.4: An example of labeled product pair.

The meaning properties of the log illustrated in Figure 4.4 can be detail as follows:

- **sku**: Corresponds to the id of the anchor product.
- **similarSku**: Corresponds to the id of the candidate product.
- **processSku**: Keeps the record of which one of the products is closed.
- **percentage**: Corresponds to the similarity score of the products.

- **process**: Can be either `Suspend' or `Cancel'. Suspend means products are duplicate. Cancel means products are not duplicate.
- **user**: Keeps the record of the referee who performed an action on the product pair.
- **processDate**: Keeps the date of the action of the referee.
- **processDetail**: A summary of the action and the referee who performed the action.
- **heroSku**: Corresponds to the id of the hero product. If none of the products is a hero product, then this field will be an empty string.

5 TEXT BASED MODEL

In the second phase, classification models are trained and tested by candidate product pairs that are labeled in the first phase. The target value of the classification models are the labels which 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 that are 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 the 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**: min (w_1, w_2)
- **Minimum word count**: min (w_1, w_2)
- **Word count difference over maximum word count**: $abs(w_1, w_2)/\max(w_1, w_2)$
- **TF-IDF cosine similarity:** TF-IDF cosine similarity of the names of products P_1 and P_2 is defined as a feature. To calculate the TF-IDF scores of words, each product name is treated as an independent document. Since each brand has its own important words

that differentiates it from other brands, the document set is limited to product names belonging to the same brand.

- **Binary cosine similarity**: Product names of the products P_1 and P_2 can be vectorized without using TF-IDF weights. Instead of using TF-IDF weights, each word weighted equally as 1. The cosine similarity of the vectors is defined as a feature and named as binary cosine similarity.
- **Jaccard similarity**: Jaccard similarity of product names P_1 and P_2 is defined as a feature and named as Jaccard Similarity.
- **Product description similarity**: When the dataset is investigated it is uncovered that some of the product pairs may not be duplicate even when they are identical textually or visually. It is possible to differentiate a significant proportion of such products by their descriptions. Hence, Jaccard similarity of product descriptions of P_1 and P_2 is defined as a feature.
- **First three letter Jaccard similarity**: The method and the reason for producing this feature is explained in detail in Chapter 4.
- **Is first three letter 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 for producing this feature is explained in detail in Chapter 4.
- **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 letter 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 product 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.

- **Minimum occurrence count:** Minimum of m and n is also defined as a feature for the same reason given for the maximum occurrence count.
- **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 chapter, 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 on the following areas:

The text-based model is able to 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 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 decrease, the products need to have a higher similarity score in order to be duplicated. This is because of the fact that as the number of words in the product name increases, the pool of words that can be used to identify the product also increases. Figure 5.1 is an example product pair for the explained scenario. The text-based model is able to classify the product pair illustrated at Figure 5.1 as ‘duplicate’.

Figure 5.1: An example product pair.

The text-based model also can adapt 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. Figure 5.2 is an example product pair for this scenario. The brand illustrated in Figure 5.2 has tens of products with identical product names: ‘Pregnant Dress’. Using the features ‘maximum occurrence count’ and ‘minimum occurrence count’ the text-based model is able to classify the product pair as ‘not duplicate’.

Figure 5.2: An example product pair.

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

In the third phase, classification models are trained and tested by candidate product pairs that are labeled in the first phase. The target value of the classification model is the labels which are produced by human referees as ‘duplicate’ or ‘not duplicate’. The purpose of this phase is to enrich the text-based model by studying the results and to produce new features in order to improve the failure 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

In addition to the features used in the text-based model we add the following features to enrich the model:

- **Image similarity:** Image similarity scores of the products are defined as a feature. We will explain how we calculate image similarities in Chapter 6.2.
- **Image filter:** When the train set is investigated it is discovered that 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:** It is also discovered that the majority of the duplicated records have Jaccard similarity 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 percentile set to 50th and 90th.
- **Position of the image similarity among other candidates of the brand:** 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 percentile set to 75th.

- **Position of the Jaccard similarity among other candidates of the brand:** The feature is defined as whether the Jaccard similarity of P_1 and P_2 is greater than the 25th percentile of Jaccard 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 percentile set to 75th.
- **Position of the TF-IDF cosine similarity among other candidates of the brand**: The feature is defined as whether the TF-IDF cosine similarity of P_1 and P_2 is greater than the 25th percentile of the TF-IDF cosine 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 percentile set to 75th.

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 [30]. 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. $1 - Frobenius\ Norm$ 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 [30] into our project.

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 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 among other candidates of the brand’ and ‘position of the TF-IDF cosine similarity among other candidates of the brand’ give the model the ability to position the text 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. In this way, 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’. Figure 6.1 is an example product pair of the explained scenario. The enriched model is able to classify the product pair illustrated in Figure 6.1 as ‘not duplicate’ even though the product names are almost identical.

Figure 6.1: An example product pair which is differentiated by image similarity.

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’. Figure 6.2 is an example product pair of the explained scenario. The enriched model is able to classify the product pair illustrated in Figure 6.2 as ‘not duplicate’ even though the product names are identical and product images are similar.

Figure 6.2: An example product pair which is differentiated by price. The left product 174,28 Turkish Lira. The right product is 88,53 Turkish Lira.

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 7.1 shows the proportional distribution of the categories of product pairs in the overall dataset. Category selection and prioritization is made by the business units of hepsiburada.com. As a result of the choices made according to the priorities of the business units, the distribution took place as in Table 7.1.

Table 7.1: Product categories and rates in dataset.

Categories	Rates	Pair Counts
Mother - Baby	0.24	8161
Auto Accessory	0.19	6459
Construction Market	0.18	6123
Supermarket	0.11	3738
Pet shop	0.07	2382
Toys	0.05	1702
Cosmetic	0.05	1698
Stationery / Office	0.04	1361
Home Decoration	0.03	1023
Sport	0.02	676
Home Textiles	0.02	684

Table 7.2 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, binary 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.

Table 7.2: Distribution 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
Binary 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
Union over intersection	1.34	0.50	1.00	14.00

7.2 Scatter Plot Comparisons of Basic Features Based on Jaccard Similarity

Figure 7.1 illustrates a scatter plot comparison of Jaccard similarity and TF-IDF cosine similarity, binary cosine similarity, image similarity, product description similarity, edit distance of the product pairs.

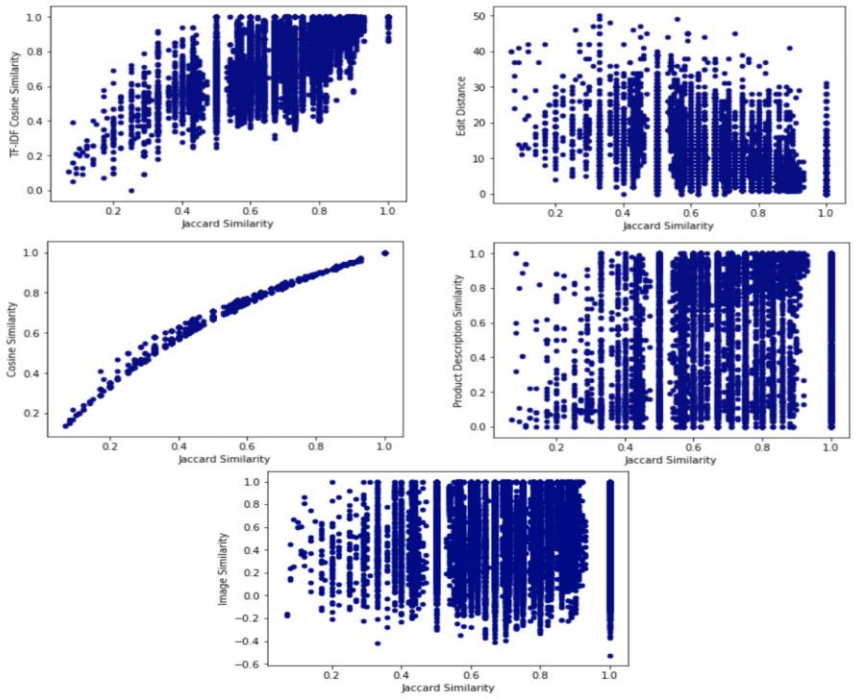

Figure 7.1: Scatter plot comparisons of basic features based on Jaccard similarity.

The prominent correlations among Jaccard similarity and TF-IDF cosine similarity, Jaccard similarity and binary cosine similarity are expected due to their token-based natures. Since both Jaccard, TF-IDF cosine and cosine 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 technique; they naturally are 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 gave the same name to the product. Second, the relation among 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 TF-IDF Vectors

As we discussed in Chapter 2, for blocking purposes, we divided the product catalog into pieces according to brands and main categories of the products. Since each piece has a different number of products, the documents collected for TF-IDF calculations of each piece have a wide spectrum of word counts. Table 7.3 shows the distribution of the TF-IDF vector sizes of the divided pieces of the dataset according to their mean, standard deviation, minimum value and maximum value. Indicators of the table are calculated using all product names of the brands that we generated candidate duplicate pairs.

Table 7.3 Distribution of the TF-IDF vector sizes.

Mean	Std	Min	Max
82.509	2269.089	2	42114

7.4 The Results Obtained from The First Phase - The Rule-Based Solution

The first phase is to identify candidate pairs with a rule-based solution and a low threshold in order to create a labeled data set. 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 phase. Therefore, metrics such as accuracy, F1 score, and recall cannot be calculated for the results obtained from the first phase.

As stated in Chapter 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.

7.5 The Results Obtained from The Second Phase - The Text Based Model

Table 7.4 shows the results from the text-based model, using various classifiers. Results are compared based on accuracy, f1 score, precision and recall metrics.

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. The precision score was calculated only for the true positives. This result confirms our argument that using adaptive methods to detect duplicate product records would yield better results than non-adaptive methods.

Table 7.4: 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 7.5 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.

Table 7.5: Feature importance of the text-based model features.

Feature Name	Importance
Maximum Occurrence Count	0.341
TF-IDF Cosine Similarity	0.120
Product Description Similarity	0.118
Edit Distance	0.062
Jaccard Numerical Entity Similarity	0.050
Maximum Word Count	0.037
Maximum Product Name Length	0.035
Minimum Product Name Length	0.030
Minimum Word Count	0.030
Union Over Intersection	0.028
First Three Letter Jaccard Similarity	0.026
Jaccard Similarity	0.025
Binary Cosine Similarity	0.025
Word Count Difference Over Max. Word Count	0.021
Important Word Similarity	0.013
Subset First Three Letter Similarity	0.013
Subset Similarity	0.009
Product Type Similarity	0.008
Jaccard Textual Entity Similarity	0.002
Is First Three Letter Jaccard Similarity Greater	0.001

7.6 The Results Obtained from The Third Phase - The Enriched Model

Table 7.6 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 7.6: 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 7.7 shows obtained impurity-based feature importance from the random forest classifier model. The results show that 'Image Similarity' dominates other features.

Table 7.7: Feature importance of the enriched model features.

Feature Name	Importance
Image Similarity	0.413
Product Description Similarity	0.087
TF-IDF Cosine Similarity	0.078
Price Difference Over Minimum Price 1	0.074
Price Difference Over Minimum Price 2	0.074
Price Difference Over Minimum Price 3	0.068
Jaccard Similarity	0.049
Image Filter	0.038
Word Count Difference Over Max. Word Count	0.029
The Position of the Image Similarity Among Other Candidates of the Brand - 1	0.015

The Position of the TF-IDF Cosine Similarity Among Other Candidates of the Brand - 2	0.012
The Position of the Image Similarity Among Other Candidates of the Brand - 2	0.009
The Position of the Jaccard Similarity Among other Candidates of the Brand - 1	0.009
Product Type Similarity	0.009
The Position of the TF-IDF Cosine Similarity Among Other Candidates of the Brand - 2	0.009
Jaccard Filter	0.008
Important Word Similarity	0.004
The Position of the Jaccard Similarity Among other Candidates of the Brand - 2	0.003
Is First Three Letter Jaccard Similarity Greater	0.002

7.7 Result Comparisons

Table 7.8 compares the accuracy scores of the text-based model, the enriched model and the rule-based solution. Since an accuracy score could not be calculated for the rule-based solution the comparison is made between the text-based model and the enriched model. Results show that the enriched model outperforms the text-based model.

Table 7.8: Accuracy score comparisons of the text-based model and the enriched model.

Classification Model	Text-Based Model	Enriched Model
Random Forest	0.833	**0.860**
SVM	0.795	**0.828**
K-NN	0.801	**0.813**
Decision Tree	0.792	**0.823**
Logistic Regression	0.760	**0.790**
AdaBoost	0.774	**0.821**

Table 7.9 compares the F1 scores of the text-based model, the enriched model and the rule-based solution. Since an F1 score could not be calculated for the rule-based solution the comparison is made between the text-based model and the enriched model. Results show that the enriched model outperforms the text-based model.

Table 7.9: F1 score comparisons of the text-based model and the enriched model.

Classification Model	Text-Based Model	Enriched Model
Random Forest	0.814	**0.842**
SVM	0.768	**0.804**
K-NN	0.779	**0.791**
Decision Tree	0.767	**0.799**
Logistic Regression	0.740	**0.767**
AdaBoost	0.749	**0.799**

Table 7.10 compares the precision scores of the text-based model and the enriched model and. The precision score of the Rule-Based Solution is calculated as 0.363. Results show that the enriched model outperforms the text-based model and the rule-based solution.

Table 7.10: Precision score comparisons of the text-based model and the enriched model.

Classification Model	Text-Based Model	Enriched Model
Random Forest	0.798	**0.853**
SVM	0.757	**0.810**
K-NN	0.740	**0.771**
Decision Tree	0.733	**0.800**
Logistic Regression	0.657	**0.721**
AdaBoost	0.699	**0.780**
Rule Based Classifier	N/A	N/A

Table 7.11 compares the recall scores of the text-based model, the enriched model and the rule-based solution. Since the recall score could not be calculated for the rule-based solution

the comparison is made between the text-based model and the enriched model. Results show that the enriched model outperforms the text-based model and the rule-based solution.

Table 7.11: Recall score comparisons of the text-based model and the enriched model.

Classification Model	Text-Based Model	Enriched Model
Random Forest	0.807	**0.832**
SVM	0.759	**0.794**
K-NN	0.775	**0.783**
Decision Tree	0.762	**0.789**
Logistic Regression	0.743	**0.764**
AdaBoost	0.745	**0.792**

Figure 7.2 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:** Features derived from product name similarities.
- **Brand Features:** Features derived from brand behaviors.
- **Product Description Features:** Features derived from product description similarities.
- **Product Type Features:** Features derived from product type similarities.

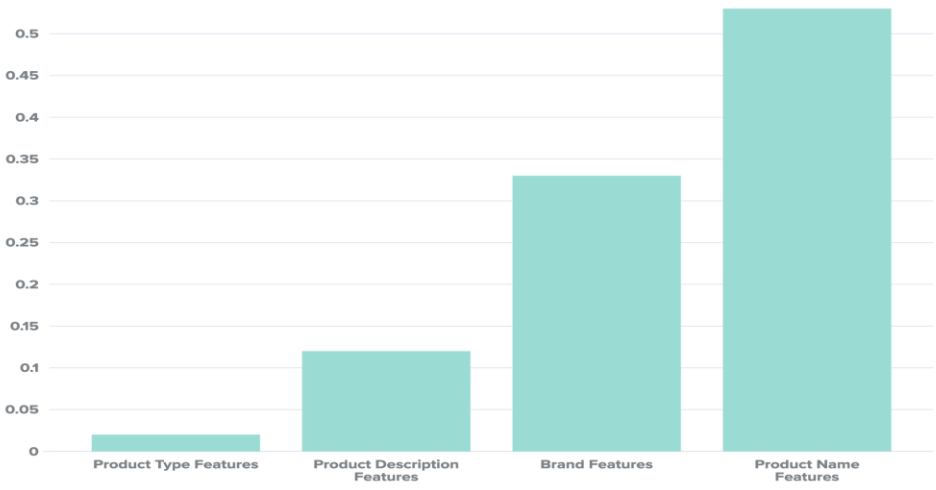

Figure 7.2: Aggregated importance of text-based model's features in terms of Gini importance.

Figure 7.3 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:** Features derived from image similarities.
- **Price Features:** Features derived from price differences.
- **Product Name Features:** Features derived from product name similarities.
- **Brand Features:** Features derived from brand behaviors.
- **Product Description Features:** Features derived from product description similarities.
- **Product Type Features:** Features derived from product type similarities.

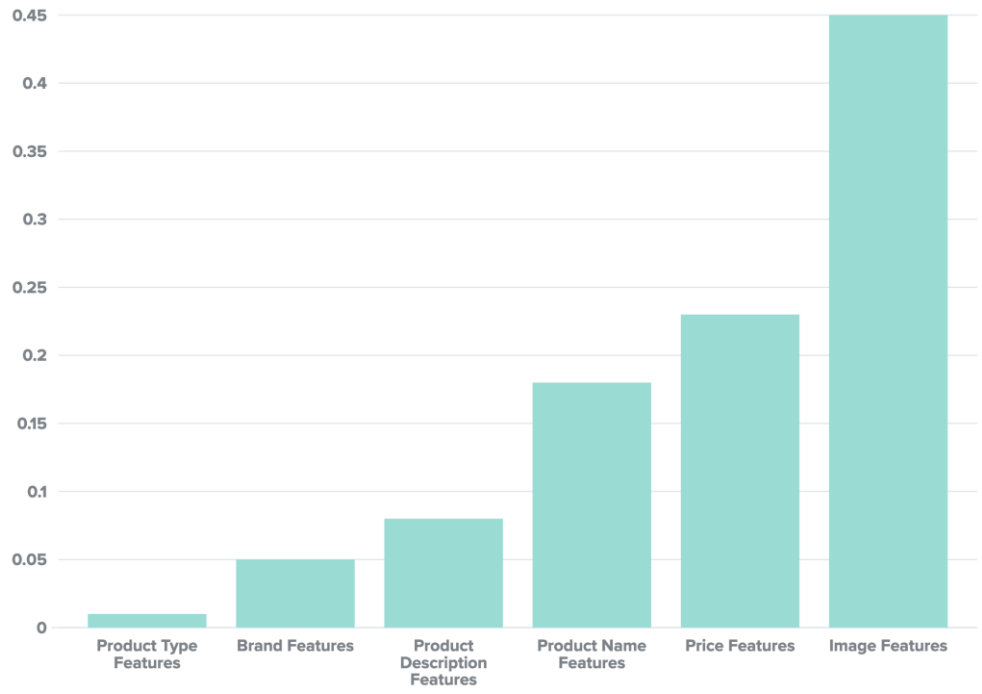

Figure 7.3: Aggregated importance of enriched model's features terms of Gini importance.

8 CONCLUSION AND FUTURE WORK

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

In the first phase, similar product pairs are detected using traditional text similarity algorithms. Product pairs that are above a certain threshold are recognized as candidate duplicate product pairs. The candidate duplicate product list generated under this rule is sent to human references to be labeled as ‘duplicate’ or ‘not duplicate’. The precision

In the second phase, classification models are trained using labeled product pairs. Binary classification models are trained using the 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 increase the precision of the classification compared to rule-based solution from 0.363 to 0.798.

In the third phase, the text-based model is enriched using similarities of the product images. It is observed that the enriched model, where product images are also used as features, performed better than the text-based model in experimental results. The experimental results show that using the enriched model in the third phase increased the precision score from 0.798 to 0.853.

The engine developed for Hepsiburada.com is put into use within the company. Approximately 15,000 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 vendors. Thus, the product catalog will be kept clean at all times.

9 REFERENCES

[1] OECD (2019), “Unlocking the Potential of E-Commerce”, OECD Going Digital Policy Note, OECD, Paris, "www.oecd.org/going-digital/unlocking-the-potential-of-e-commerce.pdf", Accessed 18 Jun 2019.

[2] A. E. Monge and C. P. Elkan, “An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records” *Proc. SIGMOD 1997 Work. Res. issues data Min. Knowl. Discov.*, no. August, pp. 23–29, 1997.

[3] S. Sarawagi and A. Bhamidipaty, “Interactive Deduplication Using Active Learning” p. 269, 2002, doi: 10.1145/775047.775087.

[4] J. Li, Z. Dou, Y. Zhu, X. Zuo, and J. R. Wen, “Deep Cross-Platform Product Matching in E-Commerce” *Inf. Retr. J.*, vol. 23, no. 2, pp. 136–158, 2020, doi: 10.1007/s10791-019-09360-1.

[5] A. McCallum, K. Nigam, and L. H. Ungar, “Efficient Clustering of High-Dimensional Data Sets With Application to Reference Matching” *Proceeding Sixth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 169–178, 2000, doi: 10.1145/347090.347123.

[6] S. E. et al. Cohen, W. W., Ravikumar, P., Fienberg, “A Comparison of String Distance Metrics for Name-Matching Tasks” *IIWeb*, vol. 2003, pp. 73–78, 2003, doi: 10.1002/spe.4380120106.

[7] M. Raeesi, M. Asadpour, and A. Shakery, “Swash: A Collective Personal Name Matching Framework” *Expert Syst. Appl.*, vol. 147, p. 113115, 2020, doi: 10.1016/j.eswa.2019.113115.

[8] J. Fisher, P. Christen, and Q. Wang, “Active Learning Based Entity Resolution Using Markov Logic” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9652 LNAI, pp. 338–349, 2016, doi: 10.1007/978-3-319-31750-2_27.

[9] T. C. Hoad and J. Zobel, “Methods for Identifying Versioned and Plagiarized Documents” *J. Am. Soc. Inf. Sci. Technol.*, vol. 54, no. 3, pp. 203–215, 2003, doi: 10.1002/asi.10170.

[10] M. Roostaee, S. M. Fakhrahmad, and M. H. Sadreddini, “Cross-Language Text Alignment: A Proposed Two-Level Matching Scheme for Plagiarism Detection” *Expert Syst. Appl.*, vol. 160, p. 113718, 2020, doi: 10.1016/j.eswa.2020.113718.

[11] W. E. Winkler, “The State of Record Linkage and Current Research Problems” *Stat. Res. Div. US Census Bur.*, pp. 1–15, 1999, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.4336.

[12] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang, “Efficient Similarity Joins for Near-Duplicate Detection” *ACM Trans. Database Syst.*, vol. 36, no. 3, 2011, doi: 10.1145/2000824.2000825.

[13] M. Bilenko and R. J. Mooney, “Adaptive Duplicate Detection Using Learnable String Similarity Measures” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 39–48, 2003, doi: 10.1145/956750.956759.

[14] Y. S. Lin, T. Y. Liao, and S. J. Lee, “Detecting Near-Duplicate Documents Using Sentence-Level Features and Supervised Learning” *Expert Syst. Appl.*, vol. 40, no. 5, pp. 1467–1476, 2013, doi: 10.1016/j.eswa.2012.08.045.

[15] V. I. Levenshtein, *Binary Codes Capable of Correcting Deletions, Insertions and Reversals*, vol. 10. 1966.

[16] A. Arasu, “Transformation-Based Framework for Record Matching” in *2008 IEEE 24th International Conference on Data Engineering*, 2008, vol. 00, pp. 40–49.
[17] S. Tata and J. M. Patel, “Estimating the Selectivity of TF-IDF Based Cosine Similarity Predicates” *SIGMOD Rec.*, vol. 36, no. 4, pp. 75–80, 2007, doi: 10.1145/1361348.1361351.
[18] N. KOUDAS, A. MARATHE, and D. SRIVASTAVA, “Flexible String Matching Against Large Databases in Practice” *Proc. 2004 VLDB Conf.*, pp. 1078–1086, 2004, doi: 10.1016/b978-012088469-8/50094-2.
[19] O. Hassanzadeh and M. Consens, “Linked Movie Data Base” *CEUR Workshop Proc.*, vol. 538, 2009.
[20] H. Köpcke, A. Thor, and E. Rahm, “Evaluation of Entity Resolution Approaches on Real-World Match Problems” *Proc. VLDB Endow.*, vol. 3, no. 1, pp. 484–493, 2010, doi: 10.14778/1920841.1920904.
[21] W. W. Cohen and J. Richman, “Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration” p. 475, 2002, doi: 10.1145/775107.775116.
[22] M. G. De Carvalho, M. A. Gonçalves, A. H. F. Laender, and A. S. Da Silva, “Learning to Deduplicate” *Proc. ACM/IEEE Jt. Conf. Digit. Libr.*, vol. 2006, pp. 41–50, 2006, doi: 10.1145/1141753.1141760.
[23] A. Thor and E. Rahm, “MOMA - A Mapping-Based Object Matching System” *CIDR 2007 - 3rd Bienn. Conf. Innov. Data Syst. Res.*, pp. 247–258, 2007.
[24] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, “Duplicate Record Detection: A Survey” *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, 2007, doi: 10.1109/TKDE.2007.250581.
[25] A. Ghani, Rayid and Probst, Katharina and Liu, Yan and Krema, Marko and Fano, “Text Mining for Product Attribute Extraction” *ACM SIGKDD Explor. Newsl.*, vol. 8, pp. 41–48, 2006.
[26] P. Ristoski, P. Petrovski, P. Mika, and H. Paulheim, “A Machine Learning Approach for Product Matching and Categorization” *Semant. Web*, vol. 9, no. 5, pp. 707–728, 2018, doi: 10.3233/sw-180300.
[27] M. H. Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg, “Where to Buy It: Matching Street Clothing Photos in Online Shops” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 3343–3351, 2015, doi: 10.1109/ICCV.2015.382.
[28] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas, “A Survey of Blocking and Filtering Techniques for Entity Resolution” *arXiv*, vol. 1, no. 1, 2019.
[29] K. O’Hare, A. Jurek-Loughrey, and C. de Campos, “An Unsupervised Blocking Technique for More Efficient Record Linkage” *Data Knowl. Eng.*, vol. 122, no. July, pp. 181–195, 2019, doi: 10.1016/j.datak.2019.06.005.
[30] J. Schroff, Florian and Kalenichenko, Dmitry and Philbin, “Facenet: A Unified Embedding for Face Recognition and Clustering” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823, [Online]. Available: https://arxiv.org/abs/1503.03832.

RESUME

Osman Semih ALBAYRAK

EDUCATION

M.S. Data Engineering, Marmara University, Istanbul, Turkey 2021

B.S. Computer Engineering, Bahcesehir University, Istanbul, Turkey 2012

WORK EXPERIENCE

April 2018 – Present:

Data Scientist, Hepsiburada, Istanbul, Turkey

July 2016 – April 2018:

Automation Developer, Hepsiburada, Istanbul, Turkey

Jan 2013 – July 2016:

Software Engineer, Huawei, Istanbul, Turkey