Data mining on Data sets of Products

Robin Klingfors
Abstract

With the increase of available data, companies are looking for ways to use it to their advantages. One tool that has become a suitable for this task is Data mining, which is a good tool to process and analyze large amounts of data. With the use of data mining, specifically machine learning, one can recognize patterns and classify data. This thesis compares different supervised machine learning algorithms with different distance algorithms for strings. App Shack, an app-developing company, is currently looking for a solution in the instance of trying to predict a commercial products original attributes in a data set despite their id or color may differ. A solution using different types of machine learning algorithms together with different string matching algorithms is evaluated and compared. The machine learning algorithms k-nearest neighbor and decision tree are compared. The distance metric is determined by the Levenshtein and Soundex algorithms. All combinations of algorithms are evaluated and the results show that the choice of string-matching algorithm is more important than the choice of machine learning algorithm. The combination of k-nearest neighbor together with Levenshtein showed the best result in the given tests.
# Contents

1 Introduction .................................................. 8
   1.1 Background .............................................. 9
   1.2 Problem ................................................ 9
       1.2.1 Delimitations .................................... 9
   1.3 Outline ............................................... 10

2 Machine learning ............................................ 11
   2.1 Supervised learning algorithms ....................... 11
       2.1.1 k - Nearest Neighbor ......................... 11
       2.1.2 Decision Tree ................................. 12

3 String comparison ......................................... 14
   3.1 Soundex ................................................. 14
       3.1.1 Soundex example ................................ 14
   3.2 Levenshtein Distance ................................. 15
       3.2.1 Levenshtein example ......................... 16
       3.2.2 Dynamic programming ......................... 17

4 Software tools ............................................. 18

5 Methodology ................................................ 19
   5.1 Libraries .............................................. 19
   5.2 Proposed Pipeline .................................... 20
       5.2.1 Data set ........................................ 20
       5.2.2 CSV file .......................................... 21
       5.2.3 Measure metric .................................. 21
       5.2.4 Example of measure metric transformation .... 22
       5.2.5 Apply the supervised algorithms ............... 22
       5.2.6 k-NN example .................................... 22
       5.2.7 Find the product ................................ 23

6 Results ...................................................... 24
   6.1 Visual Representation .................................. 25
   6.2 k-NN with Levenshtein as metric .................... 26
   6.3 k-NN with Soundex as metric ....................... 27
   6.4 Decision Tree with Levenshtein as metric ........ 28
   6.5 Decision Tree with Soundex as metric ............. 29
   6.6 Accuracy Comparison .................................. 30

7 Summary ..................................................... 31
   7.1 Discussion ............................................. 31
   7.2 Conclusion ............................................ 32
   7.3 Future Work .......................................... 32
       7.3.1 More data sets and attributes ................. 32
       7.3.2 Different Algorithms ........................... 33
1 Introduction

In the modern day, companies have access to large amounts of data. This data keeps rapidly increasing over the years and because of this, companies are looking for ways to use it to their advantages. This results in endless opportunities but it also comes with its own challenges. How do we process and analyze this increasing quantity of data and reap its benefits? For this, we need help from automated algorithms and this is where data mining comes to good use. [1].

We often want to categorize the data to determine if one component of the data is similar to another. In this thesis, the problem lies in the extracted metadata from data sets of commercial products. How do we know if a product is the same as another product in the data set? Attributes may be the same, but is it even the same brand? What types of algorithms do we use to determine this?

Data mining contains three important topics which are: statistics, artificial intelligence and machine learning. Statistics are used for the numerical study of the data. Artificial intelligence, commonly referred as a software or machine which mimic human-cognitive intelligence. Lastly, machine learning which is algorithms that adapt and learn from the data to make predictions [2]. This thesis will mainly focus on the machine learning part of data mining.

Machine learning adapts the ways of human learning. With the use of data and algorithms, it will try to mimic the way humans learn. The three main categories in machine learning is: supervised, unsupervised and reinforcement learning [3]. In this thesis, the focus lies on the aspect of supervised learning. Supervised learning is used if the goal is to predict data based on the training done on the known data.

There are a lot of supervised learning algorithms and this thesis will show the accuracy between different algorithms on a specific set of data. Due to time constraints, the algorithms which will be in focus are the common and reliable k-nearest neighbor and decision tree algorithms [4].

To use the algorithms, the choice of distance metric is important. In this thesis, we will take a look at strings. This means that we need a metric to establish a distance between two strings. This thesis will take a look at two string comparison distance algorithms, which are the Levenshtein algorithm [5] and the Soundex algorithm [6].

To test the algorithms against each other, tests were constructed to see which combination of data mining algorithm and string comparison algorithm yield they best result. Is there any difference between the combinations? Does it only matter if one would consider the data mining algorithms or do one need to consider the string comparison algorithms as well?
The results shown in the thesis proves that the choice of a good distance metric is more important than the choice of machine learning algorithm. The algorithms are good at predicting the correct brand but it’s having a harder time predicting the correct product.

1.1 Background

App Shack is a software company located in central Uppsala, specializing foremost in mobile and web-application development [7]. They currently got a client who’s looking for a way to pinpoint a specific commercial product with the help of a data set. How does one accurately tell which product in the data set are the same as the product you’re comparing to?

The data set contains meta data for over 9000 products from specific brands. Depending on the brand, you can extract different types of information and attributes. For example, some commonly available information from the Houdini brand is: Title, Description, Color, Price, Gender Brand, Images, Url Links etc. The problem arises when the brands doesn’t use the same structure for their meta data. For example, the Casall brand doesn’t have the “color” attribute. The color of Casall products are located in the title as the last word, as one can see in this example: “Tee with Cut Out Detail - Papaya Red”. How does one find a way to distinguish the different brands and then later on, different products?

1.2 Problem

The main objective of this thesis is to accurately determine which product resembles another product the most, based on the meta-data. For this, I need to determine a couple of things, which are:

- Which classification algorithm would be a good fit to determine that it’s the same brand?
- Find good metrics for string-matching and compare these.
- Calculate and analyze the result of the chosen algorithms.
- Determine the probability that this a product is the same as another one in the data set.

1.2.1 Delimitations

The work does not include finding a way to obtain the data sets. It will only focus on the two existing data sets, which are data sets of Casall and Houdini products. There is also no need to consider the time complexity of the algorithms since it’s the result that matters and not the time reaching it.
1.3 Outline

The next section covers machine learning. It goes through the background and related information about the machine learning algorithms which are used in the thesis. To use the machine learning algorithms, one would need a good distance metric, which section 3 covers. It describes the chosen string-matching algorithms and how they are executed. Section 4 explains the tools and libraries used for the thesis while section 5 describes the method used to process the data and obtain the results. Section 6 shows tables with results about the tests. Finally the final section includes a discussion of the results, summarizing the thesis and covers potential future work.
2 Machine learning

As previously mentioned, machine learning can be split into three primary parts. These are unsupervised learning which is when there is no known data given to the algorithm and it will then try to find patterns amongst the data. The second one is reinforcement learning, which discovers through trial and error. This way, it can select which action gave the greatest result. Lastly, the supervised learning method will use already known data and try to successfully predict the output based on the known data. This thesis will focus on the supervised learning [8].

2.1 Supervised learning algorithms

When it comes to supervised learning, there are a lot of different algorithms to choose from. To narrow these down, we can focus on the ones that are good for classification. Since the goal is to find the most accurate algorithm for the problem, two of the more popular algorithms are analyzed.

2.1.1 k - Nearest Neighbor

The k-nearest neighbor or k-NN for short is an algorithm which uses data points on a \( d \)-dimensional space where \( d \) is the number of attributes to determine classifications. One justification for using this algorithm is best exemplified by the following: “If it walks like a duck, quacks like a duck, and looks like a duck, then it’s probably a duck.” The \( k \) value is how many neighbors the algorithm will look at to determine which classification it belongs to. Determining a good \( k \) value can be a difficult task and will need some testing [4].

The k-NN algorithm can be broken down into smaller steps:

1. For each element in the data set:

2. Compute the distance between the unlabeled element and the element. In this thesis, Euclidean distance is used to determine the distance between two elements. The distance \( d \) between \( x \) and \( y \) is calculated by using this formula, where \( n \) is the number of dimensions and \( x_k \) is the \( k^{th} \) number of attributes [4]:

\[
d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}
\] (1)

3. When the loop above is done, select \( k \) elements closest to the unlabeled data based on the distance, where \( k \) is a predetermined value.

4. The unlabeled data is now classified based on the majority class of the selected closest neighbors.

In the figure below [4] one can find examples to get a better understanding of how the k-NN algorithm evaluate.
(a) The square to the left in the figure got k-value = 1. This can be troublesome in some cases like it is depicted in the figure. Does it belong to minus or does it belong to plus since plus is second and third closest to the data point?

(b) If there are equal number of classifications, it will pick the closest one. This is visualized in the middle square.

(c) If the k value is too high, it might not pick the correct classification. Even though it’s almost on the minus in this case, it will pick plus. The square to the right is an example of this.

2.1.2 Decision Tree

The decision tree algorithm uses if statements to determine the classification. It recursively creates a tree based on the data set. Each node is an if statement which will tell the unlabeled data where to transition. This will “lead” the data down the nodes and eventually classify the data. The nodes are made by calculating a good split between the data in the data set. To do this split, one need to evaluate the impurity of a node before it’s inserted. Some of the functions used for this impurity measure are Entropy, Gini index and Classification error. In this thesis, Gini index is used as impurity measure due to it being the default measure in the used library. To calculate the Gini index, one would use this formula:

\[
Gini\ index = 1 - \sum_{i=1}^{c} p_i(t)^2
\]

Where \( c \) is the number of classes while \( p_i \) is the relative frequency that belong to class \( i \) at node \( t \).

The decision tree algorithm is then created and evaluated by following these steps:
1. The algorithm will start with an if statement to check if it has reached a conclusion about the classification. This means that there are no other classes that could go this path through the tree, based on the data set.

2. If 1 was false, it creates a new node and finds a good split by using the Gini index.

3. The created nodes are connected to the node that the algorithm is located at and then it will recursively call itself on each of the new created nodes until we reach a conclusion in the first if statement.

<table>
<thead>
<tr>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alto Half Zip</td>
<td>Sugar Snow</td>
<td>XL</td>
<td>?</td>
</tr>
</tbody>
</table>

Figure 2: Visualization of the decision tree algorithm.

The figure above is an example of a decision tree created by the steps above. The unlabeled data is then able to be labeled with the help of the tree, which is visualized by the dotted arrows.
3 String comparison

There are a lot of algorithms used for string comparisons. There are phonetic algorithms which focus on the sound of the words, while others simply compare the strings and see how many different characters there are. String comparison algorithms are often used to determine misspellings. An example of this would be if you search for something on Google, a fast string-matching algorithm is used to determine what the user is after and based on that, give suggestions.

3.1 Soundex

The phonetic algorithm Soundex was developed by Robert C. Russel and Margaret King Odell in 1918 [9]. This algorithm use the pronunciation of words (in English) to determine the equality ratio of the words. It is designed to create a string containing the sounds instead of the actual characters [6]. It will follow this table:

<table>
<thead>
<tr>
<th>Characters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b, f, p, v</td>
<td>1</td>
</tr>
<tr>
<td>c, g, j, k, q, s, x, z</td>
<td>2</td>
</tr>
<tr>
<td>d, t</td>
<td>3</td>
</tr>
<tr>
<td>l</td>
<td>4</td>
</tr>
<tr>
<td>m, n</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 1: Values for different characters used in the Soundex algorithm.

The steps of creating the string:

1. It will start by extracting the first letter of the word.
2. Drop all occurrences of a, e, i, o, u, y, h, w.
3. Replace the letters in the table with the corresponding digit.
4. If two of the same digit are next to each other, remove one of them.
5. Add zeros until the point where the length of the result is 10 or remove characters until the length of the result is 10.
6. Concatenate the first letter from step one with the digits.

3.1.1 Soundex example

Here is an example of the Soundex algorithm being used on the word “levenshteinm” and every step is shown in the table below:
<table>
<thead>
<tr>
<th>Step</th>
<th>Word</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>evenshteinnn</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>vnstnnn</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>1523555</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>15235</td>
<td>L</td>
</tr>
<tr>
<td>5</td>
<td>152350000</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>L152350000</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Example of the steps of the Soundex algorithm.

1. The algorithm will begin by extracting the first letter of the word, which is 'L'.
2. It will remove all unnecessary letters from the word.
3. After this, the table will be used to transform the remaining letters into digits.
4. Continuous letters will be filtered out.
5. Zeros will be added to make the length of the result being ten.
6. Concatenate the 'L' with the digits to receive the result.

The initial string had been converted into a "sounding string" with the help of the Soundex algorithm. It is now possible to compared with other computed strings to see if they match. This will be done by dividing the total amount of same sounding characters with the length of the string, which is 10.

\[
\text{Number of same characters at the same index} \over \text{Length of string} \quad (3)
\]

An example of this would be to take “Robert” and “Rubin” where they yield “R163000000” and “R150000000”. Here we disregard the redundant zeros until they both have the same length. “R163” and “R150” got the same character at the same index at two places and the length of the string is 4.

\[
\frac{2}{4} = 0.50 = 50\% \quad (4)
\]

This means that the sound of making these two words is 50% similar. With this, one can say that the probability of these strings being equal is 50% based on the Soundex algorithm.

### 3.2 Levenshtein Distance

The Levenshtein algorithm measure the similarity of two strings. It will return a distance based on the number of actions needed to transform one of the strings into the other one. The operations that can be used on the strings are insert,
delete and replace a character. Each of these operations increases the distance by one. This means that a high distance value results in two very different words.

\[
lev_{a,b}(i, j) = \begin{cases} 
  \max(i, j) & \text{if } \min(i, j) = 0 \\
  \min(lev_{a,b}(i - 1, j) + 1, lev_{a,b}(i, j - 1) + 1, lev_{a,b}(i - 1, j - 1) + 1_{(a_i \neq b_j)}) & \text{otherwise} 
\end{cases} \tag{5}
\]

The recursive function above shows the steps of the Levenshtein algorithm. It will recursively test every possible way to change both strings into the other and see which solution is the “cheapest”.

### 3.2.1 Levenshtein example

Let’s consider the following example where the Levenshtein algorithm is used on “algorithm” and “agorothms”.

<table>
<thead>
<tr>
<th>a</th>
<th>l</th>
<th>g</th>
<th>o</th>
<th>r</th>
<th>i</th>
<th>t</th>
<th>h</th>
<th>m</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g</td>
<td>o</td>
<td>r</td>
<td>o</td>
<td>t</td>
<td>h</td>
<td>m</td>
<td>s</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Example of the Levenshtein algorithm.

If one would follow the algorithm on the words above, the following operation will be applied:

- At position 2, the algorithm would insert an “l” which would increase the distance by 1.
- Position 6 would replace the “o” with an “i”, also increasing the distance by 1.
- At the last position, the algorithm will remove s. Distance is increased by 1.

All of these operations above results in the Levenshtein distance value being 3. To find the ratio for these two strings being equal, one would do:

\[
(1 - \frac{\text{Levenshtein Distance}}{\text{length of longest string}}) \cdot 100 \tag{6}
\]

If this is applied on the words above, It would result in:

\[
(1 - \frac{3}{9}) \cdot 100 \approx 67\% \tag{7}
\]

This would imply that the probability for that one of the strings were meant to be the other one is approximately 67%.
3.2.2 Dynamic programming

The recursive function for the Levenshtein algorithm has two useful properties:

- The first one is optimal substructure. This means that it must be solvable using the optimal solution from its subproblems. Since the equation goes through the characters, step by step to compute the optimal solution in the end. This is shown after the “min” in the equation. The recursive calls after that will create substructures and, in the end, combine them to return the optimal solution.

- The next property is that overlapping subproblems do exist in the equation. This is where the same computations must be computed multiple times even though it has already been done earlier. Let’s say that the function has recursively called itself func(‘abd’, ‘bc’) and func(‘bd’, ‘bc’). This means that the next recursive call of func(‘abd’, ‘bc’) would also end up with one being func(‘bd’, ‘bc’). This means that the same computations at different places in the recursive tree which is an overlapping subproblem.

These two properties are needed for one to use Dynamic programming on the equation, which will result in faster computations. Dynamic programming is a way to divide and conquer a problem without computing the same computations more than once [10].
4 Software tools

The programming language Python 3 is used together with some external libraries:

- Feedparser is a library which is used to parse files. This will convert the meta data into readable information for the Pandas library. The feedparser can grab this information directly from an url, which is helpful since that’s where the data sets are available [11].

- For reading and manipulating the data, the Pandas library is used. Some examples would be create and read CSV files (comma-separated file), re-shaping, slicing or normalize. It is also highly optimized for performance [12].

- The library Numpy is used on the data. This will put some constraint on the data array. Some of these are: Fixed length, same data types and therefore the same size. This will result in much faster computations being done on the array. This is done because it would otherwise be too much for the modern laptop to handle in an accepted amount of time [13].

- Scikit-learn is a simple and efficient tool for predictive data analysis. This library is used for the supervised algorithms [14].

Rapidminer Studio is a visual workflow designer for data mining. With this software, one can create data mining projects and easily visualize graphs, trees, etc, from them. This is used to easier understand and compare the results [15].
5 Methodology

The supervised machine learning algorithms used in this thesis are k-NN and decision tree. The algorithms both come with their pros and cons. k-NN is very easy to implement for multi-class problems and it doesn’t need a training step since the data would be classified by the majority of the nearest neighbors. One of the cons is its speed declines when increasing the size of the data set. Since computation speed isn’t an issue, the k-NN algorithm is chosen for its simplicity and high accuracy. The decision tree algorithm doesn’t need a lot of pre-processing of the data sets and is easy to understand, interpret and visualize. The algorithm can’t be used on big data sets since a single tree may grow too many nodes which results in high computation time and overfitting. The initial idea was to compare even more algorithms, but due to time constraints, these two were chosen while other algorithms were considered but left out due to their cons being worse than the selected algorithms cons. Some of them were:

- Naïve bayes were left out due to it being highly biased. This means that it could classify the data based on a single attribute.

- Logistic regression requires a lot of pre-processing of the data in form of normalization and rescaling. This is possible but for simplicity, this algorithm was left out.

- The Random Forest algorithm was also left out for the same reason as Naïve Bayes. It tends to be biased to the attributes.

5.1 Libraries

To apply the data mining algorithms, the Scikit-learn library is used. The library contains simple and effective functions to train (if needed), compute and visualize the result of the algorithms. The provided data sets for the algorithms are Numpy arrays. The Numpy library makes it able to use Numpy arrays, though constraining the array, makes it faster. This will make it possible to use the data mining algorithms on the arrays in an acceptable amount of time on a standard laptop. To make the Numpy array, a dataframe is provided. This is provided by the Pandas library, which is used to read and manipulate data from a CSV-file into a dataframe. The CSV-files are obtained through the Feedparser library which is used to read information from an URL and converting it into a CSV-file.
5.2 Proposed Pipeline

The method used in the thesis, can be split into 4 steps.

1. The first step will take a look at how the data is represented and decide which attributes would be relevant. After this is done, filter out the attributes and create a CSV file with the combined data sets which only contains the relevant attributes.

2. Apply the string-matching algorithms on the needed attributes to be used as metric for measuring the distance between the strings in the CSV file and the inputted string.

3. Use supervised machine learning algorithms to figure out if a product entered by the user has higher probability of being of the brand Houdini or Casall.

4. Use string-matching algorithms to find the product with highest probability of being the product entered by the user.

5.2.1 Data set

The data sets that are provided contain a lot of different data which are not necessary relatable. Because of this, primary attributes that are available for both data sets need to be established. This table shows some of the attributes that are interesting for the classification.
When inspecting the table, one notice that some of the attributes won’t be relevant for our testing. Since one of the goals is to establish if a product got a higher probability of being Houdini or Casall, one need to filter attributes that can be extracted from both data sets. If one takes a closer look at the table, one will notice that the “price” attribute can be very different depending on which reseller we look at. The useful attributes which are available in both data sets would be “id”, “title”, “size” and “color”. “Color” in the Casall data set would be obtained through the “title” since it’s always located at the end of every products title.

### 5.2.2 CSV file

To make the data useful for the libraries later, one will need to create a CSV file of the relevant data. The Python library Feedparser, is used to parse the data sets. The parsed data will then be filtered and modified to match each other. The Pandas library will then be used to create the CSV file.

<table>
<thead>
<tr>
<th>Id</th>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>22108282038</td>
<td>Tee with Cut Out Detail</td>
<td>Papaya Red</td>
<td>38</td>
<td>Casall</td>
</tr>
<tr>
<td>20502083005</td>
<td>Seamless Leo Strap Top</td>
<td>Moving Red</td>
<td>L</td>
<td>Casall</td>
</tr>
<tr>
<td>1584349004</td>
<td>W’s Desoli Zip</td>
<td>True Black</td>
<td>L</td>
<td>Houdini</td>
</tr>
<tr>
<td>2202342294</td>
<td>M’s Alto Half Zip</td>
<td>Sugar Snow</td>
<td>XL</td>
<td>Houdini</td>
</tr>
</tbody>
</table>

Table 5: The modified data of four different products in the generated CSV file.

### 5.2.3 Measure metric

The supervised machine learning algorithms need to know the distance between two attributes and since the attributes are strings, string-matching algorithms are used as measurement metric. Every product in the data set will have the
attributes title, color and size converted into their corresponding similarity to
the inputted products attributes using the string-matching algorithms.

5.2.4 Example of measure metric transformation

<table>
<thead>
<tr>
<th>Id</th>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Desoli zip</td>
<td>black</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6: Product to compare the data set to.

If one would compare this product to the table 5, the Levenshtein algorithm
would convert the table into this:

<table>
<thead>
<tr>
<th>Id</th>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>2210823482038</td>
<td>17.39</td>
<td>10.0</td>
<td>0</td>
<td>Casall</td>
</tr>
<tr>
<td>20502083005</td>
<td>22.73</td>
<td>0</td>
<td>100</td>
<td>Casall</td>
</tr>
<tr>
<td>1584349004</td>
<td>71.43</td>
<td>50.0</td>
<td>100</td>
<td>Houdini</td>
</tr>
<tr>
<td>2202342294</td>
<td>41.18</td>
<td>10.0</td>
<td>50.0</td>
<td>Houdini</td>
</tr>
</tbody>
</table>

Table 7: The modified data in the generated CSV file.

5.2.5 Apply the supervised algorithms

Now that the data has been converted into comparable numbers, the supervised
machine learning algorithms can be used. The algorithm is used on the table 7
and it compares the products to the inputted product which now look like this:

<table>
<thead>
<tr>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 8: Product to compare the data set to.

This is because the closer the values are 100(%), the higher probability they
have to be the same product.

Before the next step, The Numpy library is used to convert the CSV file into
a data list which will increase the computation speed of the algorithms, and to
apply the machine learning algorithm, the Scikit-learn library is used.

5.2.6 k-NN example

One of the supervised algorithms used in the thesis is the k-nearest neighbor
algorithm. If k-NN with k value of 3 is used on the previous examples, the
algorithm will predict that “Seamless Leo Strap Top”, “W’s Desoli Zip” and
“M’s Alto Half Zip” are the closest neighbors and classify the inputted product
as a Houdini product since it has majority.
5.2.7 Find the product

The last step would be to use the string-matching algorithms on the specific brand in the CSV file to figure out which of these products have the highest probability of being the inputted product. In the example above, it would result in us comparing all of the Houdini products.

<table>
<thead>
<tr>
<th>Id</th>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1584349004</td>
<td>W’s Desoli Zip</td>
<td>True Black</td>
<td>L</td>
<td>Houdini</td>
</tr>
<tr>
<td>2202342294</td>
<td>Desoli zip</td>
<td>black</td>
<td>l</td>
<td>Houdini</td>
</tr>
</tbody>
</table>

Table 9: String-matched Houdini products.

The conclusion would be that the inputted product is most likely to be

<table>
<thead>
<tr>
<th>Id</th>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1584349004</td>
<td>W’s Desoli Zip</td>
<td>True Black</td>
<td>L</td>
<td>Houdini</td>
</tr>
<tr>
<td>-</td>
<td>Desoli zip</td>
<td>black</td>
<td>l</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 10: The conclusion of the examples.

The probability of it being this product is:

\[
\frac{71.43 + 50.0 + 100}{100 + 100 + 100} = \frac{221.43}{300} = 0.7381 \approx 74\%
\] (8)
6 Results

To test the classification- and string-matching algorithms on a larger scale, strings that are similar to the strings in the data sets are used. The tests will go through the products and randomly remove characters from the attributes. If one would use this on a product, it could look like this:

<table>
<thead>
<tr>
<th>Id</th>
<th>Title</th>
<th>Color</th>
<th>Size</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>2202342294</td>
<td>M’s Alto Half Zip</td>
<td>Sugar Snow</td>
<td>XL</td>
<td>Houdini</td>
</tr>
</tbody>
</table>

Table 11: Modified string

Algorithms are used on this modified product to determine its brand and id in the data set containing over 9000 of Casall and Houdini products in total.

To compare the result of different combinations of using the string distance and the machine learning algorithms, a confusion matrix [4] will be represented.

<table>
<thead>
<tr>
<th>True/Actual</th>
<th>Houdini</th>
<th>Casall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td>Houdini</td>
<td>True positive (TP)</td>
</tr>
<tr>
<td></td>
<td>Casall</td>
<td>False negative (FN)</td>
</tr>
</tbody>
</table>

Table 12: Confusion matrix when the focus lies on the prediction of the Houdini class.

The confusion matrix provide the information needed to determine the precision, recall and accuracy [4] of different classes. This is done by using different types of formulas which are listed below.

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (9)
\]

The precision is used to measure the the accuracy of the predictions. How many Houdini predictions were actually a Houdini product.

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (10)
\]

Recall measure the accuracy of how many Houdini products did it manage to
predict out of all the Houdini products.

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN}
\]  \hspace{1cm} (11)

The accuracy measures the total accuracy of all correct guessed products.

6.1 Visual Representation

When the chosen string-matching algorithm is done executing on the data. The chosen data mining algorithm will be used on the data. This visual environment represents the position of the products in a 2-dimensional space after using the Levenshtein algorithm by comparing the example above to the data.

![Visualization of the data sets obtained while comparing a product. Red dots are Houdini while black dots are Casall products.](image)

Figure 4: Visualization of the data sets obtained while comparing a product. Red dots are Houdini while black dots are Casall products.

If one would use k-NN with k-value of 11 on this, the algorithm can make a prediction that the product is of the Houdini brand. This is because the product, which is being classified, got the values 100, 100 and 100 (top right corner) for title, color and size. This represents a 100% match for the string algorithms.
6.2 k-NN with Levenshtein as metric

Table 13: Confusion matrix for predicting the brand using k-NN together with Levenshtein.

<table>
<thead>
<tr>
<th>Prediction</th>
<th>True/Actual</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Houdini</td>
<td>3699</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Casall</td>
<td>35</td>
<td>5061</td>
<td></td>
</tr>
</tbody>
</table>

Table 14: Confusion matrix for predicting the correct product using k-NN together with Levenshtein.

<table>
<thead>
<tr>
<th>Prediction</th>
<th>True/Actual</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Houdini</td>
<td>236</td>
<td>3467</td>
<td></td>
</tr>
<tr>
<td>Casall</td>
<td>4390</td>
<td>706</td>
<td></td>
</tr>
</tbody>
</table>

The confusion matrices above displays the average result of running the tests with k-NN and Levenshtein as metric five times.

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Accuracy</th>
<th>Precision(H)</th>
<th>Precision(C)</th>
<th>Recall(H)</th>
<th>Recall(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>99.6%</td>
<td>99.9%</td>
<td>99.3%</td>
<td>99.1%</td>
<td>99.9%</td>
</tr>
<tr>
<td>Product</td>
<td>10.7%</td>
<td>6.4%</td>
<td>13.9%</td>
<td>5.1%</td>
<td>16.9%</td>
</tr>
</tbody>
</table>

Table 15: The accuracy, precision and recall retrieved from the tests. H = Houdini and C = Casall.

As the table implies, this combination of data mining and string-matching algorithms is very reliable to use for the prediction of the brand with an average
accuracy of 99.6% correct prediction. The prediction of the correct product is 10.7%. With the help of the confusion matrices precision and recall is determined for each brand and product.

### 6.3 k-NN with Soundex as metric

<table>
<thead>
<tr>
<th>True/Actual</th>
<th>Houdini</th>
<th>Casall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Houdini</td>
<td>3210</td>
<td>493</td>
</tr>
<tr>
<td>Casall</td>
<td>410</td>
<td>4686</td>
</tr>
</tbody>
</table>

Table 16: Confusion matrix for predicting the brand using k-NN together with Soundex.

<table>
<thead>
<tr>
<th>True/Actual</th>
<th>Houdini</th>
<th>Casall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Houdini</td>
<td>47</td>
<td>3656</td>
</tr>
<tr>
<td>Casall</td>
<td>4966</td>
<td>130</td>
</tr>
</tbody>
</table>

Table 17: Confusion matrix for predicting the correct product using k-NN together with Soundex.

The confusion matrices above displays the average result of running the tests with k-NN and Soundex as metric five times.
<table>
<thead>
<tr>
<th>Prediction</th>
<th>Accuracy</th>
<th>Precision(H)</th>
<th>Precision(C)</th>
<th>Recall(H)</th>
<th>Recall(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>89.7%</td>
<td>86.7%</td>
<td>92.0%</td>
<td>88.7%</td>
<td>90.5%</td>
</tr>
<tr>
<td>Product</td>
<td>2.0%</td>
<td>1.3%</td>
<td>2.6%</td>
<td>0.9%</td>
<td>3.4%</td>
</tr>
</tbody>
</table>

Table 18: The accuracy, precision and recall retrieved from the tests. H = Houdini and C = Casall.

k-NN with Soundex as metric have an accuracy of 89.7% for predicting the correct brand of the product. The accuracy for predicting the correct product is 2.0%.

6.4 Decision Tree with Levenshtein as metric

Table 19: Confusion matrix for predicting the brand using decision tree together with Levenshtein.

Table 20: Confusion matrix for predicting the correct product using decision tree together with Levenshtein.
The confusion matrices above displays the average result of running the tests with decision tree and Levenshtein as metric five times.

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Accuracy</th>
<th>Precision(H)</th>
<th>Precision(C)</th>
<th>Recall(H)</th>
<th>Recall(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>99.2%</td>
<td>98.5%</td>
<td>99.7%</td>
<td>99.6%</td>
<td>98.9%</td>
</tr>
<tr>
<td>Product</td>
<td>10.8%</td>
<td>6.6%</td>
<td>13.8%</td>
<td>5.3%</td>
<td>16.9%</td>
</tr>
</tbody>
</table>

Table 21: The accuracy, precision and recall retrieved from the tests. H = Houdini and C = Casall.

Decision tree together with Levenshtein got an accuracy of 99.2% for the prediction of the brand while the prediction of the product is 10.8%.

6.5 Decision Tree with Soundex as metric

Table 22: Confusion matrix for predicting the brand using decision tree together with Soundex.

Table 23: Confusion matrix for predicting the correct product using decision tree together with Soundex.
The confusion matrices above displays the average result of running the tests with decision tree and Soundex as metric five times.

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Accuracy</th>
<th>Precision(H)</th>
<th>Precision(C)</th>
<th>Recall(H)</th>
<th>Recall(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>85.1%</td>
<td>76.0%</td>
<td>91.7%</td>
<td>86.9%</td>
<td>84.0%</td>
</tr>
<tr>
<td>Product</td>
<td>2.1%</td>
<td>1.2%</td>
<td>2.7%</td>
<td>0.9%</td>
<td>3.7%</td>
</tr>
</tbody>
</table>

Table 24: The accuracy, precision and recall retrieved from the tests. H = Houdini and C = Casall.

The last test shows that if decision tree is used together with Soundex, one would get the accuracy of 85.1% for predicting the brand. The accuracy for predicting the product is 2.1%.

6.6 Accuracy Comparison

Figure 5: Comparison of the Accuracy of the different combination of algorithms.

Comparing these algorithms, one can see that in this case, the choice of the machine learning classification algorithm isn’t the most important one. The string-matching algorithm is the more important one in this specific scenario which one can see in the picture. The bars to the left indicates that all the tests predict the brand correctly to at least 85%.
7 Summary

7.1 Discussion

Based on the results, it is possible to use machine learning algorithms in combination with the string-matching distance algorithms to determine which brand the product belong to in the data set. The accuracy between the different combinations can differ and therefore the selection of algorithms is important. While the prediction of the brand was very successful, the prediction of the exact product was not as simple to predict.

The optimal performance was obtained by using the Levenshtein algorithm as a distance measure together with the k-nearest neighbor algorithm. Although using the Levenshtein algorithm together with decision tree gave a slightly worse result, however it’s still very reliable. This indicates that the choice of distance measure algorithm was the most important in this instance.

Python and the chosen libraries proved to be a good choice for the problem. All of them worked great and were very easy to install and operate thanks to the documentation. The Feedparser and Pandas library proved to be very useful for generating and manipulating the data sets from the given URLs. Numpy made it possible to run the machine learning algorithms from Scikit-learn on the data sets in an accepted amount of time. Thanks to Scikit-learn, which already had the machine learning algorithms implemented, the implementation was effortless.

The confusion matrices for the prediction of the correct product shows a higher percentage of true positives while predicting Casall compared to Houdini products. The precision for predicting Casall’s products is more than doubled. The reason for this is that Houdini use the same substring for multiple products, for example: “Mono Air Houdi”, “Mono Air Pullover” and “Mono Air Crew”. This would in some cases throw off the string-matching algorithm depending on which random adjustments were made to the inputted string.

Soundex was outperformed by the Levenshtein algorithm by quite a large margin. One reason for this is that Soundex was developed for indexing and detect misspellings in names by using sound. This proved hard to use on strings with multiple words and when it’s comparing the result to a random adjusted inputted string. If the inputted string would be adjusted to have an important sounding letter missing from the string, the Soundex algorithm wouldn’t be able to easily predict the product.

Another data set might have given a more accurate result since it’s a binary prediction. This means that if the machine learning algorithm is uncertain of which classification it should predict for the product, it could still get it right 50% of the time.
Instead of using string-matching algorithms, one can use word embeddings [16]. Word embedding is a way to represent a word in a vector space. Different words with the same meaning is located closer to each other in the vector space. These word embeddings are obtained by using sets of language modelling and feature learning where words or phrases are mapped to vectors of numbers. This would result in a whole new set of results which can be compared and analyzed. One of the current limitations of using word embeddings is its difficulty to calculate the location of misspelled words within the vector space. The reason is the language model isn’t designed to use misspelled words. One could argue that misspelled variants of every word should be included close to their corresponding word in the vector space, but this would render an exponential growth in the size of the training model. Since this thesis uses misspelled words in its test cases, a transition to use word embeddings instead of the string-matching algorithms won’t be that simple. Since this is a known problem, there have been efforts of trying to fix this issue. MOE (Misspelling oblivious word embeddings) is one of the solutions to this problem [17]. This model uses a combination of FastText [18] with a supervised task which embeds misspellings close to their corresponding variants.

7.2 Conclusion

This thesis investigates the possibility to predict a commercial product’s brand and the original product when the product could have a different id, title, size or color. This is done by using different machine learning algorithms combined with different string-matching algorithms for measurement. The machine learning algorithms uses the brands own data sets with the original id, title, color etc. to make a prediction. The algorithms used in this thesis are k-nearest neighbor and decision tree for the machine learning part and Levenshtein and Soundex for the string-matching. The results show that the prediction of brand is very accurate using the right combination of algorithms while the prediction of the correct product proved to be a difficult task. The tests concluded that k-nearest neighbor together with Levenshtein gave the best accuracy of 99.6% while predicting the product and 10.7% while predicting the correct product. The phonetic algorithm Soundex proved to be outperformed by the Levenshtein algorithm while the choice of machine learning algorithms wasn’t as important because they performed close to equal while using the Levenshtein algorithm.

7.3 Future Work

7.3.1 More data sets and attributes

In this thesis, only two different data sets were used. They contained different types of attributes and the attributes that was extracted and analyzed were far from all the interesting ones. It would have been beneficial if the data sets followed the same structure and therefore more attributes would have been
compared. If more data sets were obtained, it might have made it possible to compare more attributes and thus yield a more precise prediction.

### 7.3.2 Different Algorithms

According to the tests, both Levenshtein and Soundex received very poor product-prediction result and the choice of machine learning algorithm wasn’t as important as the choice of string-matching algorithms. Both k-nearest neighbor and decision tree gave similar results. Therefore future work should include testing of different types of matching algorithms. Word embeddings or specifically MOE would be one of those algorithms to use as distance algorithm between the strings.

### 7.3.3 Further testing

The tests were done using a random adjusted input from the data set. While this made it possible to test many inputs in a shorter amount of time, it might have generated a more accurate result if the inputted product were taken from a reseller. Other than that, weighted results could also yield a more accurate result. Since in this case, the title was the hardest to predict, therefore the title comparison may need to weigh higher than size and color.
8 References


