Dipartimento di Informatica

Università di Pisa

Mining Glasgow Norms

Data Mining I Project

Thursday 6th January, 2022

Authors: Giulio Cordova
588294
g.cordova@studenti.unipi.it

Moreno D'Alfonso
627885
m.dalfonso@studenti.unipi.it

Bruno Javier Limon Avila
646137
b.limonavila@studenti.unipi.it

Elena Scaglione
645638
e.scaglione1@studenti.unipi.it

Contents

1 Data Understanding & Preparation

The *Glasgow Norms* is a set of ratings of English words based on 9 different features (*arousal*, *valence*, *dominance*, *concreteness*, *imageability, familiarity, age of acquisition, size, gender*). These attributes will be described during this section of the report, other aspects will be discussed in further sections. Additionally, there are other variables (not in the *Glasgow Norms* set) useful for the analysis, such as the *length* and a *polysemy* index for each word, and the *frequency* of the word found in the Google *web corpus*. The provided dataset consists of 4682 unique words.

1.1 Data Semantics

In the next paragraphs the variables will be analyzed one by one.

Arousal (continuous numerical) indicates how much excitement is felt by the person interviewed when reading the word. Its range varies from 1 to 9, where 1 means very calm and 9 very excited. This feature is an indicator of the emotional impact of a word.

Valence (continuous numerical) is a measure of how much the referent indicated by the word is valuable. The range varies from 1 to 9, where 1 is a low value indicator and 9 high value. The meaning of *value* is not specified in the dataset. It could be interpreted in a sense of economics or also personal (affection, etc.). Like *arousal*, *valence* is also an indicator of the emotional impact of a word.

Dominance (continuous numerical) is the parameter that treats the degree of control that is felt. The range varies from 1 to 9, where 1 is low control and 9 high. This is the last feature regarding the emotional impact of a word.

Concreteness (continuous numerical) represents the degree to which something can be experienced by our senses. It is a numerical variable that ranges from 1 (more abstract concepts) to 7 (words that refer to people, places and things that can be seen, heard, felt, smelled or tasted).

Imageability (continuous numerical) measures how difficult is to generate a mental image of something. It is a numerical attribute that ranges from 1, hard to imagine, to 7, easy to imagine.

Familiarity (continuous numerical) is a measure of how commonly a word is experienced. The interval scale of this numerical variable is from 1 to 7, based on how familiar participants were with the given word: 1 if they were unfamiliar; 7 if they were greatly familiar.

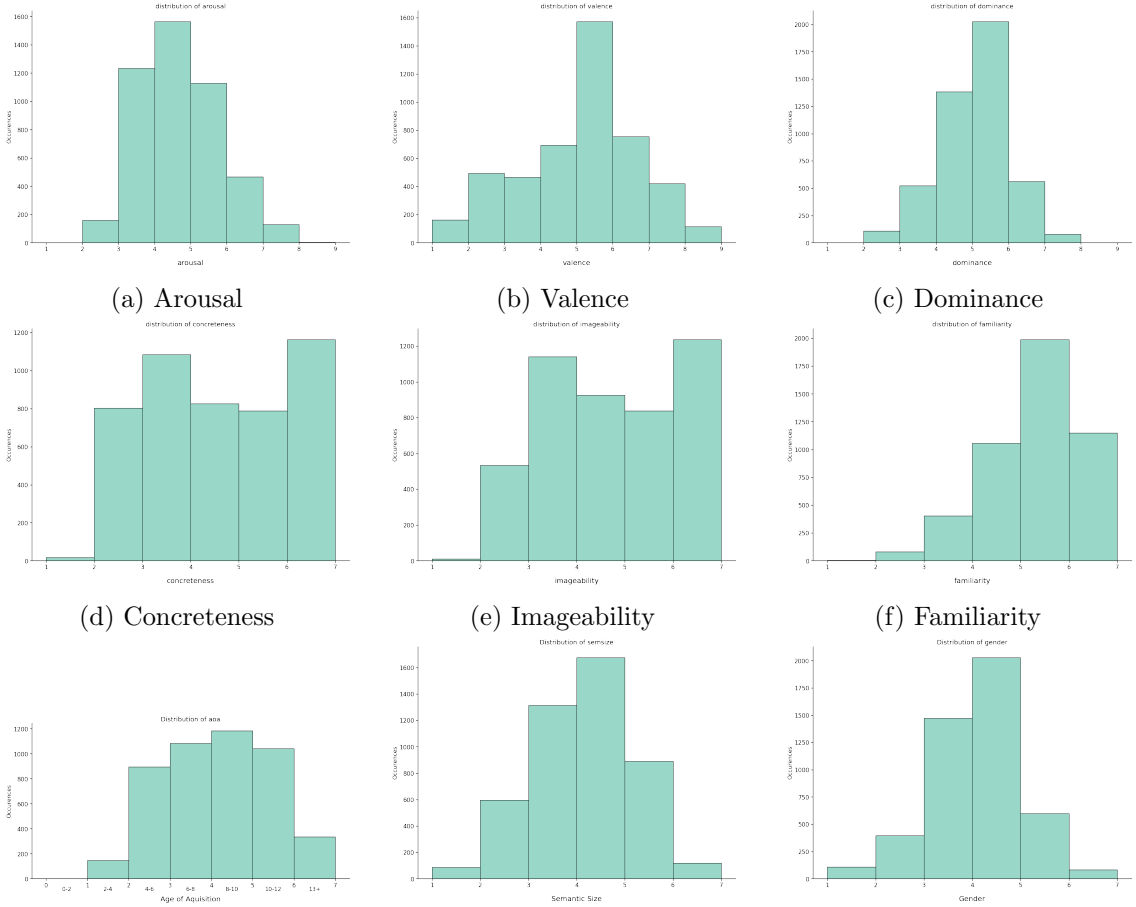
Age of Acquisition [aoa] (continuous numerical) indicates the supposed age in which a person first learned that specific word. The scale is defined as a series of consecutive 2-year periods from the age of 0 to 12 years, and a final period referring to 13 years and older. This leads us to define 7 different ranges, 0-2, 2-4, 4-6, 6-8, 8-10, 10-12 and 13+.

Size (continuous numerical) is a measure of magnitude expressed in either concrete or abstract terms (big, small). That is, if a word can be associated with adjectives like big or small (e.g. palace or qubit). The range adopted varies from 1 to 7, where 1 is small and 7 big.

Gender (continuous numerical) refers to how strongly its meaning is associated with male or female behavior. This feature could be very interesting in regards to the social bias that might, or might not, be present.
The variable is numerical and has a range that goes from 1 to 7. This may make the attribute not self-intuitive. There is no visible correlation between the number and the perceived gender of that word. Our speculation is that the higher the value, the more *masculine* the word is perceived. For example the word `actor` has a perceived gender value of 5.588, while `actress` 1.303. This may lead to a transformation of the variable, which will be eventually discussed in another section.

Length (discrete numerical) refers to the size, in number of letters, of each word in the dataset.

Web Corpus Freq (discrete numerical) is the frequency of any word appearing within the collection of all words contained within the databases of Google Newspapers, which forms a corpus, that is, a body of all words found during a given timeframe and from a specific source. Due to the enormous amount of external factors that might influence the frequency of a word, this variable is the one that shows the biggest variation between results and, as such, the most outliers.



(a) Arousal (b) Valence (c) Dominance

(d) Concreteness (e) Imageability (f) Familiarity

(g) Age of Acquisition (h) Size (i) Gender

Figure 1: Distributions of variables

Polysemy (discrete boolean) is a binary variable that denotes the capacity for a word to have multiple related meanings. This relatedness sets it apart from simple homophony, in which the similarity is

accidental. Thus, two words with polysemy have a different but related sense, be it historical or etymological, most of the time, this attribution relies on personal judgment.

1.2 Distribution of variables and statistics

A brief overview of the distribution of each variable can be seen in Figure 1.

The attributes *concreteness*, *imageability* and *age of acquisition* tend to a uniform distribution. *Familiarity* is heavily centered on the right side of the graphs, meaning that apparently the words sound very familiar to the readers. It is also the feature with the smallest standard deviation. *Concreteness* and *imageability* have a distribution similar to a bimodal shape. The other attributes are all middle-centered. To have a better idea of the values, a complete statistical overview of the variables (mean, std deviation, percentile, etc.) is reported in Table 1.

Polysemy has a very unbalanced distribution: for a better visualization, a Pie Chart is drawn in Figure 4b.

	Length	Arousal	Valence	Dominance	Concreteness	Imageability
count	4682	4682	4682	4682	4682	4682
mean	6.35	4.68	5.09	5.04	4.57	4.72
std	2.01	1.10	1.59	0.93	1.43	1.36
min	2	2.06	1.03	1.94	1.67	1.74
25%	5	3.85	4.12	4.53	3.24	3.52
50%	6	4.57	5.29	5.12	4.47	4.68
75%	8	5.42	6.09	5.60	5.97	6.03
max	16	8.18	8.65	8.37	6.94	6.94
range	1-∞	1-9	1-9	1-9	1-7	1-7
	Familiarity	Age of Aq	Size	Gender	Polysemy	Web Cor Freq
count	4682	4682	4682	4682	4682	4.67e+03
mean	5.27	4.14	4.14	4.10	0.08	2.98e+07
std	0.92	1.25	1.02	0.91	0.27	8.49e+07
min	1.65	1.22	1.37	1	0	1.28e+04
25%	4.71	3.11	3.44	3.61	0	1.67e+06
50%	5.44	4.18	4.19	4.12	0	5.70e+06
75%	5.97	5.15	4.88	4.66	0	2.23e+07
max	6.94	6.97	6.91	6.97	1	2.02e+09
range	1-7	1-7	1-7	1-7	0/1	0-∞

Table 1: Basic Statistics of each feature

1.3 Assessing Data Quality

In this section, the quality of the data will be discussed, i.e. missing values, outliers, errors and semantic inconsistencies. This kind of analysis is performed using the pandas library, unless otherwise specified.

1.3.1 Missing Values

The dataset seems to be almost free of null values. In fact, there are only 14 NaN, and all are concentrated in the *web_corpus_freq* variable, shown in Figure 2. Given the small number of missing values, a substitution does not lead to a change in the distribution, so we chose to substitute the 14 NaN with the mean value of *web_corpus_freq*.

	word	length	arousal	valence	dominance	concreteness	imageability	familiarity	aoa	semsize	gender	polysemy	web_corpus_freq
585	burgle	6	5.118	2.303	3.656	4.970	5.424	5.200	3.735	4.697	5.333	0	NaN
753	Christmas	9	7.516	7.914	5.600	5.086	6.571	6.710	1.600	6.394	3.771	0	NaN
1070	Dad	3	4.912	6.849	4.618	6.257	6.400	6.853	1.265	5.147	6.706	0	NaN
1076	Dame	4	4.194	5.594	5.469	5.125	4.969	3.697	4.969	4.548	1.242	0	NaN
1540	Facebook	8	4.971	4.857	4.486	5.943	6.229	6.829	6.314	5.114	4.171	0	NaN
1559	FALSE	5	4.636	2.941	4.206	3.455	2.765	5.700	3.086	4.500	4.353	0	NaN
2673	Mom	3	5.667	7.936	4.813	6.424	6.250	6.594	1.333	5.094	1.097	0	NaN
2724	Mum	3	4.594	7.938	4.219	6.091	6.625	6.906	1.219	5.061	1.212	0	NaN
2726	Mummy	5	5.364	7.471	4.879	5.794	6.515	6.182	1.771	4.677	1.455	0	NaN
3773	skijump	7	5.914	5.771	5.486	6.200	6.529	4.758	5.028	5.389	5.000	0	NaN
4347	TRUE	4	5.743	7.914	6.219	2.529	2.719	6.156	2.400	5.424	3.182	0	NaN
4365	TV	2	4.824	5.706	4.559	6.677	6.857	6.706	2.206	3.333	4.629	0	NaN
4373	Twitter	7	4.235	4.943	4.824	4.886	5.600	6.273	6.971	4.771	3.829	0	NaN
4668	yo-yo	5	5.059	5.800	5.636	6.455	6.424	4.484	2.800	1.875	4.206	0	NaN

Figure 2: Missing values

Looking at the values that are present in the dataset but are not present in the web corpus, we can spot some similarities. There are 3 recurrences of the same word but slightly different from one another: `Mom`, `Mum` and `Mommy`. Also the word `Dad` is not present in the corpus. In addition, there are 3 words written with full capital letters: `FALSE`, `TRUE`, and `TV`. If the corpus is case sensitive, that could be an explanation of why those words are not present.

1.3.2 Outliers

First, some attributes had to be dropped to perform the boxplot analysis. *word* is not a numerical variable, and for that reason it is not used in this plot. *web_corpus_freq* and *polysemy* can be analyzed on their own.

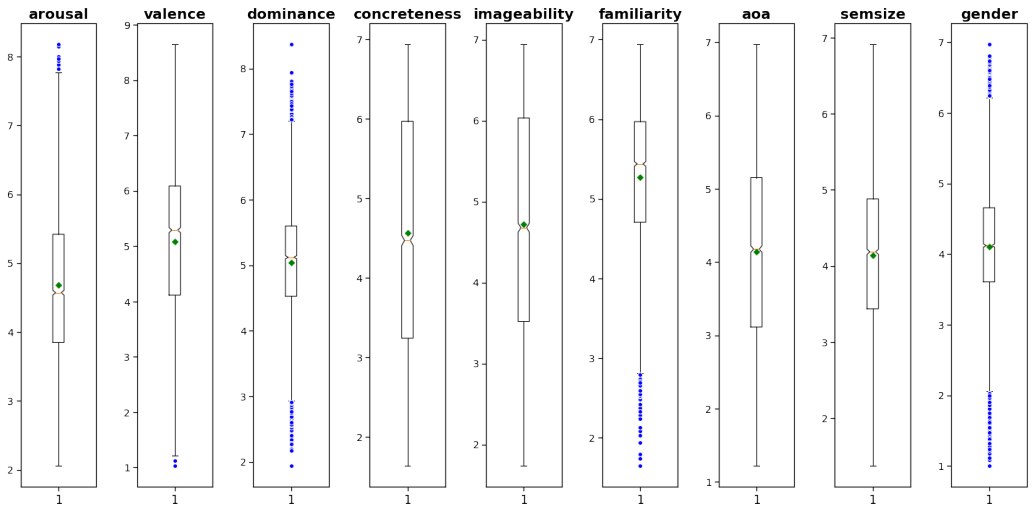
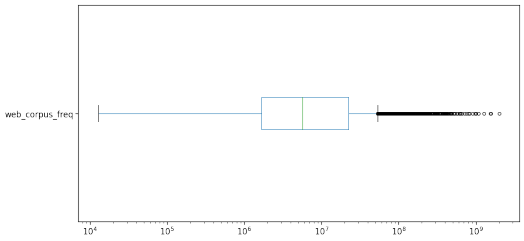


Figure 3: Boxplots for detection of outliers.

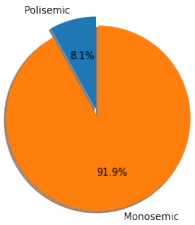
The plot in Figure 3 shows that some variables are distributed over the whole range, like *concreteness* and *imageability*, while others are not, like *length*, *arousal* and *dominance*. This issue will be later addressed with more depth.

web_corpus_freq instead seems to have a lot of outliers, as shown in Figure 4a.

polysemy has only 2 possible values, 0 and 1, that are `False` and `True`. For that reason it does not



(a) Outliers of the variable *web_corpus_freq*. The x axis was transformed using the logarithmic scale.



(b) Pie Chart of the attribute *polysemy*

Figure 4: Non-standard outliers

have outliers.

Given the high number of outliers, we decided to keep them in the dataset, because a removal could lead to a drastic reduction of the dimensionality. If the analysis in the next sections will result unsatisfying, we will remove them and perform a new one.

1.3.3 Errors

The errors in this dataset could be some values out of range or values that do not correspond to the datatype, i.e. syntactic inaccuracies. In the Table 1 one can compare the rows of range and min/max to see that there are no values out of range. With a for loop the datatype of each row was compared to the datatype expected and no errors were found.

Another test was to compare the length of the values of the variable *word* with the values of the variable *length*. The resulting numbers were exactly the same.

1.3.4 Semantic Inconsistencies

It is hard to define a semantic inconsistency, given that each word can be interpreted differently by the reader. There are many factors that can influence the understanding of the word such as background, age, origin, education, income, social status etc. Therefore every variable is difficult to evaluate in an unbiased way.

The one variable that we chose to analyze is polysemy, given its greater objectivity. One can compare all the polysemic word with a definition in a dictionary to check if the word is actually polysemic. Our analysis led to no semantic inconsistencies.

1.4 Variable transformations

Since we did not agree with the way the variables were organized in the dataset, we decided to transform some of them. These transformations will be explained in the following paragraphs.

1.4.1 Gender

As already said, the variable *gender* is not self intuitive. Our assumption is that the higher the value, the more *masculine* the word is perceived. For this reason, it was decided to change the name of this attribute to *masculinity*.

1.4.2 Web Corpus Frequency

The variable *Web Corpus Frequency* has a very vast range and does not have an easy comprehension on the linear scale. We decided to discretize the variable taking the logarithm in base 10 and flooring

the result. For example, if a word occurs $7 \cdot 10^4$ times in the Google corpus the new value of the variable will be 4; if $web\_corpus\_freq = 4 \cdot 10^7$ the new value will be 7, and so on. The new range of the variable will be between 4 and 9.

1.4.3 Normalization

Given that each variable has a different range, it was decided to normalize all the variables between 0 and 1 with the MinMax algorithm, i.e. reshaping the array with new values. The new and transformed variables are normalized as well.

1.5 Pairwise correlations and elimination of variables

An overview of the relation between the nine variables is provided in Figure 5. Where a correlation greater than |0.6| is found, we plotted the values of the two variables for a better visualization (Figure 6).

There is a strong correlation that can be seen in Figure 6a between *concreteness* and *imageability*: it is difficult to imagine an abstract word and easier to imagine a concrete one. Moreover, *concreteness* and *imageability* relate to the other variables similarly, with a margin of ±0.14. Therefore we merged them into a new variable, *perceivability*. The values of *perceivability* are the mean of concreteness and *imageability* values.

Other positively correlated variables are *valence* and *dominance*, with 0.72: the more valuable an item is perceived, the higher the degree of control over the object, as seen in Figure 6b. *Familiarity* and *age of acquisition* are instead negatively related: from the pairplot in Figure 6c is apparent that every word acquired in early age is highly familiar.

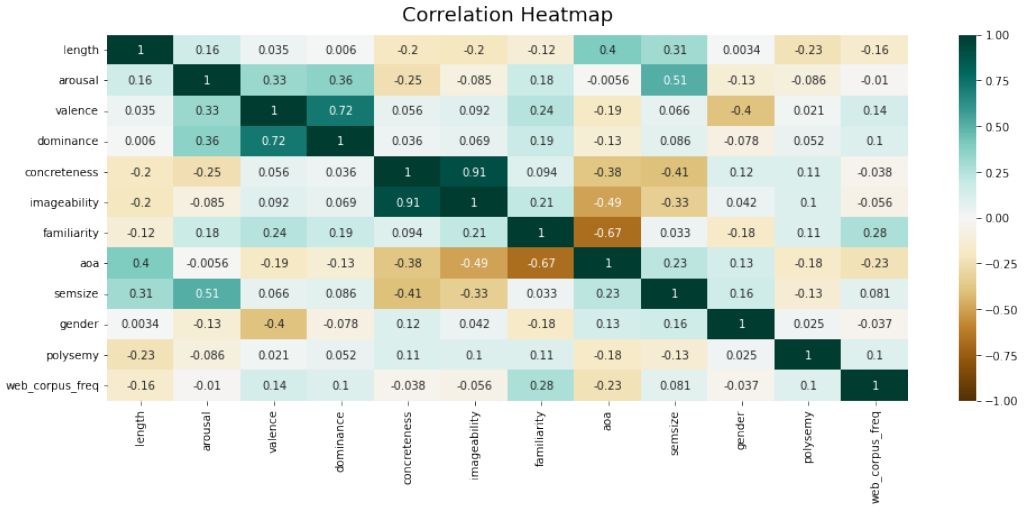
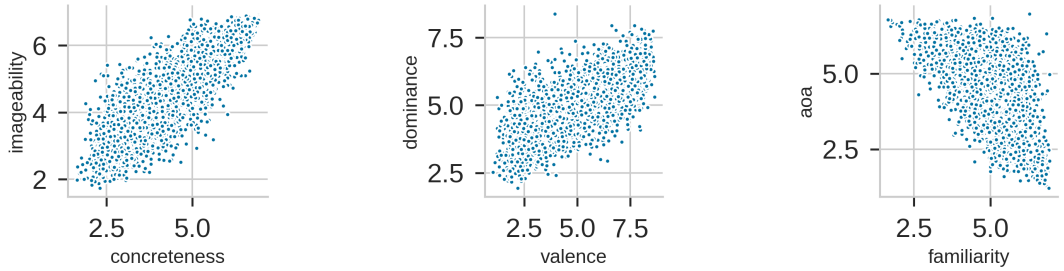


Figure 5: Pearson Correlation Heatmap of the variables.

1.6 Recap of the preprocessing

The preprocessing of the dataset is useful not only for a better visualization, but to have a clean dataset to work with.

To define it, we will recap what we just did in this section. First, we made sure that there were no errors or semantic inconsistencies. Then, we substituted the 14 missing values with the average of the missing variable. The variable *gender* was renamed to *masculinity* and the variable *web corpus frequency* was discretized. We unified the variables *concreteness* and *imageability* in a new one,



(a) Imageability vs Concreteness (b) Dominance vs Valence (c) Aoa vs Familiarity

Figure 6: Pairplot of correlated features

perceivability, substituting the mean values of the two original ones. Finally, the whole dataset (except for the variable *word*) was normalized between 0 and 1.

2 Clustering

To evaluate a cluster analysis, three different algorithms are used. Both the algorithms and the obtained results will be described in the following sections.

2.1 Preprocessing

As discussed in the previous section, the dataframe has already been reduced and normalized to perform a better analysis. The last thing to do before moving forward is to drop the variable *word*. The statistics of this dataset can be seen in Figure 7.

	length	arousal	valence	dominance	familiarity	aoa	semsize	masculinity	polysemy	web_corpus_freq	perceivability
count	4682.000000	4682.000000	4682.000000	4682.000000	4682.000000	4682.000000	4682.000000	4682.000000	4682.000000	4682.000000	4682.000000
mean	0.310597	0.428289	0.532598	0.482728	0.684871	0.508419	0.498718	0.519165	0.080948	0.457027	0.554749
std	0.143302	0.179275	0.209314	0.144739	0.174077	0.217797	0.184810	0.152787	0.272785	0.168797	0.266786
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.214286	0.292810	0.405015	0.402488	0.578042	0.329451	0.372584	0.436443	0.000000	0.400000	0.313092
50%	0.285714	0.410784	0.559275	0.494868	0.716364	0.514256	0.507766	0.522693	0.000000	0.400000	0.538709
75%	0.428571	0.549346	0.664041	0.569051	0.816704	0.683762	0.633375	0.612293	0.000000	0.600000	0.813629
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 7: Statistical information of the dataset used in the cluster analysis

In order to reduce the dimensionality of the variables to use in the cluster analysis, a Principal Component Analysis is performed. The variables are reduced to a dimension of 4682 rows x 2 columns.

2.2 Clustering analysis by K-Means

The first clustering algorithm that we analyzed is *K-Means*. It follows a simple procedure of classifying a given dataset into a number of clusters, defined by the letter k, which is fixed beforehand. The clusters are then positioned as points and all observations or data points are associated with the nearest cluster, computed and adjusted. Then the process starts over using the new adjustments until a desired result is reached.

2.2.1 Identifying the optimal number of clusters

Since the number of clusters for this algorithm is fixed beforehand, it becomes essential to identify the optimal quantity of clusters. This task can be done in a variety of ways.

For instance, one option is an empirical one, which is to try various number of clusters and then look for the best one. Another one is the *Elbow method*, that looks at the total WSS (*Within Cluster Sums of Squares*, the sum of distances between the points and the corresponding centroids for each cluster) as a function of the number of clusters: one should choose a number of clusters so that adding another cluster does not improve much the total WSS. The total WSS measures the compactness of clustering and ideally it should be as small as possible.

Finally, the approach we chose is the *Average Silhouette Method*. It measures the quality of a cluster, i.e. it determines how well each object lies within its cluster. A high average silhouette width indicates a good clustering. It computes the average silhouette of observations for different values of k. The optimal number of clusters k is the one that maximizes the average silhouette over a range of possible values for k[KR90].

After the computation (we varied the parameter from 2 to 50), the result is that the ideal number of clusters for *K-Means* is 3, as shown in Figure 8.

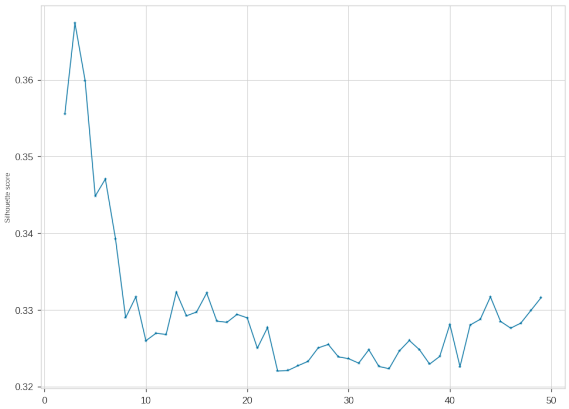


Figure 8: Average silhouette score

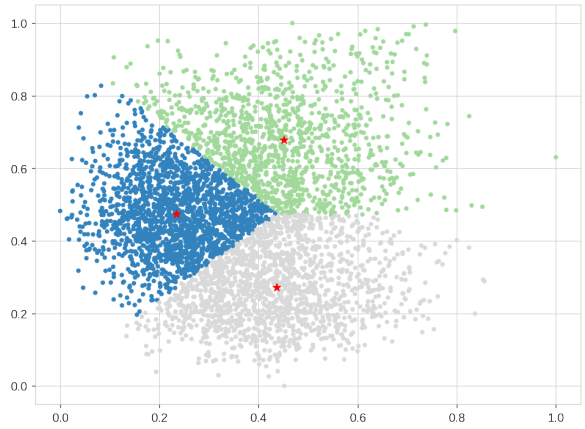


Figure 9: K-Means clustering

The value of the Silhouette score varies from -1 to 1. If the score is 1, the cluster is more dense and well-separated than other clusters. A value near 0 represents overlapping clusters with samples very close to the decision boundary of the neighboring clusters. A negative score [-1, 0] indicates that the samples might have got assigned to the wrong clusters. In this case, the highest value (closest to 1) is given by 3 clusters: 0.37.

Finally, after calculating the best possible number of cluster to use for *K-Means*, it is possible to compute the algorithm. The three clusters with their relative centroid can be seen in Figure 9.

2.3 Analysis by Density-Based Clustering

Density-based clustering locates regions of high density that are separated from one another by regions of low density. In this case, density is defined with the center-based approach, where it is estimated for a particular point in the dataset by counting the number of points within a specified radius ε of that point. This allows to classify a point as *core* point if it falls withing the interior of a dense region, *border* point if on the edge of a dense region, or finally as a *noise* point if in a sparsely occupied region.

DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) is a simple yet effective density-based clustering algorithm, and operates as follows: any two core points that are close enough—within a distance ε of one another are put in the same cluster. Likewise, any border point

that is close enough to a core point is put in the same cluster as the core point. Ties need to be resolved if a border point is close to core points from different clusters. Noise points are discarded.

Being density-based, it is resistant to noise and can handle clusters of arbitrary shapes and sizes. Thus, DBSCAN can find many clusters that could not be found using K-means.

2.3.1 Identifying the value of ε and MinPts

The basic approach is to look at the distance from a point to its k^{th} nearest neighbor, which is called the k-dist. If we compute the k-dist for all the data points for some k, sort them in increasing order, and then plot the sorted values, we expect to see a sharp change at the value of k-dist that corresponds to a suitable value of ε. If we select this elbow point as the ε parameter and take the value of k as the MinPts parameter, then

- points for which k-dist is less than ε will be labeled as core points;
- other points will be labeled as noise or border points.

Given the dimension d of the dataset, the value of k should be close to the double of d[San+98].

For this particular dataset, the ideal value of distance is located at around $\varepsilon \approx 0.02$, which is the point of maximum curvature. It represents the optimization point where diminishing returns are no longer worth the additional cost, since increasing the number of clusters will improve the fit of the model, but could also increase the risk of overfitting. To estimate more accurately ε, we varied its value between 0.012 and 0.026 with a step of 0.001, as seen in Figure 10.

In the end, it was decided to set on $\varepsilon = 0.018$ and MinPts $= 20$, for a total of 11 cluster, not counting the noise. The results of the clustering can be seen in Figure 11.

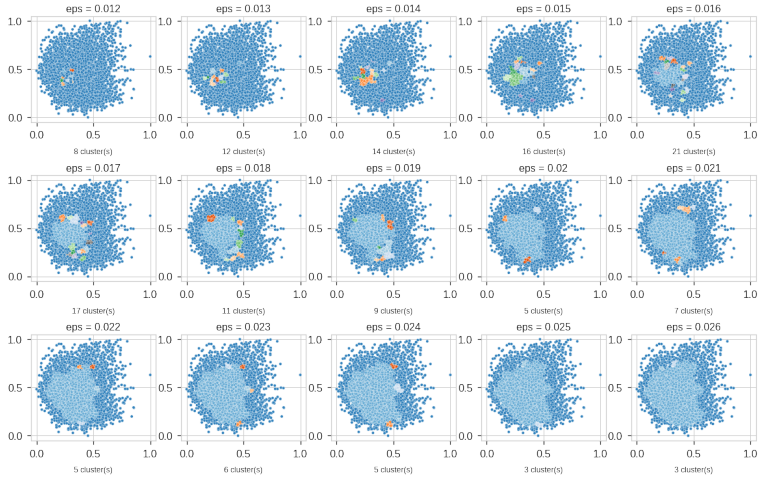


Figure 10: DBSCAN varying ε between 0.012 and 0.026

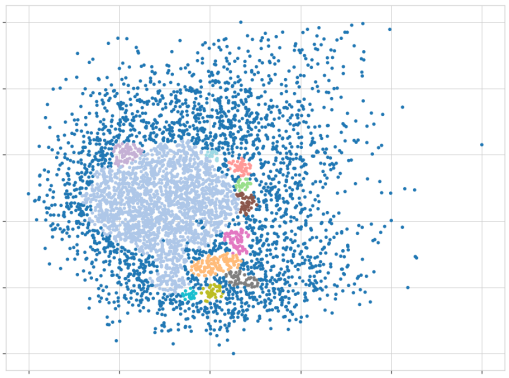


Figure 11: DBSCAN with $\varepsilon = 0.018$ and MinPts $= 20$: 11 cluster in total

2.4 Analysis by Hierarchical Clustering

Hierarchical clustering is a group of unsupervised learning algorithms that merge or split the clusters in sequence. This process can be visualized in a diagram structured as a tree, the *dendrogram*.

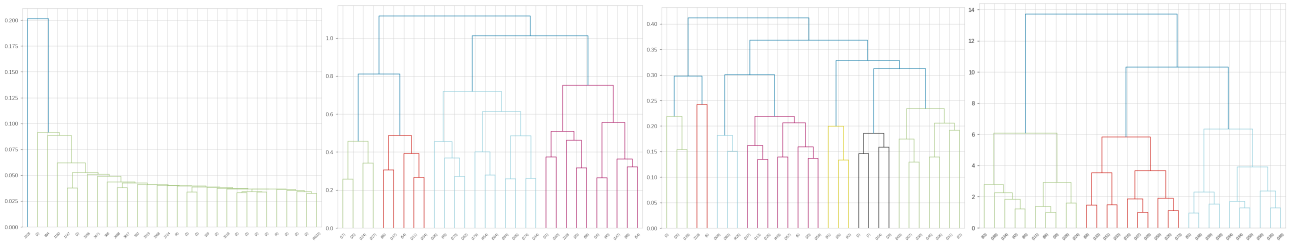
The agglomerative techniques are the most common: at first, each point represents a single cluster and then, step by step, the clusters are merged according to the distance between them. In particular, we chose the Euclidean distance.

The distance can be defined with different criteria:

- *Min* or *Single linkage* computes the proximity between the closest two points in different clusters;
- *Max* or *Complete linkage* measures the distance between the farthest points in different clusters;
- *Group Average* computes the average of pairwise proximity between points in the clusters;
- *Ward's method* minimizes the total within-cluster variance.

2.4.1 Identifying the linkage criterion and the number of clusters

In Figure 12, we can see the *dendrograms* generated by the different methods.



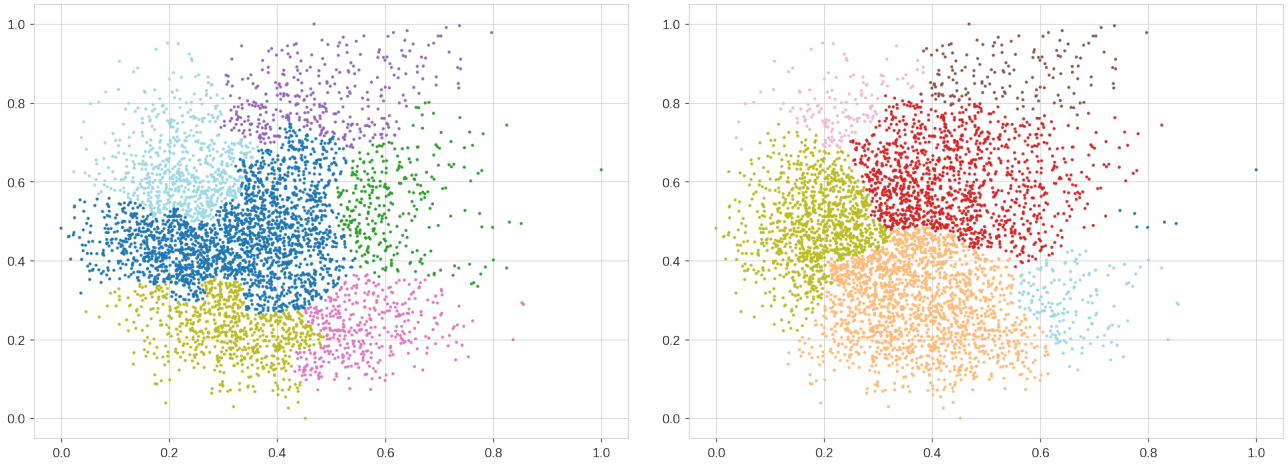
(a) Single Link (b) Complete (c) Average (d) Ward

Figure 12: Dendrograms

The *dendograms* show that the *Single Link* method gives us the worst result, with one big cluster that includes the majority of the points (regardless of the number of clusters we set), whereas *Complete* and *Average* linkage are more balanced. As expected, however the most balanced is *Ward's method*, given the nature of the algorithm.

Single Link method does not give much information, while *Ward's Method* leads to a very similar analysis to K-Means. For this reason we decided to plot the scatterplot of the *Complete* and *Average* linkage, choosing respectively to cut at 6 and 7 clusters.

The results of the clustering can be seen in Figure 13.



(a) Complete Linkage (b) Average Linkage

Figure 13: Hierarchical Clustering

2.5 Final Discussion

To validate each clustering algorithm, we decided to use two different test scores:

- the **Silhouette Score:** it estimates the distances between clusters and provides a number between -1 and 1. Where **1** means that the clusters are well apart from each other and clearly distinguished; **0** means that the clusters are indifferent, or we can say that the distance between clusters is not significant; **-1** means that the clusters are assigned in the wrong way.
- **Calinski-Harabasz (Variance Ratio Criterion):** is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters: the higher the score, the better the performances.

As a drawback, both test scores are generally higher for convex clusters than other concepts of clusters, such as density based ones (e.g. DBSCAN). The scores for the clusters are tabulated in Table 2.

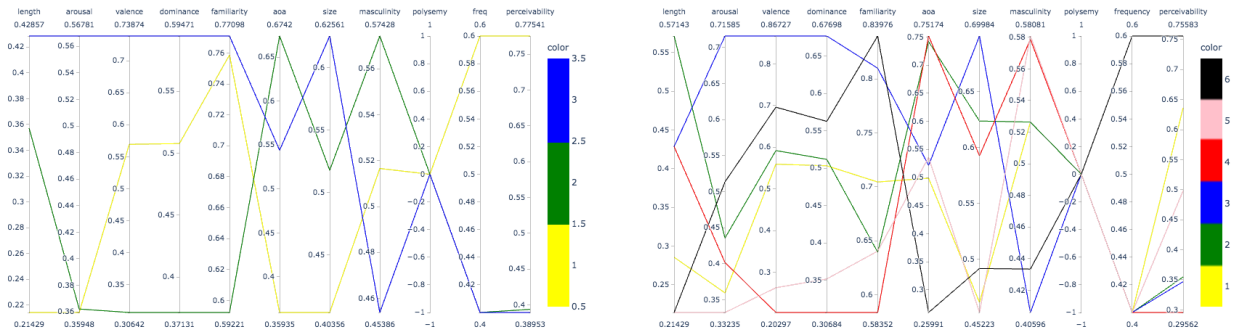
Evaluation scores	K-Means	DBSCAN	Hierarchical
Silhouette:	0.36	-0.34	0.21
Calinski-Harabasz:	3575.89	59.95	2074.84

Table 2: Evaluation of different test scores for the type of analyzed cluster algorithms

According to the values in the Table 2, K-Means has the best scores among the three algorithms. But we have to take into consideration that the high value of the coefficients for the K-Means algorithm may be biased due to the low number of clusters used (i.e. 3).

Additionally, one can perform an analysis by looking at the parallel plots of the variables according to the assigned cluster. These plots are obtained by taking the median value of every attribute (grouped by cluster) and connecting them with a colored line. The results are showed in Figure 14.

This kind of analysis is useful to determinate some key characteristics of the clusters.



(a) K-Means (b) Hierarchical Clustering

Figure 14: Parallel Plots for the two cluster algorithms. Each line represents a cluster.

For instance, in the K-Means analysis (Fig. 14a) the Second Cluster (green line) has a low value for *familiarity*, high value for *aoa*, and low value for *web corpus freq*. This indicates that the words that belong to this cluster may be difficult words, and for that reason they are not commonly used. To cross check this analysis one can see some words of this cluster (e.g. *abbey*, *zeal*), which confirm this hypothesis. Some of these words are classified by the Hierarchical Clustering (Figure 14b) algorithm in the Fourth cluster (red), which has similar characteristics to the Second cluster of K-Means.

Conversely, the words of the Third Cluster (blue) of Hierarchical Clustering have high *valence*, high *dominance*, high *semsize* and low *perceivability*. These properties may indicate that this cluster contains words that convey a high (emotional) impact. Finally, the Sixth cluster (black) may represent words that are easy, short and acquired in early age.

3 Classification

The classification task was performed using three different methods, which will be described in following sections. Before that, we briefly expose the data preprocessing used for this analysis.

3.1 Preprocessing

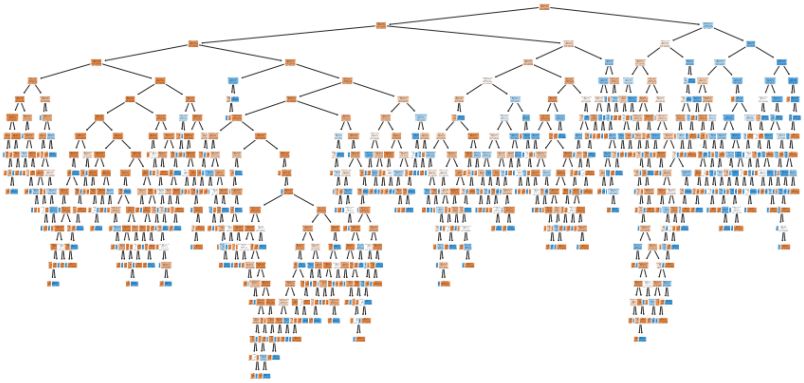
Similarly to the cluster section, we decided to use our transformed dataset, described in Figure 7.

However, for the classification process the features excluded from the normalization are *length* and *web_corpus_frequency*, which remain discrete.

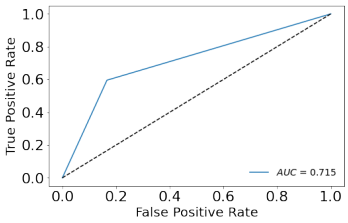
Once chosen, the target variable is removed from the dataset and stored in another array. The choice and handling of this variable is described in section 3.5. It is also necessary to split the dataset in a training (70% of total) and a testing set (30% of total), which remain the same for the three analyzed algorithms.

3.2 Decision Trees

According to the scikit-learn documentation, Decision Trees (DTs) are a *non-parametric supervised learning method used for classification and regression*[Ped+11]. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.



(a) Overfitted tree



(b) ROC curve for the testing set

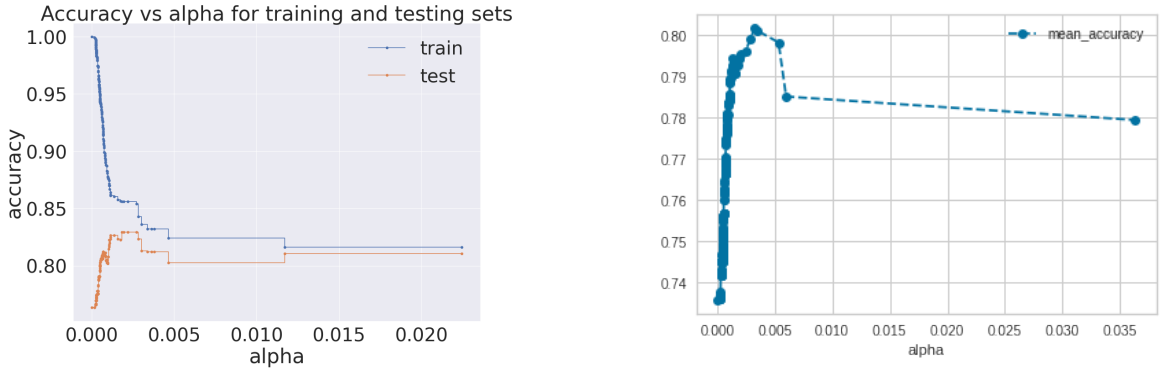
Figure 15: Unpruned tree and ROC for target variable *arousal*

If we build the tree in a naive way, using the algorithm with the default parameters (Gini impurity measure, max_depth=0, etc...), we obtain a very complex tree (Figure 15a). This tree is overfitted and therefore does not perform well on the testing set, as seen in the ROC curve in Figure 15b.

To avoid this behavior, we can introduce the parameter *Cost Complexity Pruning* (CCP) α, which prevent the tree from overfitting. This parameter is particularly useful because the tuning of the other parameters (*Max Depth*, *Min Samples Leaf*, etc.) is not necessary anymore.

The choice of the value of the parameter α is determined by varying it over several attempts and calculating the tree score for the training set as well as the testing set, as shown in Figure 16a. To avoid statistical errors, the whole process is repeated, performing a 10-fold cross validation. Finally,

the mean accuracy value for the testing and training set is calculated with a 10-fold cross validation varying α, as shown in Figure 16b.

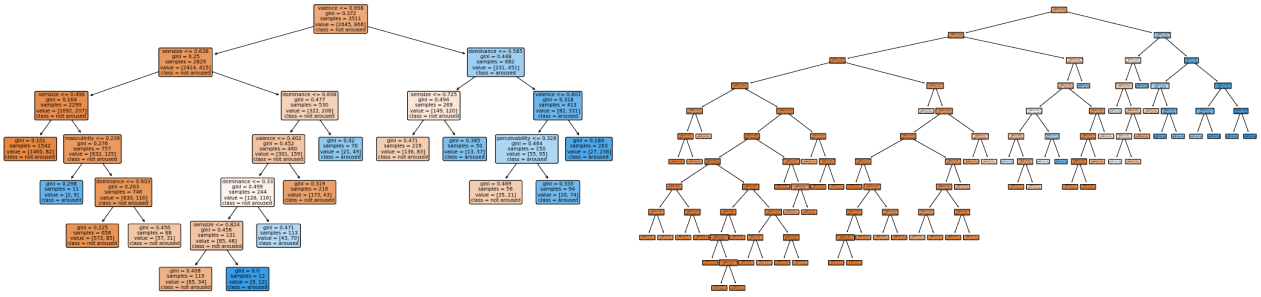


(a) Plot of accuracy vs *alpha* for 1-fold validation

(b) Mean accuracy vs *alpha* for 10-fold validation

Figure 16: Plots for the cost complexity pruning

Based on these results, we choose the α that maximizes the accuracy and we build a new pruned tree using this parameter. As shown in Figure 17a, the new tree is simpler and performs better.



(a) CCP

(b) Grid Search

Figure 17: Pruned Trees

This analysis is performed with both the *gini* and *entropy* impurity measures. The results obtained with this classification algorithm will be discussed in the following sections.

An alternative to this method is a *grid search* for the best parameters (*Max Depth, Min Samples Split, Min Samples Leaf*) using the F1 score as measure. With the grid search, the accuracy and the other test scores obtained are similar to the ones for CCP, but the complexity of the tree is higher, as shown in Figure 17b. For this reason we opted for the cost complexity pruning method, tuning the parameter α.

3.3 Random Forest

Decision Trees are a useful tool for classification because of their simplicity, but they present problems when it comes to classifying a new set. An improvement of this algorithm may be the Random Forest. This method consists in creating different bootstrapped datasets from the training sets and creating for each one a decision tree (more generally, an *estimator*), as described in section 3.2, without tuning the parameters. Then, each record will be evaluated by every tree (a.k.a. by the *forest*) and classified according to the majority of the outputs.

To determine the number of estimators we run different simulations and pick the value with the lowest *Out Of Bag error*, i.e. the misclassification error on the records that did not enter the bootstrapped dataset.

Taking the confusion matrices and the scores table into consideration, the results will be commented in the conclusion section.

3.4 K - Nearest Neighbors

The last classification algorithm we used is K-Nearest Neighbors (KNN). With this method, the target variable is classified by the majority of votes of its k^{th} neighbors. If, for example, the majority of neighbors of a new record belong to class 1, the new item will be classified as 1 as well.

To determine optimal k, we iterate the algorithm for each variable over several values of k and pick the one that maximizes the accuracy. The most accurate value for k depends on the target variable. The results will be commented in the next sections.

3.5 Choice of target variable

The provided dataset is not meant to have a specific target variable. For that reason, we perform the analysis described in the previous sections for every single attribute. We decided to divide them into *binary* and *multi-split* target variables.

Binary target variables are obtained from continuous variables by setting a threshold. The values under the threshold are labeled as 0, while those above are labeled as 1. The thresholds for each variable are chosen according to the last quartile. We report the values of the thresholds in Table 3.

	Arousal	Valence	Dominance	Familiarity	Size	Masculinity	Polysemy	Perceivability	Aoa
Threshold	0.55	0.67	0.57	0.60	0.63	0.60	0.50	0.80	0.60

Table 3: Thresholds used for the binary split

The accuracy, precision, recall and F1 score of the Decision Tree using the *entropy* criterion for each variable are shown in Table 4.

	Arousal	Valence	Dominance	Familiarity	Size	Masculinity	Polysemy	Perceivability	Aoa
Accuracy	0.83	0.87	0.82	0.82	0.80	0.76	0.93	0.81	0.81
Precision (0)	0.85	0.79	0.87	0.71	0.82	0.85	0.93	0.84	0.83
Precision (1)	0.75	0.89	0.66	0.85	0.66	0.52	0.00	0.68	0.79
Precision (wgt mean)	0.86	0.86	0.82	0.82	0.78	0.77	0.87	0.80	0.81
Recall (0)	0.94	0.94	0.89	0.58	0.92	0.82	1.00	0.90	0.89
Recall (1)	0.52	0.67	0.61	0.91	0.44	0.58	0.0	0.55	0.69
Recall (wgt mean)	0.87	0.87	0.82	0.82	0.80	0.76	0.93	0.81	0.81
F1 score (0)	0.89	0.91	0.88	0.64	0.87	0.84	0.96	0.87	0.86
F1 score (1)	0.61	0.73	0.63	0.88	0.53	0.55	0.00	0.61	0.73
F1 score (wgt mean)	0.86	0.86	0.82	0.82	0.78	0.76	0.90	0.80	0.81

Table 4: Score values of DT for binary target variables, whereas the mean is weighted on the support.

Based on these results, we chose to focus our analysis on three variables. The most interesting one appears to be *valence*, given its high scores. In addition, we analyze *age of acquisition* since the comparison with the multi-split analysis may be interesting. *Polysemy* is a variable to considerate because it is already binary.

The scores of Random Forest and KNN for these variables are shown in the Tables 5 and 6, respectively. The number of k used in the KNN algorithm is 6 for *valence*, 6 for *aoa* and 3 for *polysemy*.

Taking also into consideration the confusion matrices, the results will be commented in the conclusions section.

	Valence	Polysemy	Aoa
Accuracy	0.88	0.93	0.82
Precision (0)	0.89	0.93	0.83
Precision (1)	0.85	0.33	0.80
Weighted mean	0.88	0.89	0.82
Recall (0)	0.96	1.0	0.90
Recall (1)	0.67	0.01	0.68
Weighted mean	0.88	0.93	0.82
F1 score (0)	0.92	0.96	0.86
F1 score (1)	0.75	0.02	0.74
Weighted mean	0.87	0.90	0.82

Table 5: Score values of Random Forest

	Valence	Polysemy	Aoa
Accuracy	0.88	0.95	0.83
Precision (0)	0.87	0.96	0.81
Precision (1)	0.92	0.70	0.88
Weighted mean	0.88	0.94	0.84
Recall (0)	0.98	0.99	0.95
Recall (1)	0.58	0.38	0.63
Weighted mean	0.88	0.95	0.83
F1 score (0)	0.92	0.97	0.87
F1 score (1)	0.71	0.49	0.73
Weighted mean	0.87	0.94	0.82

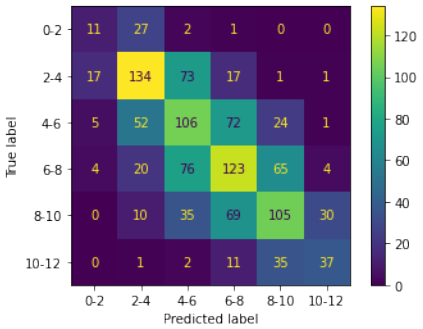
Table 6: Score values of KNN

Multi-split target variables are *web corpus frequency* and *age of acquisition*.

The variable *web corpus frequency* has already been discretized in the section 1.4.2.

The attribute *age of acquisition* can be seen as a discrete variable, truncating the original values. In this way, we obtain a binned feature between 1 and 6, which represents an interval for the age as described in section 1.1.

We report the Confusion Matrix and the test scores directly in Figure 18.

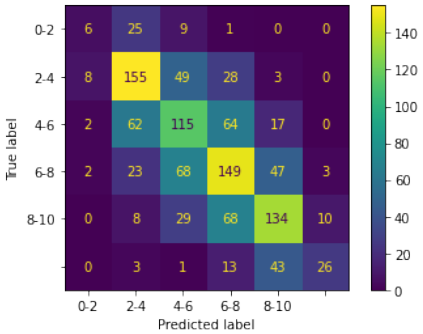


	Precision	Recall	F1-Score	Support
age 0-2	0.30	0.27	0.28	41
age 2-4	0.55	0.55	0.55	243
age 4-6	0.36	0.41	0.38	260
age 6-8	0.42	0.42	0.42	292
age 8-10	0.46	0.42	0.44	249
age 10-12	0.51	0.43	0.47	86
weighted avg	0.44	0.44	0.44	1171
accuracy	0.44	0.44	0.44	0.44

Figure 18: Confusion Matrix and score values for Age of Acquisition

As seen in Figure 18, these results are not really useful. The results for *web corpus frequency* are not reported because they were not found of interest, given their low accuracy.

Theoretically, the KNN algorithm performs well with multi-split target variable. The analysis for *age of acquisition* performed with K=14 is summarized in Figure 19.



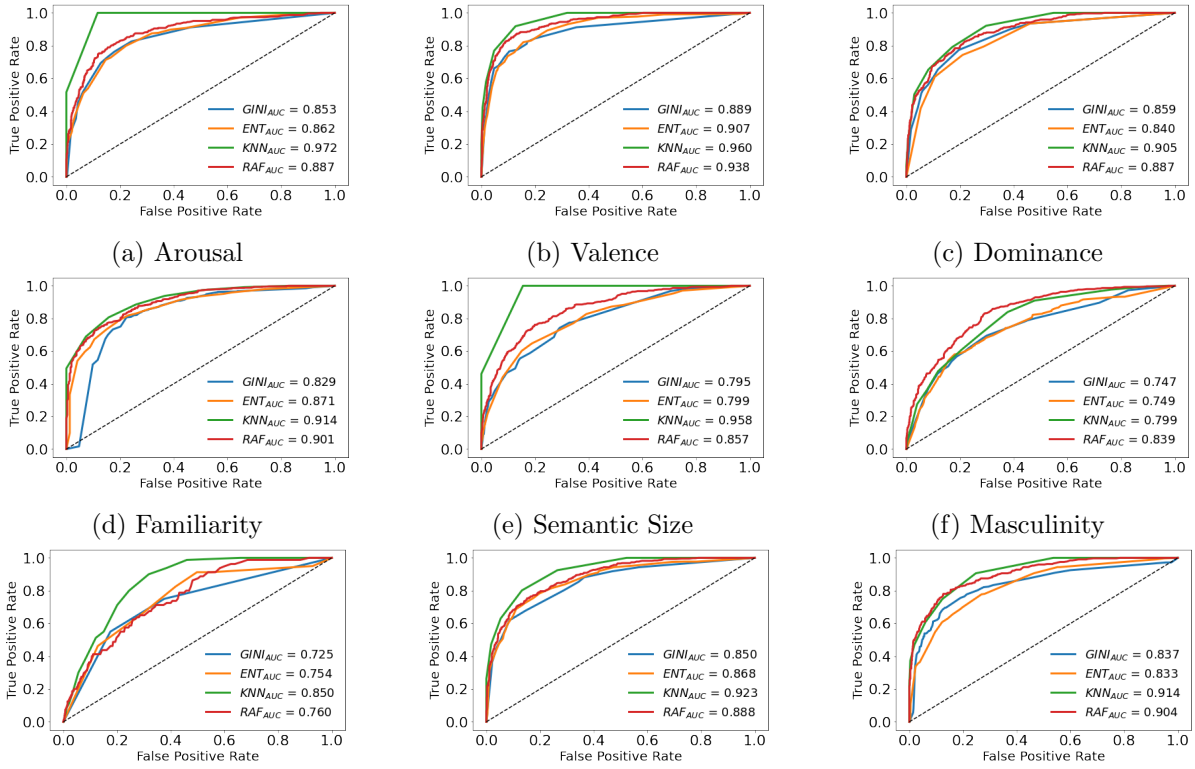
	Precision	Recall	F1-Score	Support
age 0-2	0.33	0.15	0.20	41
age 2-4	0.56	0.64	0.60	243
age 4-6	0.42	0.44	0.43	260
age 6-8	0.46	0.51	0.48	292
age 8-10	0.45	0.54	0.54	249
age 10-12	0.66	0.30	0.42	86
weighted avg	0.50	0.50	0.49	1171
accuracy	0.50	0.50	0.50	0.50

Figure 19: Confusion Matrix and score values for *age of acquisition* multi-split with KNN

As predicted, we can see an improvement with respect to the Decision Tree classification method. However, the scores remain low and for this reason, the multi-target-variable analysis will not be discussed in the conclusions.

3.6 Final Discussion

The ROC and AUC of all the models applied to the binary variable are reported in Figure 20.



(a) Arousal (b) Valence (c) Dominance

(d) Familiarity (e) Semantic Size (f) Masculinity

(g) Polysemy (h) Age of Acquisition (i) Perceivability

Figure 20: ROC and AUC of variables

The identification of the best model is not unique and changes from variable to variable. However, one may extract some general considerations. Regarding Decision Trees, the *entropy* impurity measure generally produces a smaller error than *gini*. However, one can notice that overall KNN and Random Forest perform better than Classification Trees. This behavior is confirmed by the good results with respect to every score (accuracy, precision, recall, F1 score) as shown in Table 6 for the attributes of interest. One must pay attention to the fact that the good looking shape of the curves 20a and 20e for KNN is due to the low number of k, which makes it more exposed to statistical fluctuation.

Another useful tool for the analysis of the variables of interest is given by the Confusion Matrix of Table 7. In this Matrix one can see the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN), as labeled in the Table.

This matrix highlights the fact that KNN can predict some of the few polysemic words, unlike Decision Tree or Random Forest.

The matrix is also helpful for the calculation of sensitivity, specificity and the other scores already mentioned. These are parameters of interest because we could give one more importance than another depending on the variable of consideration and the kind of analysis to perform. In our case, we do not have any particular need, so we look for a trade-off between precision and sensitivity. For this

	Predicted 1	Predicted 0	Predicted 1	Predicted 0	Predicted 1	Predicted 0	
True 1	203	104	0	80	279	159	Decision
True 0	41	823	0	1091	70	663	Tree
True 1	207	100	1	79	299	139	Random
True 0	36	828	2	1080	73	660	Forest
True 1	178	129	30	50	276	162	KNN
True 0	15	849	13	1078	39	694	
	Valence		Polysemy		Age of Acquisition		

Table 7: Confusion Matrices for features of interests according to the different algorithms. Green indicates the True Positives, while Yellow the True Negatives.

reason, the most useful variable for our analysis is the F1 Score, defined as a harmonic mean of the two variables mentioned above.

On one hand, one can infer that the differences between these algorithms as well as for the binary target variables are not particularly significant, because the F1 scores are not that far apart from each other. On the other hand, the differences regarding the multi-split target variables between KNN's and the algorithms' scores are more evident.

4 Pattern Mining

Pattern mining is a process which allows to find out co-occurrences and patterns in our data. First, we preprocessed our variables, as described in the section below. Then, we discovered the frequent itemsets and finally extracted the association rules.

4.1 Preprocessing

In order to apply a pattern discovery algorithm, we first normalized our dataset with the MinMaxScaler. For a better analysis, we chose to divide the range of the variables in four intervals, i.e. we multiplied the normalized values by 3, so that each variable falls between 0 and 4. We then truncate all the variables, rounding them down. Afterwards, we added the name of the feature to its corresponding values. For the only binary feature, *polysemy*, we implemented a dictionary where 0 is *not polysemic* and 1 is *polysemic*. However, after some attempts in the association rules mining, we noticed that the attribute *polysemy* may result in just noise, without contributing to the analysis. This is due to the high frequency of not polysemic words in the dataset, which implies an irrelevance of said variable in the rules. Therefore, we decided to discard *polysemy*.

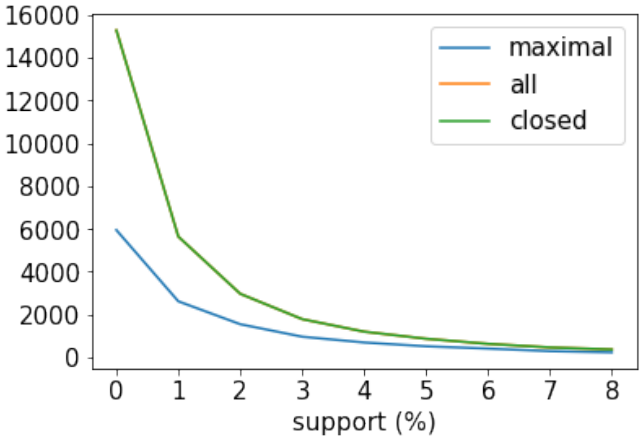
4.2 Frequent Itemsets

To perform the pattern extraction, we applied the *apriori* algorithm, in order to find all the frequent itemsets.

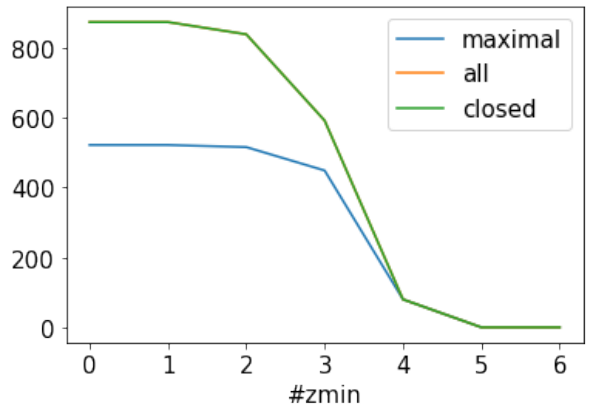
4.2.1 Choice of parameters

The parameters to set for the *apriori algorithm* are the support, the minimum number of items in a set, and the maximum items in a set. We decide not to set a limit for the max parameter. The number of frequent itemsets varying the other 2 parameters are plotted in Figure 21. One can also look at how the plots change considering only the closed and maximal itemsets. Specifically, we noticed that the

green line (closed itemsets) overlaps with the yellow one (all itemsets). We thus deduced that all the itemsets are closed.



(a) Number of frequent itemsets vs support (zmin=1)



(b) Number of frequent itemsets vs minimum number of elements in set (supp=6)

Figure 21: Plots for the parameters of the apriori algorithm (frequent itemsets)

Given the dimensions of the dataset, we decided to set the support parameter to 6.5%. This choice was also due to the fact that the number of itemsets does not decrease significantly after this value. Regarding the minimum number of elements in a set, we decided to leave the variable as loose as possible (zmin=1), because for this specific dataset it does not seem appropriate to us to exclude a priori one-to-one rules. The one-to-one rules that are not relevant will be left out by the confidence or lift threshold.

Confidence	0.50	0.60	0.70	0.80	0.90
Number of rules	743	216	62	26	3
Lift	1.5	1.6	1.7	1.8	2.2
Number of rules	33	22	11	9	2

Table 8: Number of rules by confidence and lift

Word	Target Variable	Counts
Christmas	7.0 Web Corpus Freq	2
Dad	7.0 Web Corpus Freq	6
FALSE	6.0 Web Corpus Freq	3
FALSE	7.0 Web Corpus Freq	1
Mom	7.0 Web Corpus Freq	7
Mum	7.0 Web Corpus Freq	3
Mummy	7.0 Web Corpus Freq	2
TRUE	7.0 Web Corpus Freq	1
TV	7.0 Web Corpus Freq	10
Twitter	7.0 Web Corpus Freq	1
yo-yo	6.0 Web Corpus Freq	6

Table 9: Prediction and counts for missing values

4.3 Association Rules

The Association Rules are extracted using the *apriori algorithm*.

4.3.1 Choice of parameter and rules extraction

Varying the parameters of confidence and lift we obtained different number of rules. First, we vary the confidence parameter in a range between 0.50 and 0.90. Once decided to put the threshold at 75% for the confidence we varied the lift within a range from 1.5 to 2.2. The results are shown in Table 8.

The threshold for the lift was set at 1.6. The obtained rules, ordered by target and lift, are reported in Table 10.

	Antecedent	Target	Lift	Conf	Supp (%)	Supp
1	(0.0_SemSize,)	3.0_Perceivability	2.45	0.76	7.78	363
2	(0.0_Age_of_Acquisition, 7.0_Web_Corpus_Freq)	3.0_Familiarity	2.23	0.93	7.26	339
3	(0.0_Age_of_Acquisition, 2.0_Dominance)	3.0_Familiarity	2.16	0.90	8.53	398
4	(0.0_Age_of_Acquisition, 1.0_Masculinity)	3.0_Familiarity	2.16	0.90	7.46	348
5	(0.0_Age_of_Acquisition, 0.0_Length)	3.0_Familiarity	2.11	0.88	10.00	467
6	(0.0_Age_of_Acquisition,)	3.0_Familiarity	2.08	0.87	14.29	667
7	(0.0_Age_of_Acquisition, 2.0_Valence)	3.0_Familiarity	2.07	0.87	8.78	410
8	(0.0_Age_of_Acquisition, 3.0_Perceivability)	3.0_Familiarity	2.05	0.86	8.27	386
9	(0.0_Age_of_Acquisition, 1.0_Arousal)	3.0_Familiarity	2.03	0.85	6.79	317
10	(3.0_Perceivability, 2.0_Dominance, 1.0_Arousal)	2.0_Valence	1.77	0.90	8.05	376
11	(7.0_Web_Corpus_Freq, 2.0_Dominance, 1.0_Arousal)	2.0_Valence	1.74	0.88	6.75	315
12	(3.0_Perceivability, 3.0_Familiarity, 1.0_Arou...	2.0_Valence	1.68	0.85	6.51	304
13	(3.0_Familiarity, 2.0_Dominance, 1.0_Arousal)	2.0_Valence	1.68	0.85	8.63	403
14	(1.0_SemSize, 2.0_Dominance, 1.0_Arousal)	2.0_Valence	1.67	0.84	8.40	392
15	(3.0_Perceivability, 1.0_Masculinity)	2.0_Valence	1.64	0.83	8.61	402
16	(1.0_Masculinity, 2.0_Dominance, 1.0_Arousal)	2.0_Valence	1.64	0.83	6.62	309
17	(2.0_Dominance, 1.0_Arousal, 2.0_Masculinity)	2.0_Valence	1.63	0.82	9.06	423
18	(3.0_Perceivability, 3.0_Familiarity, 2.0_Domi...	2.0_Valence	1.62	0.82	8.20	383
19	(2.0_Dominance, 1.0_Arousal, 1.0_Length)	2.0_Valence	1.61	0.82	8.80	411
20	(1.0_SemSize, 3.0_Familiarity, 2.0_Dominance)	2.0_Valence	1.61	0.81	6.79	317
21	(3.0_Perceivability, 2.0_Dominance, 1.0_Length)	2.0_Valence	1.60	0.81	6.68	312
22	(3.0_Valence, 2.0_SemSize)	2.0_Dominance	1.69	0.76	6.90	322

Table 10: Association Rules with a confidence≥ 75%, lift≥ 1.6 and support≥ 6.5%

4.3.2 Replacing Missing Values

A useful aspect of the Association Rules is the replacement of missing values with some more accurate predictions. However, in our case, the only feature that has missing values is *web corpus frequency* (Figure 2), which does not appear as a target variable for our rules. One could lower the support, confidence and lift to find new rules with this target. In order to do so, we set the parameters respectively to 2, 50 and 1.3. To decrease the noise, we set a threshold at zmin=2. We then counted how many rules would predict a specific value for the word without a *web corpus frequency* value. The obtained results are shown in Table 9.

For the results with a high count (e.g. TV, Mom, yo-yo) one may replace the predicted value for *web corpus frequency*. However, the lift, support and confidence for the majority of these rules are not quite good.

Trying to tighten up the cuts on the parameters (supp¿4, conf¿60, lift¿1.4), we can find only one rule that predicts the missing value for the words in Table 9. This rule applies only for yo-yo and predicts the same result shown in the table.

4.4 Final Discussion

In the choice of the threshold for the confidence and the lift, we opted for a balance between quantity and quality of the rules.

The obtained rules in Table 10 all have a high lift and the majority of them has the variable *valence* as target. However, the variable *familiarity* is the one with the most interesting rules, because of its

high lift and confidence. We can also notice the negative correlation between *age of acquisition* and *familiarity*: every high value for *familiarity* is predicted by a low value of *age of acquisition*.

Regarding the replacement of missing values, we can only infer with certainty that the word *yo-yo* has a *web corpus frequency* value of 6. Given the fact that the words in Table 9 seem common, the results for the other words appear reasonable. However, these predictions have a low confidence and lift level, so there is no certainty of their accuracy.

5 Conclusions

The Glasgow Norms has resulted to be an interesting dataset to work with.

The application of unsupervised and supervised algorithms has shown some trends. For instance, there is a group of words, belonging to the sixth cluster (from the Hierarchical Clustering algorithm) which are short in length, acquired in early age and very familiar: this is reflected in the fifth association rule *((0.0 Age of Acquisition, 0.0 Length) ⇒ 3.0 Familiarity)* in Table 10. Similarly, the third cluster (from the Hierarchical Clustering algorithm) contains words with high valence, high dominance and high semantic size: an extracted rule (n. 22 in Table 10) implies that words with high valence and semantic size are high in dominance as well.

Valence proves to be a good target variable for classification, with high accuracy, precision, recall and F1-score; at the same time, it is a consequent (meaning target variable) for a large number of extracted rules.

The same is true for the variable *familiarity*, which proves to be an interesting variable also because of its high anti-correlation with *age of acquisition*. One can compare the rules found with the classification tree obtained in Section 3.5 and shown in Figure 22. The variables used to split the tree entirely confirm the association rules found.

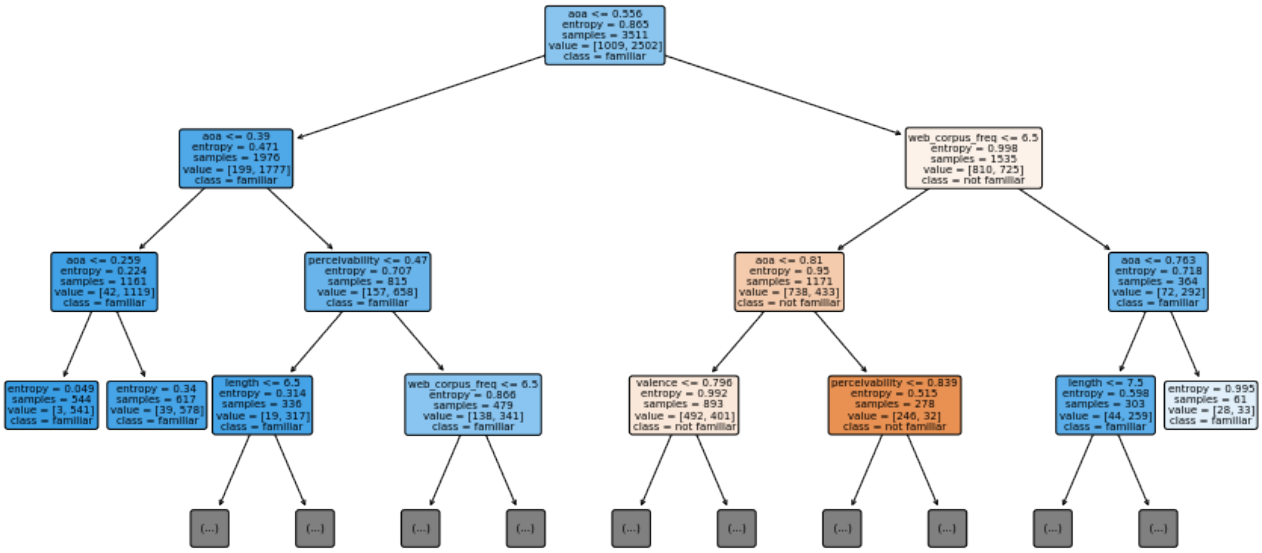


Figure 22: Familiarity Decision Tree

On the other hand, the only binary feature, *polysemy*, was not a satisfying target variable, given the imbalance of its distribution: the large majority of words in the dataset are not polysemic, leading thus to inaccurate or unexpected results during the clustering, classification and pattern mining tasks.

References

[KR90] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley, 1990. ISBN: 978-0-47031680-1.

[San+98] Jörg Sander et al. "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". In: *Data Mining and Knowledge Discovery* 2.2 (June 1998), pp. 169–194. ISSN: 1573-756X. DOI: 10.1023/A:1009745219419. URL: https://doi.org/10.1023/A:1009745219419.

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining.* US ed. Addison Wesley, May 2005. ISBN: 0321321367. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20%5C&path=ASIN/0321321367.

[Ped+11] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.