Automatic Classification of Menu Items

Uppsala University

Aljaž Kovač

May 1, 2022

Abstract

Will write this at the very end.

Acknowledgements

I want to thank...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem definition and overview

Caspeco AB is a company in Uppsala that provides IT services to businesses, mainly from the hospitality industry (restaurants, pubs, etc.). These companies sell food and beverages to their patrons, and therefore have a selection of menu items that they offer. It is established practice in the restaurant industry that the menu has some sort of a structure to it, or in more technical, machine learning (ML) terminology, that the menu items are classified. But while things might look clear-cut when printed on a well-designed menu, the reality behind the curtains can be somewhat blurred. The problem lies in the fact that every restaurant is given the freedom to classify their items as they please. Furthermore, the classification of items within a particular restaurant is not consistent.[1]

It is not hard to imagine the benefits that a consistent and uniform classification would have for a company like Caspeco, which not only provides IT infrastructure but also aims to improve the efficiency of the customers' business through in-depth data analysis. Some of the possible benefits would be: better insight into the sales of items by category; an improved search functionality through the restaurants' stock; an enhanced customer experience when ordering items online,[2] etc.

Since our primary data consists of only item names (names of various types of food and beverages, e.g., 'Pepsi Max', 'Cheeseburger', etc.),[3] we find ourselves in the realm of a specific branch of ML, natural language processing (NLP). NLP deals with the automated analysis of natural language, and its applications are wide and far-reaching: Google Translate, chatbots, sentiment analysis (e.g., determining if a movie review was positive or negative), topic analysis (e.g., determining the main topics in a text), etc. We use a specific tool, designed for NLP applications, called 'sentence embeddings' to find semantically similar menu items.

There are many pre-trained sentence embeddings models available and it can be difficult

[1] To get a better picture of what is meant by inconsistent classification, please refer to subsection 3.2.1.

[2] Imagine that a customer wants to order a Carlsberg beer but the restaurant only stocks Falcon; the latter could be shown as a possible suggestion since they both fall into the category of 'Beer'.

[3] To learn more about the data sets we work with, please refer to subsection 1.2.2.

to know which one to choose for a specific purpose.[4] There are general performance benchmarks published[Reid], but our data is very specific and we can only guess which of the models will perform best with it. The first part of this thesis is therefore devoted to examining the tools at our disposal, i.e., evaluating the quality of the sentence embeddings produced by 3 pre-trained models. The second part of the thesis is then aimed at exploring the possibilities of using these tools for our specific downstream task, i.e., the automatic classification of menu items.

QUESTIONS: Is this style too informal? Does the reader feel engaged or bored by this?

1.2 Background

1.2.1 Sentence embeddings and SBERT

Sentence embedding is a set of techniques used in NLP, by which sentences are mapped to high-dimensional vectors of real numbers. There exist several state-of-the-art sentence encoders, e.g., BERT[Dev+18], RoBERTa, Universal Sentence Encoder[Cer+18], etc. We use Sentence-Transformers (SBERT), a Python framework for state-of-the-art sentence, text and image embeddings[Reig].

SBERT can be used to compute text embeddings for a multitude of languages, and comes with many pre-trained models, specialised for different languages and downstream tasks. Once the embeddings have been computed, they can be compared using cosine similarity to find semantically similar sentences[RG19]. SBERT even offers the option to fine-tune your own models, geared towards a specific downstream task[Reih]. SBERT has been chosen over other sentence encoders for several reasons. Firstly, simplicity of use and a multitude of available pre-trained models. Secondly, SBERT is much faster than other encoders, which makes it more suitable for both unsupervised learning like clustering as well as for semantic similarity search.[5]

Cosine similarity is a measure of similarity between two sequences of numbers. If two sentences are represented as vectors, then the cosine similarity is the cosine of the angle between them[Wik22a]. In the case of SBERT embeddings, cosine similarity is essentially a number in the interval [0, 1], which indicates how similar or dissimilar the two sentences are[Reie]. For example, one of the pre-trained models that we evaluate later in the thesis, the 'sentence-transformers/paraphrase-multilingual-mpnet-base-v2' model, returns the following cosine similarity scores[Reic] (see Table 1.1):

A cosine similarity score of 1 indicates that the sentences are completely equal, whereas a cosine similarity score of 0 indicates that the sentences are completely dissimilar. Cosine similarity can be used to calculate the similarity of various kinds of embeddings, not just sentence embeddings. For example, AirBnb uses an advanced ML model where the home listings are rep-

[4] In ML terminology, downstream task.

[5] The creators of SBERT state that "finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (65 hours) with BERT", whereas SBERT only takes about 5 seconds, "while maintaining the accuracy from BERT"[RG19].

Source sentence	Comparison sentence	Cosine similarity
I am happy.	I am happy.	1.0
I am happy.	You are happy.	0.802
I am happy.	Writing this thesis makes me happy.	0.682
I am happy.	This is just an example.	0.131

Table 1.1: Cosine similarity using embeddings produced by SBERT's paraphrase-multilingual-mpnet-base-v2 model.

Model name	Name abbreviation	Performance embeddings	Performance search	Dimensions
distiluse-base-multilingual--cased-v2	distiluse	60.18	27.35	512
paraphrase-multilingual--miniLM-L12-v2	para-mini	64.25	39.19	384
paraphrase-multilingual--mpnet-base-v2	para-base	65.83	41.68	768

Table 1.2: An overview and comparison of the embeddings produced by three pre-trained multilingual SBERT models.

resented as vectors, which then enables the algorithm to infer the similarity between different listings and improve the website's recommendation and real-time personalisation system[Grb].

Now that we understand what sentence embeddings and cosine similarity are, we need to decide which of the pre-trained SBERT models to use. At the time of writing, SBERT provides 473 pre-trained models, 449 of which can be used for sentence similarity[Reif]. Fortunately, our choice is made easier by our data set, which consists of menu items in 3 different languages (English, Swedish, Spanish). This means that we will have to choose among the available multilingual models. The SBERT documentation website lists 4 such models, one of which does not support Swedish[Reib]. We are therefore left with 3 pre-trained multilingual models. Please refer to Table 1.2 for an overview and comparison[Reid].

The distiluse model[6] is a distilled version of the multilingual Universal Sentence Encoder[Yan+19], whereas the para-mini and the para-base models are multilingual versions of two other SBERT pre-trained models[Reib]. All three models are intended to be used for semantic similarity and clustering. They do, however, have a slightly different architecture and they map words to different numbers of dimensions (see Table 1.2). As mentioned in section 1.1, the first part of this project will be devoted to assessing the performance of the sentence embeddings produced by these models on our very specific data set.

QUESTIONS: Should I add more technical detail about the models or sentence embeddings in general?

[6]For brevity, we will refer to these models by their abbreviated names, see Table 1.2.

Article Name	Article Group Name
Carlsberg Export 50 cl	Öl
Parmesan Garlic Wings 5 pcs	Mat
6 cl Ron Zacapa	Sprit
Personalöl	Tanköl
1/2 Vitt vin	Vin
EXTRA TAVO VEG 1ST	Mat

Table 1.3: A short extract from the raw data set.

Article Name	Article Group Name
carlsberg export	Öl
parmesan garlic wings	Mat
ron zacapa	Sprit
personalöl	Tanköl
vitt vin	Vin
extra tavo veg	Mat

Table 1.4: A short extract from the processed data set.

1.2.2 The data set

The reasons why we have referred to our data set as 'specific' should be understood within the context of the SBERT models. These models were primarily trained to find semantically similar sentences and paragraphs of text. Our input to the models, however, are not really sentences, but rather item names. Furthermore, some of the names are simply made-up and probably cannot be found elsewhere (or at least not necessarily in the context of food and beverages), e.g., 'Kalaspaket', which in our case refers to a selected menu of food that can be pre-ordered for patrons that are celebrating their birthday. Last but not least, plenty of item names include a form of quantity, e.g., 'Zingo 40 cl', 'Hot Buffalo Chicken Wings - 10 PCS, '1/2 Vitt vin', etc.

Due to this last quality of the data set, we are interested if there are any significant differences between using the data set as-is, and a slightly pre-processed data set where the quantities would be removed from the item names and the names would be lower-cased. We therefore work with and compare two data sets throughout this project, the raw one (containing data as provided by Caspeco), and a processed one (containing lower-cased item names with removed quantities).[7] We present short extracts of both data sets below, the raw data in Table 1.3[8] and the processed data in Table 1.4[9]. In Table 1.5 we present a short summary of the relevant value counts for both data sets.

[7]Here we follow the approach taken by the authors of [Lee+17] where they build a classifier to predict the nationality of persons based solely on their names.

[8]Notice the spelling mistake in the last entry of the table, i.e., 'EXTRA TAVO VEG 1 ST' should be 'EXTRA TACO VEG 1 st'.

[9]Notice we don't correct the spelling mistake.

Raw data		
	Item Name	Item label
count	575	575
unique	509	61
Processed data		
count	546	546
unique	394	61

Table 1.5: Value counts for raw and processed data, original labels.

1.2.3 Approaches to the problem

In the first part of the project, where we explore and compare the performance of the embeddings produced by the 3 chosen models on our data set, we treat the problem as an unsupervised learning problem. Unsupervised learning in ML refers to the set of algorithms that attempt to distinguish patterns from unlabelled data[IBMd]. Technically, our data set is labelled but since the classification is greatly inconsistent we cannot rely on it as the ground truth[10] and therefore choose to ignore it. We run two classical clustering algorithms, K-Means and hierarchical or agglomerative clustering. We observe the clusters, the quality of the clusters, and try to draw conclusions regarding the performance of the embeddings.

In the second part of the project, where we attempt to automatically classify the items, we treat the problem as a supervised learning problem. In supervised learning we use labelled data sets to train algorithms how to classify data or predict outcomes[IBMc]. To this purpose, we need to label our data set manually.[11] We present a short extract from the labelled raw data set in Table 1.6.

Article Name	Article Group Name
Carlsberg Export 50 cl	Beer
Parmesan Garlic Wings 5 pcs	Wings
6 cl Ron Zacapa	Spirits
Personalöl	Beer
1/2 Vitt vin	Wine
EXTRA TAVO VEG 1ST	VariedFood

Table 1.6: A short extract from the labelled, raw data set.

1.2.4 Metrics

In the first part of the project, where we compare the performance of the sentence embeddings produced by 3 different pre-trained SBERT models after running K-Means and hierarchical clustering algorithms on unlabelled data, we use 3 classical metrics to measure the quality of the clusters produced. The 3 metrics used are the Silhouette Coefficient, the Calinski-Harabasz Index, and the Davies-Bouldin Index. They are all used when the ground truth labels are not known[leaa].

[10] The term 'ground truth' in ML generally refers to the ideal expected result, which usually depends on the purpose of the model or the downstream task the model is used for[Koz].

[11] In subsection 3.2.1 we discuss the process of labelling the data set.

The Silhouette Coefficient for a set of samples is the mean of the Silhouette Coefficient for each sample[leaa]. The Silhouette Coefficient is bounded within $[-1, 1]$, where a score close to -1 would indicate highly incorrect clustering, and a score close to 1 would indicate highly dense clustering. A score around 0 indicates overlapping clusters. It is important to note that several metrics can be used to calculate the intra-cluster distance and the mean nearest-cluster distance for each sample[leaa]. We use the cosine metric (instead of Euclidean, for example) because it is more common to use cosine when working with text data and high-dimensional vectors[Emm].

The Calinski-Harabsz Index represents the ratio of the sum of the dispersion between clusters and the dispersion within clusters (dispersion is defined as the sum of distances squared)[leaa]. The Calinski-Harabasz Index is higher for dense and well-defined clusters, and lower for sparse and overlapping clusters.

The Davies-Bouldin Index is another commonly-used metric for the evaluation of cluster quality. It reflects the average similarity between clusters, similarity being defined as a measure that compares the distance between clusters with the size of the clusters[leaa]. It is important to note that the lowest possible score of the Davies-Bouldin Index is 0. A score close to 0 would therefore indicate well-separated clusters.

These 3 metrics are used in the first part of the project, where we treat the problem as an unsupervised learning problem and use unlabelled data. In addition to these metrics, we also do a visual, qualitative inspection of the clusters and try and form conclusions and assess their quality. It is standard practice when evaluating the performance of a clustering algorithm that a quantitative analysis is followed by a qualitative analysis to determine whether the results are meaningful[Zucb].

In the second part of the project, where we use our own manual classification as the ground truth and treat the problem as a supervised learning problem, we use our own custom metric to measure the quality of the 'clusters' (essentially classes or groups of items) that we create. For more on that, please refer to subsection 3.2.2.

QUESTIONS: Should I myself give more mathematical details about the metrics or does referring to the literature suffice?

1.2.5 Delimitations

There is one concession we had to make during this project. Ideally, the problem of automatic classification of menu items could be treated as a semi-supervised learning problem[12] where only a few data labels would be needed. This would only be possible if the sentence embeddings would be specifically geared towards this downstream task. Then simply running a clustering algorithm like K-means would perhaps produce semantically meaningful and well-separated clusters, and the classification of every item in that cluster could be inferred from the few that have been labelled. In that regard, a possible approach to solving this problem would be to train or fine-tune our own model that would be specific to the menu data we have. However, training or fine-tuning already

[12]This approach, as the name suggests, sits between supervised and unsupervised learning: a small amount of labelled data is combined with a large amount of unlabelled data[IBMa].

pre-trained SBERT models requires pair-wise data that is labelled for semantic similarity. We would need to devise a system for scoring the similarity between items in a meaningful way, and we would need millions of input examples to achieve competitive performance[Reia]. A few such examples are presented in Table 1.7:

Sentence 1	Sentence 2	Similarity score
Falcon Export	Carlsberg Export	0.9
Falcon Export	Tea	0.6
Falcon Export	Nachos	0.3

Table 1.7: A few possible examples of pair-wise training data.

There is the option of using Augmented SBERT[Tha+20], which is intended for smaller data sets, but we would still need to prepare several thousands of such input examples. We would then also need to experiment with various loss functions and evaluate their performance. Unfortunately, this could not be achieved in the allocated time. It could, however, provide a possible direction for future work with this data.

Chapter 2

Clustering and evaluating the quality of the embeddings

2.1 An overview

In this section we examine and compare the performance of sentence embeddings produced by 3 multilingual pre-trained SBERT models on our raw and processed data. We run two clustering algorithms, the K-means and the agglomerative (hierarchical) clustering algorithm, visually inspect the clusters and measure their quality using 3 classical metrics, the Silhouette Score, the Calinski-Harabasz Index and the Davies-Bouldin Index. We aim to see which of the models performs best with our data and how the performances compare on the raw and the processed data sets.

2.2 K-means clustering

2.2.1 Background

K-means clustering is a popular and easy-to-understand clustering algorithm. It aims to separate the data samples into n-groups of equal variance. In doing so, it minimises what is known as 'within-cluster sum-of-squares'[leae]. The algorithm has 3 steps: in the first step, the initial centroids are chosen randomly. Then the algorithm loops between steps 2 and 3, which consist of assigning each sample to its nearest centroid, and creating new centroids by recalculating the mean value of all the samples assigned to each previous centroid.[1]

The K-means algorithm is simple and efficient but it does have potential drawbacks. One of them is that the measure of how internally coherent the clusters are (also called 'inertia') works well with convex clusters, but poorly with irregularly-shaped clusters. Another potential drawback is that inertia is not normalised, and can therefore suffer from the 'curse of dimensionality'[Kar] in high-dimensional spaces[leae].

[1] For an easy-to-grasp visualisation, please refer to StatQuest's explanation[Staa].

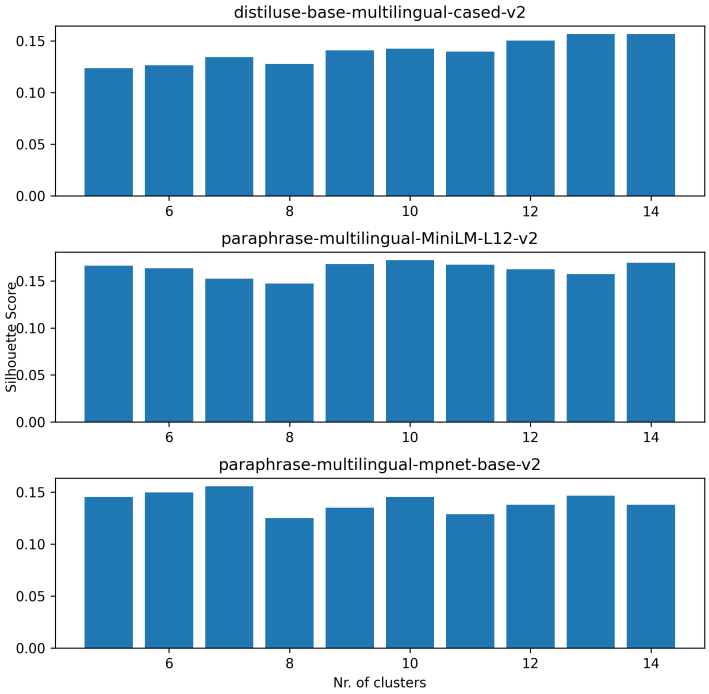

Figure 2.1: The Silhouette Score for all sets of embeddings after K-means on raw data.

2.2.2 Quantitative results

We use Scikit-learn's implementation of K-means. We want to gauge the quality and the contents of the clusters for various numbers of clusters, so we run K-means on both raw and processed data with the number of clusters in the range of $[5, 14]$.[2] We then calculate the Silhouette Score, the Calinski-Harabasz Index and the Davies-Bouldin Index.

The Silhouette Score for raw data is presented in Figure 2.1, and for the processed data in Figure 2.2. We can see that the para-mini and para-base models perform slightly better than the distiluse model, but the differences are negligible. The scores are also close to 0, which would indicate overlapping clusters.

The Calinski-Harabasz Index is presented in Figure 2.3 (raw data) and Figure 2.4 (processed data). We see that the para-mini and the para-base models outperform the distiluse model both with the raw and processed data, however, the differences are again small.

The results for the Davies-Bouldin Index are presented in Figure 2.5 (raw data) and Figure 2.6 (processed data). There seem to be no significant differences in the performance of the models, both on raw and processed data.

Although the para-mini and the para-base model perform slightly better according to the Silhouette Score and the Calinski-Harabasz Index, none of the models outperforms the other two in a significant way. The comparison of the raw and the processed data tells a similar tale: the raw data seems to have a slight edge over the processed data, but the differences are again too small to draw any definitive conclusions.

[2]The reasons for choosing this particular range will become clear later, in the second part of the project; in a nutshell, after inspecting the data, we realised that we would like to have between 5 and 14 groups of items.

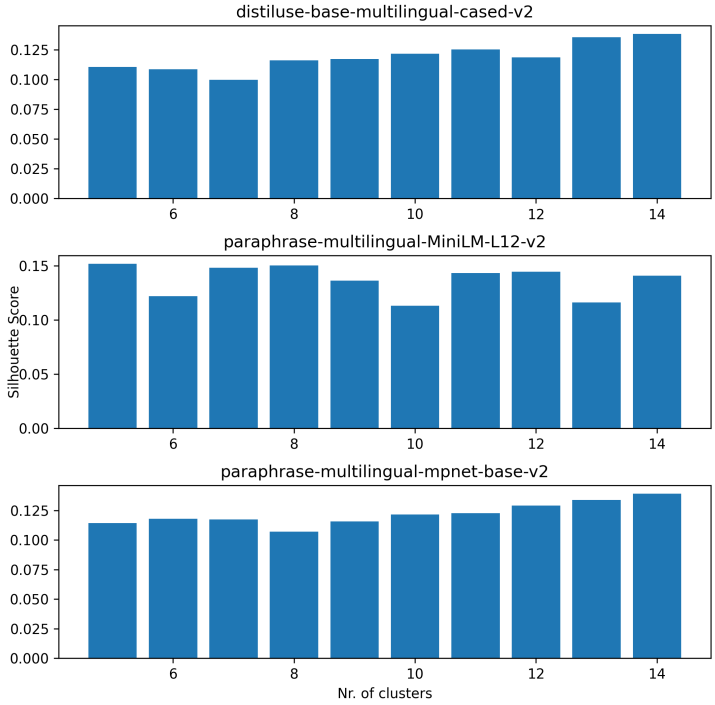

Figure 2.2: The Silhouette Score for all sets of embeddings after K-means on processed data.

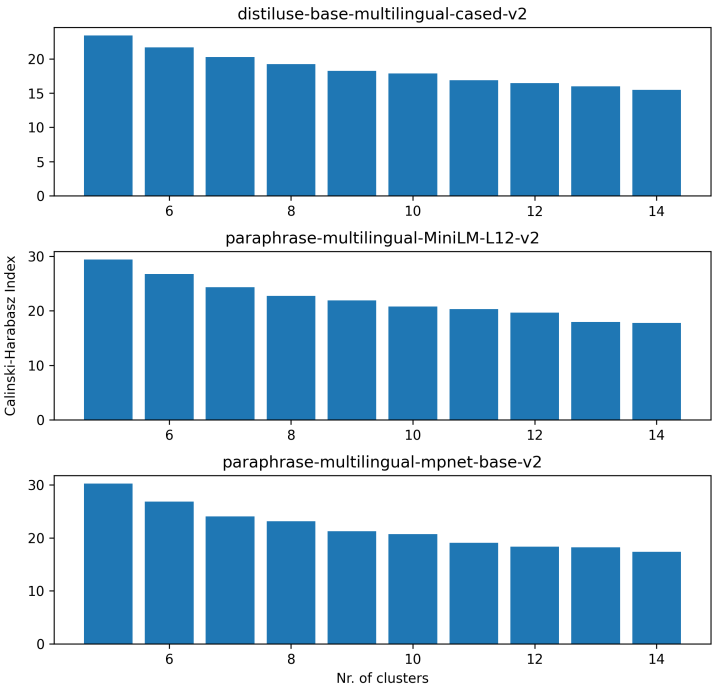

Figure 2.3: The Calinski-Harabasz Index for all sets of embeddings after K-means on raw data.

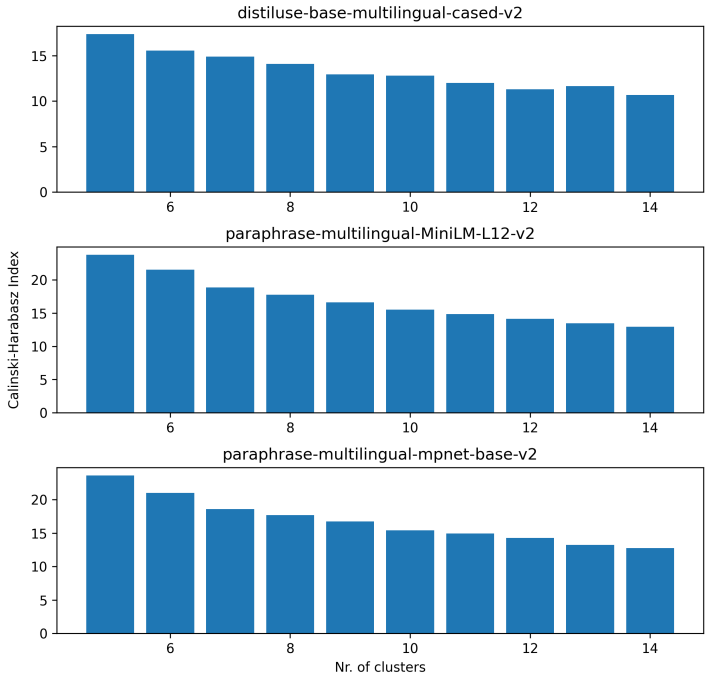

Figure 2.4: The Calinski-Harabasz Index for all sets of embeddings after K-means on processed data.

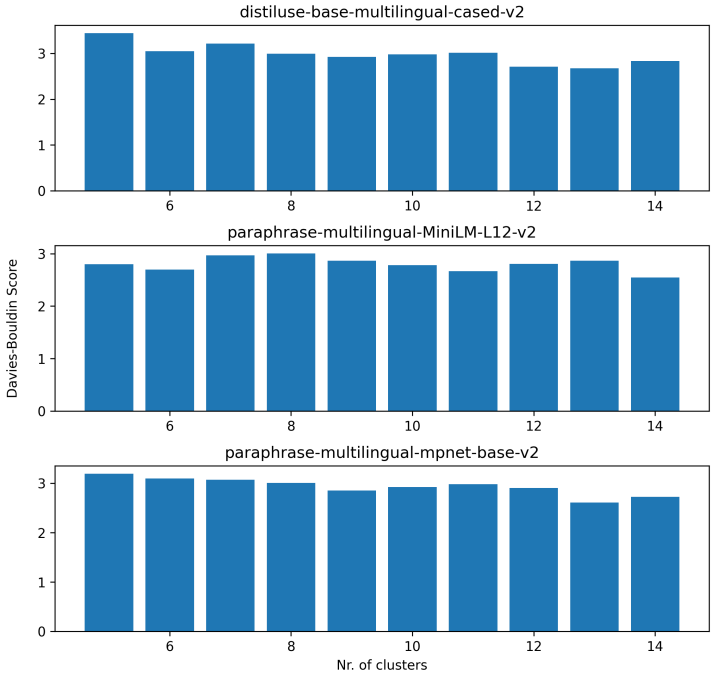

Figure 2.5: The Davies-Bouldin Index for all sets of embeddings after K-means on raw data.

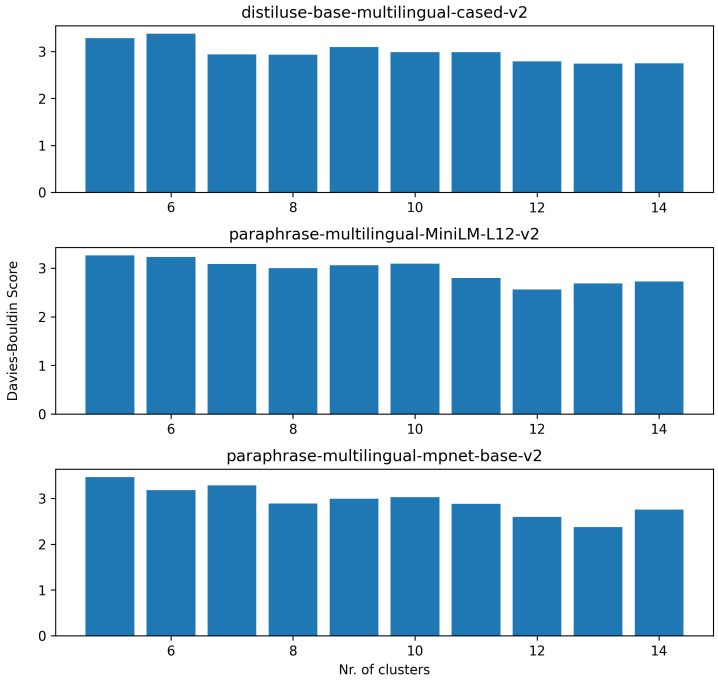

Figure 2.6: The Davies-Bouldin Index for all sets of embeddings after K-means on processed data.

2.2.3 Qualitative results

For a visual inspection of the contents of the clusters, the cluster assignments are printed.[3] A visual inspection is much different to inspecting the clusters with the aforementioned metrics. Such an inspection infers a certain semantic partitioning. What do we expect the clusters to contain? That is the question that determines the outcome of the analysis. Let us say that we would divide our data set into 5 categories: 'Wine', 'Beer', 'Food', 'Coffee', 'Other'. If we judge the clusters from that viewpoint, what labels do we see in each cluster?

We quantify the results of our qualitative inspection in Table 2.1 (distiluse), Table 2.2 (para-mini) and Table 2.3 (para-base). We define a cluster as having no overlap if items of only one category appear in it; with medium overlap we mean clusters with items of no more than 2 categories; and with great overlap clusters with items of 3 or more categories.

A visual, qualitative inspection confirms what the metrics tell us: certain clusters are coherent and semantically meaningful, but in general there is a significant deal of overlap between the clusters and there are no great differences between the various models and the raw and processed data.

2.2.4 PCA and cluster visualisation

To get a better intuition of what the clusters look like in 3-dimensional space, we perform Principal Component Analysis (PCA). PCA is a technique used to identify the orthogonal components of a data set that capture the maximum amount of the variance in the data[leah]. Once we have

[3] In the Appendix we present the 5-cluster partitions for the distiluse embeddings, both raw (Appendix A) and processed data (Appendix B).

	Raw data	Processed data
Cluster number	Categories present	Categories present
1	Food	Other, Wine
2	Beer, Wine, Other	Beer, Food, Wine
3	Wine, Beer, Food, Coffee, Other	Beer, Other, Food, Wine
4	Other, Wine	Coffee, Other
5	Food, Other, Wine, Beer	Food
Degree of overlap	Nr. of clusters with degree of overlap	Nr. of clusters with degree of overlap
None	1	1
Medium	1	2
Large	3	2

Table 2.1: A qualitative inspection of clusters after K-means, distiluse model.

	Raw data	Processed data
Cluster number	Categories present	Categories present
1	Other, Coffee, Wine	Beer, Other, Food
2	Beer, Food, Wine, Other	Food, Other
3	Beer	Coffee, Other
4	Food, Beer, Wine	Beer, Other, Food, Wine
5	Food	Other, Food, Wine, Beer
Degree of overlap	Nr. of clusters with degree of overlap	Nr. of clusters with degree of overlap
None	2	0
Medium	0	2
Large	3	3

Table 2.2: A qualitative inspection of clusters after K-means, para-mini model.

	Raw data	Processed data
Cluster number	Categories present	Categories present
1	Beer, Other, Food, Wine	Food
2	Beer, Other, Wine	Beer, Wine
3	Other, Beer, Coffee	Coffee, Beer, Other, Food
4	Wine, Food, Other	Beer, Other, Wine
5	Food	Other, Food, Wine
Degree of overlap	Nr. of clusters with degree of overlap	Nr. of clusters with degree of overlap
None	1	1
Medium	0	1
Large	4	3

Table 2.3: A qualitative inspection of clusters after K-means, para-base model.

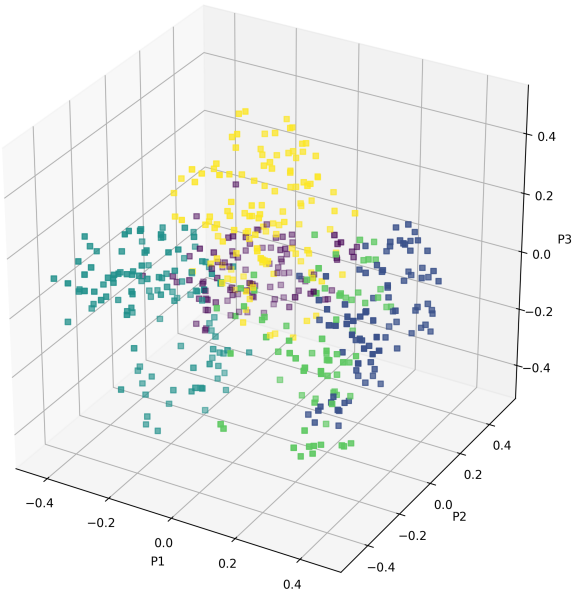

Figure 2.7: 5 clusters after K-means, distiluse model, raw data.

identified the three most prominent principal components, we can reduce the dimension of the data set to those 3 components and visualise the clusters.

We present a visualisation of the 5 clusters after K-means on the raw data using sentence embeddings produced by the distiluse model in Figure 2.7, and on processed data in Figure 2.8. For the para-mini model, raw data, the 5 clusters after K-means are shown in Figure 2.9, and for the processed data, please refer to Figure 2.10. For the para-base model, raw data please refer to Figure 2.11, and for the processed data please see Figure 2.12.

A visualisation confirms our previous findings: the clusters are not well separated and there seem to be no significant differences between the models nor do there seem to be any noteworthy differences between the models' performance on raw as opposed to processed data.

2.3 Hierarchical or agglomerative clustering

2.3.1 Background

Hierarchical clustering refers to the family of clustering algorithms that build nested clusters. This is done by either merging or splitting the clusters successively. It is possible to run hierarchical clustering by specifying the number of clusters, or by allowing the algorithm to find as many clusters as possible. We run hierarchical clustering in both ways. The splitting and merging of the clusters can be done with various linkage criteria. When limiting the number of clusters, we run the algorithm with two of the four possible linkages: 'ward' linkage, which minimises the sum of squared differences within all clusters; and 'average' linkage (with cosine similarity as the metric), which minimises the average of the distances between all observations of pairs of clusters[lead].

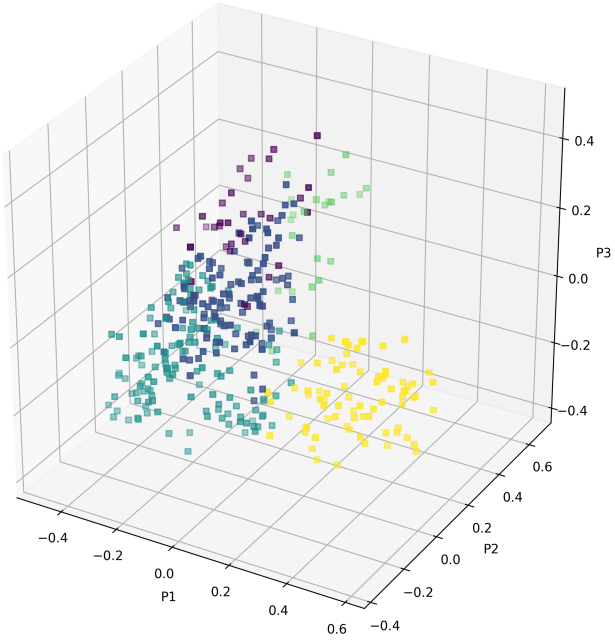

Figure 2.8: 5 clusters after K-means, distiluse model, processed data.

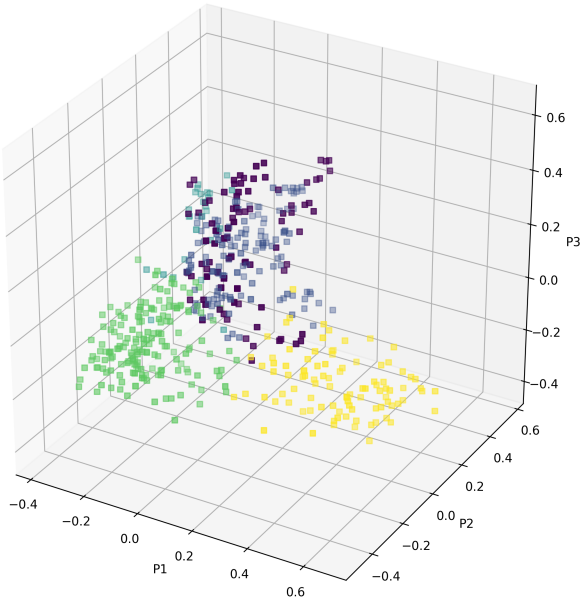

Figure 2.9: 5 clusters after K-means, para-mini model, raw data.

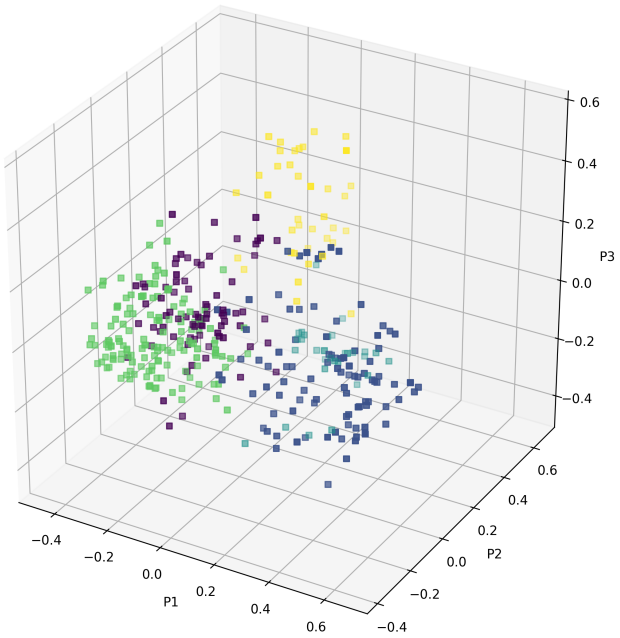

Figure 2.10: 5 clusters after K-means, para-mini model, processed data.

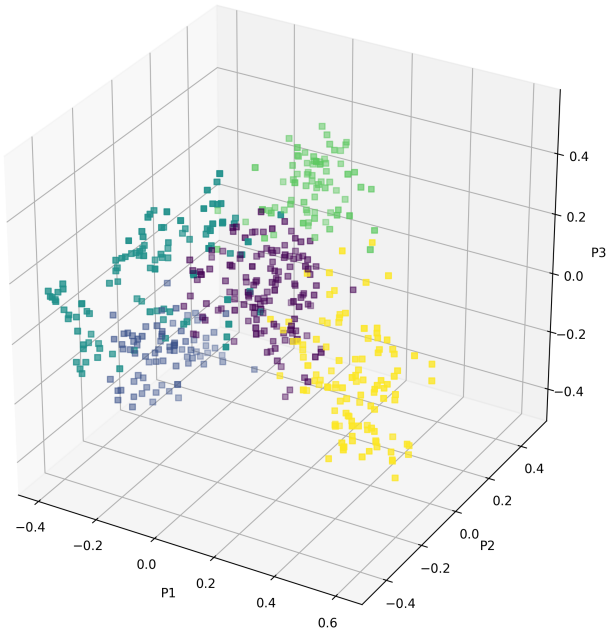

Figure 2.11: 5 clusters after K-means, para-base model, raw data.

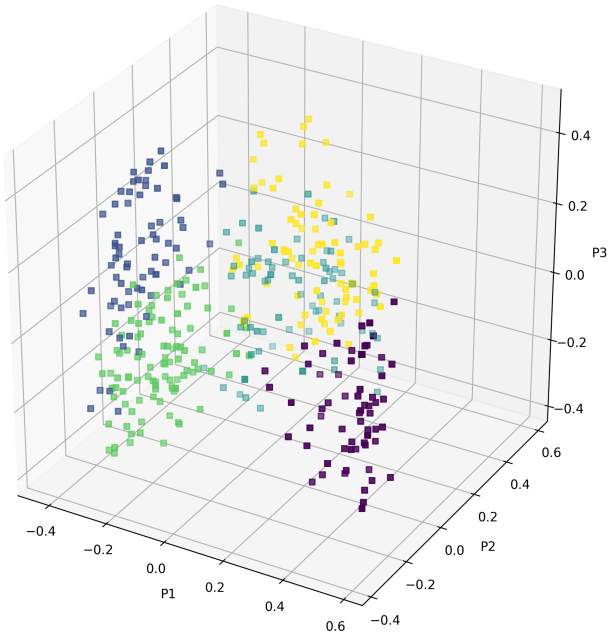

Figure 2.12: 5 clusters after K-means, para-base model, processed data.

2.3.2 Results

We begin by presenting the results of hierarchical clustering with no limitation on the number of clusters. In Figure 2.13 we present and compare the numbers of clusters found for the 3 different models. We see that the embeddings of the distiluse model find the highest number of clusters for both raw and processed data, but the differences are small.

The Silhouette Score, shown in Figure 2.14, shows that the models perform similarly, on both raw and processed data. The Calinski-Harabasz Index, presented in Figure 2.15 is also very similar for both the raw and processed data and for all 3 models. The Davies-Bouldin Index in Figure 2.16 tells a similar tale, and there are no significant differences to be observed between the models or the raw and processed data.

We now run hierarchical clustering while limiting the number of clusters in the same range as we did with K-means, i.e., $[5, 14]$. We run this twice, first with average linkage and cosine metric, and then with ward linkage and Euclidean metric. We then calculate the scores.

The Silhouette Score for the average linkage on raw data is presented in Figure 2.17. The Calinski-Harabasz Index, average linkage, on raw data is presented in Figure 2.18 and the Davies-Bouldin Index, average linkage, raw data in Figure 2.19. The para-mini model outperforms the other two, according to the Silhouette Score and the Calinski-Harabasz Index, but only slightly.

The results for the average linkage on processed data are given in Figures 2.20 (Silhouette Score), 2.21 (Calinski-Harabasz Index) and 2.22 (Davies-Bouldin Index). The only noticeable difference between the models is in the Calinski-Harabasz Index, where the distiluse model again seems to perform worse than the other two, but only by a small margin.

The ward linkage does not improve on this result and we therefore choose to omit the details. The results of the qualitative inspection, performed under the same conditions as with

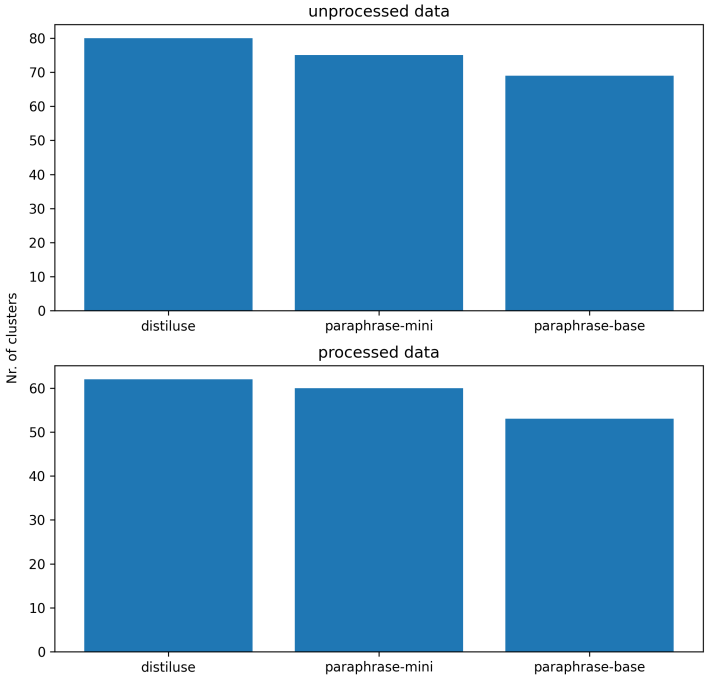

Figure 2.13: The number of clusters formed for every set of embeddings, hierarchical clustering, unlimited nr. of clusters.

K-means clustering (see subsection 2.2.3), are in agreement with the quantitative results. There are some coherent clusters, but there is significant overlap and no noticeable differences between the 3 models. As these findings give no significant new information, we decide to omit the details.

QUESTIONS: Are there too many figures here? Should I present the results in a more concise manner?

2.4 Evaluation conclusions

We have run both the K-means and the hierarchical clustering algorithms on the raw and processed data sets. We have measured and compared the quality of the clusters by calculating 3 classical metrics, the Silhouette Score, the Calinski-Harabasz Index and the Davies-Bouldin Index. In addition, we have performed a visual, qualitative inspection of the clusters formed. This was done so that we could understand how sentence embeddings work when used in unsupervised learning, and to explore if any of the three pre-trained multilingual SBERT models would show outstanding performance on our specific data.

As to the first part of the question, we did gain some insight into how sentence embeddings work when used in an unsupervised learning setting. The visual, qualitative inspection showed that certain item names get clustered together nicely (see Appendix A and Appendix B). We did not, however, manage to find a model of sentence embeddings that stands out in terms of performance. This means that we will need to continue to use and compare all 3 of them even in the second part of this project.

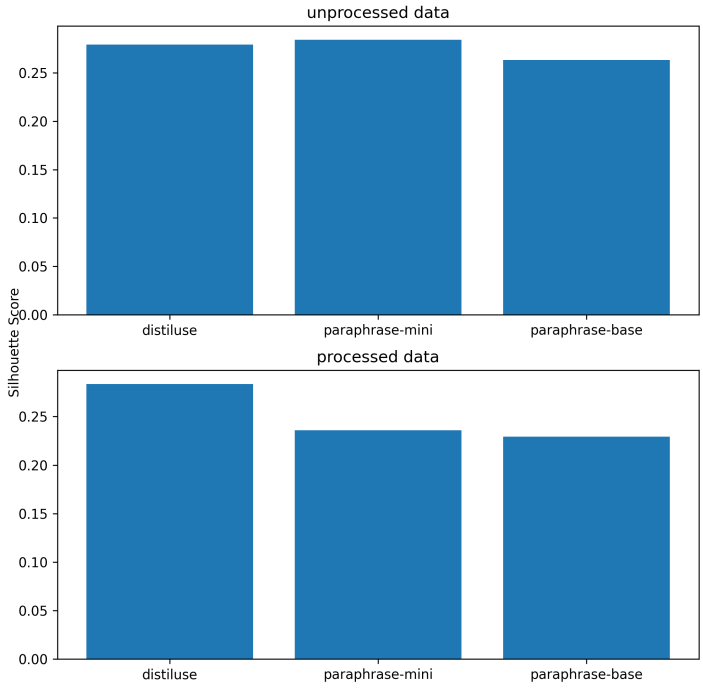

Figure 2.14: The Silhouette Score for every set of embeddings, hierarchical clustering, unlimited nr. of clusters.

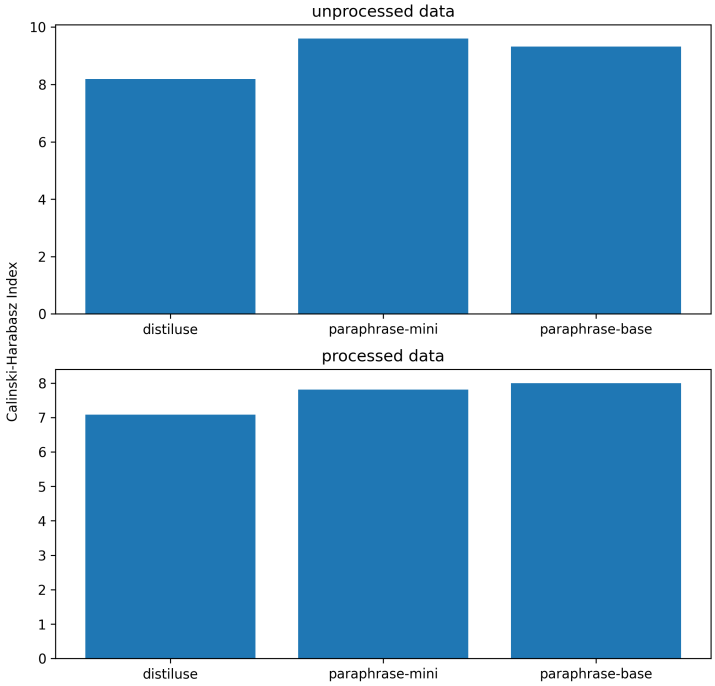

Figure 2.15: The Calinski-Harabasz Index for every set of embeddings, hierarchical clustering, unlimited nr. of clusters.

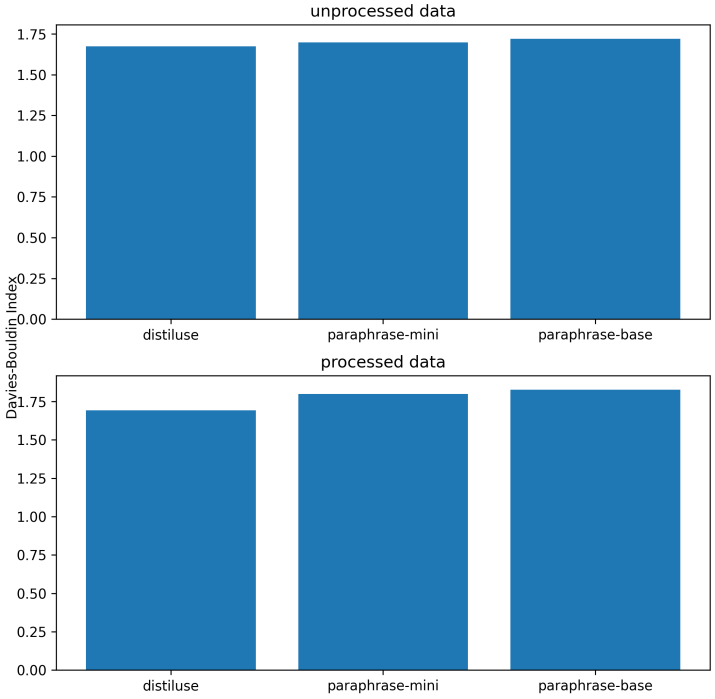

Figure 2.16: The Davies-Bouldin Index for every set of embeddings, hierarchical clustering, unlimited nr. of clusters.

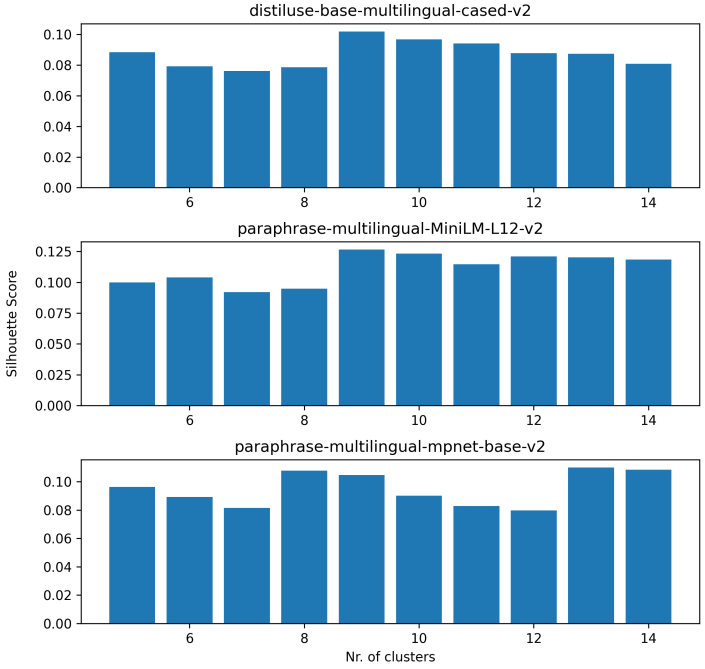

Figure 2.17: The Silhouette Score for every set of embeddings, hierarchical clustering, limited nr. of clusters, average linkage, raw data.

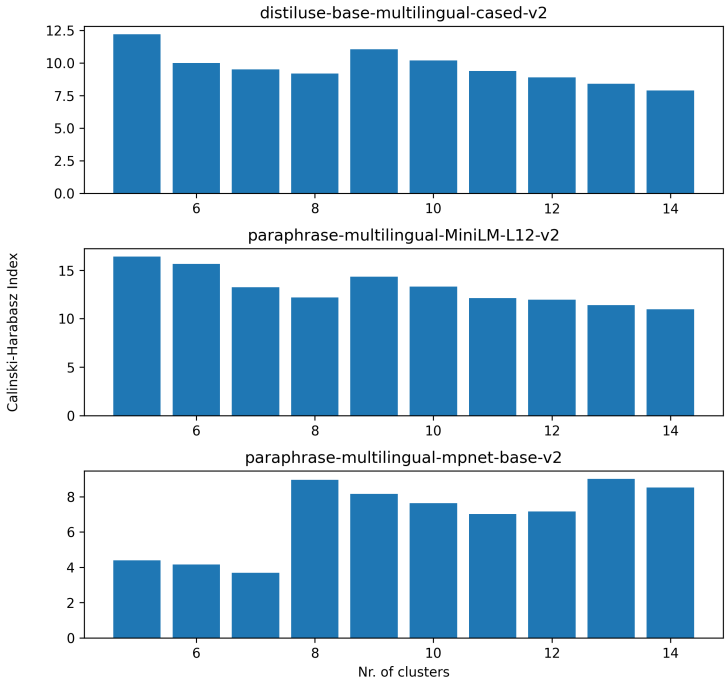

Figure 2.18: The Calinski-Harabasz Index for every set of embeddings, hierarchical clustering, limited nr. of clusters, average linkage, raw data.

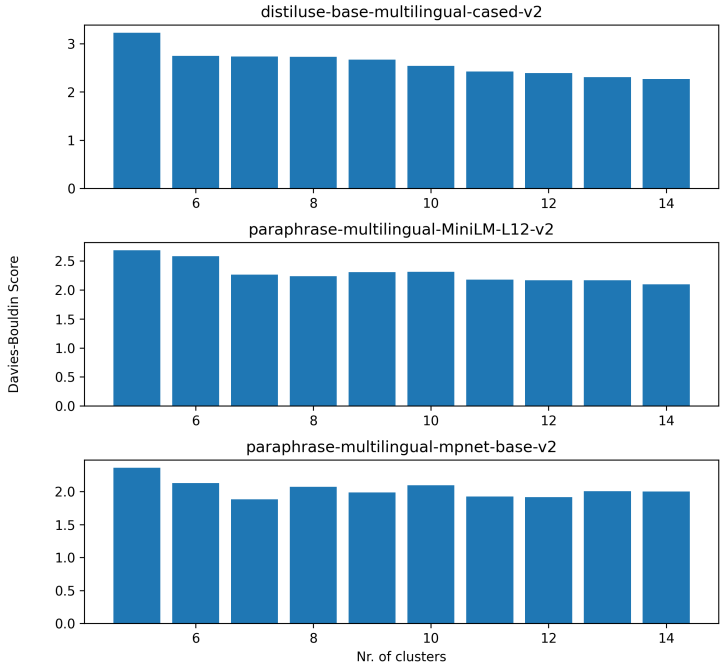

Figure 2.19: The Davies-Bouldin Index for every set of embeddings, hierarchical clustering, limited nr. of clusters, average linkage, raw data.

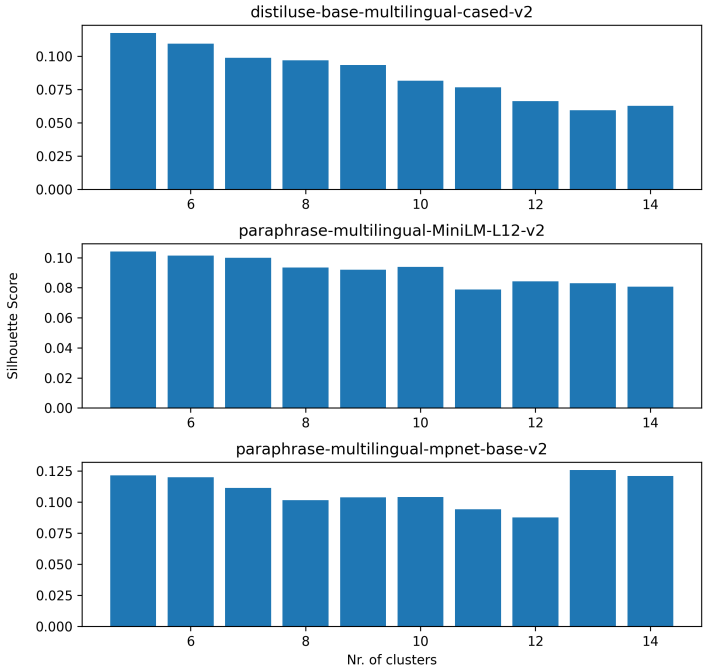

Figure 2.20: The Silhouette Score for every set of embeddings, hierarchical clustering, limited nr. of clusters, average linkage, processed data.

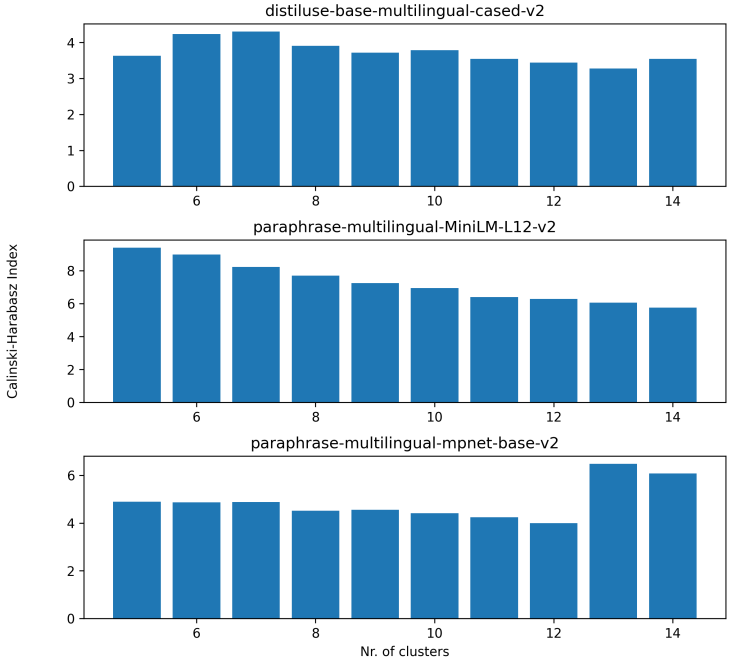

Figure 2.21: The Calinski-Harabasz Index for every set of embeddings, hierarchical clustering, limited nr. of clusters, average linkage, processed data.

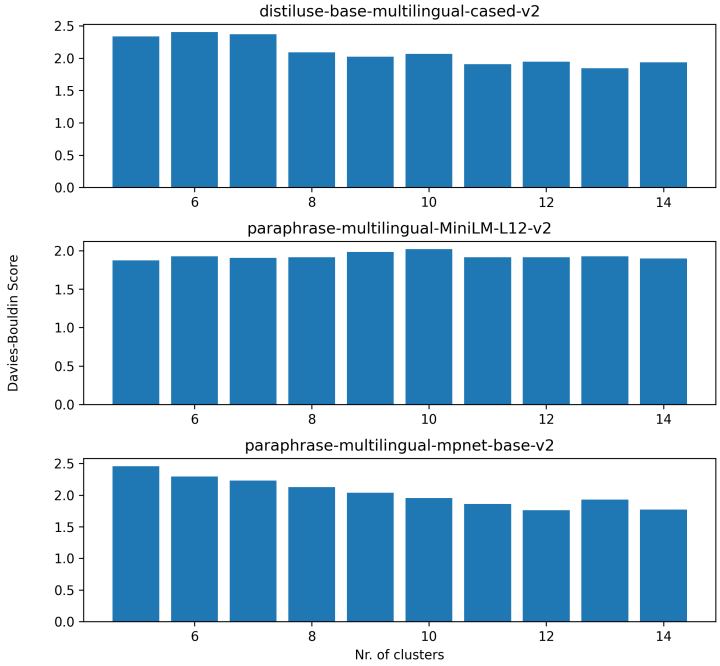

Figure 2.22: The Davies-Bouldin Index for every set of embeddings, hierarchical clustering, limited nr. of clusters, average linkage, processed data.

Chapter 3

Using the embeddings for classification of menu items

3.1 An overview

In this chapter, which represents the second part of the thesis, we use the sentence embeddings produced by 3 multilingual pre-trained SBERT models, to try and classify the menu items into meaningful groups. To this end, we need to first manually label the data, and then use those labels as the ground truth that we train our classifier on. We develop our own custom fit and predict methods, as well as implement the classical K-nearest neighbours (k-NN) algorithm. We present and compare the results and draw conclusions.

3.2 Manual classification

In this section we transition from unsupervised learning to supervised learning. We discuss the process and challenges of manually labelling the data, and we present a custom metric intended to measure the quality of the 'clusters' or groups that we have formed. This section is therefore a bridge between the first part of the project, where we compare the embeddings produced by the 3 chosen models, and the second part of the project where we attempt to use the embeddings for automatic classification of menu items.

3.2.1 Labelling the data

As mentioned before, the data set, which represents the menu items from a restaurant in Uppsala, is labelled but the classification is inconsistent. What do we mean by that? Sometimes the same article is labelled into several different groups, e.g., 'Pripps Blå'[1] is first categorised as 'Vatten'[2] but a later entry has the label 'Alkoholfritt'.[3] Furthermore, as this very same example shows,

[1] A type of light beer.
[2] Meaning 'water' in Swedish.
[3] Meaning 'alcohol-free' in Swedish.

	Raw	Processed
Label	Nr. of items	Nr. of items
Mat	72	71
Sprit	61	56
Vin	58	58
Fatöl Starköl	43	32
Öl	30	27
Diverse Mat	30	30
Syrup	15	11
Vatten/läsk	13	13
Kaffe	13	13
BURGERS/SANDWICH	10	10
Kids	8	8
Vitt Vin	6	6
Cognac/Calvados/Grappa	6	6
Bowling	1	1

Table 3.1: Value counts by category for raw and processed data, original labels, extract.

certain items are simply misclassified. Then there is the question of the granularity of classes. How large or small do we want the classes to be, and how semantically precise or narrow would we like them to be? The data set contains 61 unique categories, with varying levels of finesse and degrees of overlapping. We show an extract from the list of original labels along with their value counts in Table 3.1.[4]

Some of the questions we therefore had to answer were: What categories do we want? What categories do we recognise in the data set? What level of granularity of the categories do we want? The answers were not black-or-white, and it is important to note that the classification that we present here is only one of the many possible we could think of. Our goal was to maintain a level of granularity that the restaurant could be pleased with, while trying to unify some of the overlapping categories. Another goal was to try and balance the categories in terms of value counts, for the later purpose of training the model on this data. After a lengthy consideration 14 categories were identified. We present them, along with their value counts, in Table 3.2.[4]

After the categories have been identified, there was the difficult and often impossible task of classifying the items into these established groups. It was of utmost importance that we keep the classification uniform, i.e., we don't classify extremely similar items into different groups. For example, should 'Tullammore Irish Coffee'[5] and 'Jamaican Coffee'[6] fall under 'Spirits' or the group called 'CoffeeTeaMilk'? Should we perhaps have a separate, specific group for this type of drink, called 'Coffee Cocktails'? In the end, we decided to place all items that contain coffee in the group 'CoffeeTeaMilk'. This group is furthermore an example of our attempt to not have groups with only one or two items, which is what would have happened had we also had two

[4]The discrepancies in the counts are the result of removing duplicate items; since the processed items do not contain quantities after their names, there were more duplicates in that data set that were removed.

[5]A special kind of drink, consisting of 2 parts Irish whiskey, 4 parts coffee, 1 and 1/2 parts fresh cream and 1 tsp brown sugar.

[6]A similar drink to Irish Coffee, only that the liquor used is Jamaican rum.

	Raw	Processed
Label	Nr. of items	Nr. of items
VariedFood	120	117
Beer	105	60
Spirits	101	79
Wine	71	66
SoftDrinks	50	32
CoffeeTeaMilk	32	29
Burger	21	21
Kids	14	14
Cider	14	14
Wings	12	5
Dessert	11	11
Other	7	7
Champagne	5	4
Salad	5	5

Table 3.2: Value counts by category for raw and processed data, manual labels, full list.

separate categories for 'Tea' and 'Milk'.

As mentioned before, the classification we present here is just one of the many possible and equally correct options. An interesting extension of this project would be to prepare several different classifications, and compare the results of the predictions.[7]

QUESTIONS: Should I talk more about the classification? In the remaining weeks, should I focus on preparing several different classifications and comparing the results?

3.2.2 Custom fit method and a custom metric

Now that we have classified our data set according to the newly-formed groups, we would like to somehow measure and compare the 'goodness-of-fit' between our classification and the sentence embeddings produced by the 3 models. To this purpose, we have developed a custom metric. The metric is based on the following steps:

- Step 1: Calculate the 'average embedding' for each category;
- Step 2: Calculate the average error for every category;
- Step 3: Calculate the average error for all categories (the entire data set).

We now explain these steps in detail. In the first step, we calculate the average embedding for each category, where the average embedding is simply the column-wise mean of all the embeddings for a given category. We illustrate with a simplified example. Let us imagine we have three items in the category 'Beer'. The items, along with their fictional sentence embeddings (we take only 3 dimensions for simplicity sake), are presented in Table 3.3. The average embedding of the category 'Beer' would then be the column-wise mean of the three vectors, as indicated in the same table. In other words, the average embedding is a kind of a 'centroid' for the cluster 'Beer'.

[7] For other possible extensions of the project, please refer to chapter 6.

Item	Dimension 1	Dimension 2	Dimension 3
Carlsberg Hof	2	5	7
Falcon Export	1	3	4
Brooklyn Lager	6	4	1
Avg embedding 'Beer'	3	4	4

Table 3.3: Average embedding for the category 'Beer', fictional example with 3 items and made-up embeddings.

Item	Cosine_sim (item, avg_emb)	Error
Carlsberg Hof	0.7	0.3
Falcon Export	0.8	0.2
Brooklyn Lager	0.9	0.1
Avg error = (0.3 + 0.2 + 0.1) / 3 = 0.2		

Table 3.4: Average error for the category 'Beer', fictional example with 3 items and made-up cosine similarities.

In the second step we then calculate the average error per category. This we do by calculating the error that every item in the category has in relation to the average embedding for that category, and then taking the average mean of those errors. The error is defined as:

$$err = 1 - cosine\_similarity(item\_embedding, average\_embedding\_category)$$

In other words, the error is a measure of how dissimilar the item is to the average embedding of the category. The error is bounded within $[0, 1]$, where a score close to 0 would mean that the item is extremely similar to the average embedding, and a score close to 1 would indicate that the item is extremely dissimilar to the average embedding. Looking at our example again, and assigning each item a fictional cosine similarity to the average embedding, we can observe the error each item has to the average embedding, as well as the resulting average error for the category 'Beer' in Table 3.4.

In the third and final step, we then calculate the average error for the entire data set, which we simply do by taking the average mean of all the average errors per category. To illustrate this with yet another example, if we added two more groups to our fictional data set, group 'Food' and group 'Wine', with their respective average errors (as calculated in steps 1 and 2), then the average error for the entire data set is presented in Table 3.5.

Now that we understand how our custom metric works, let us take a look at the results

Category	Average error
Beer	0.2
Wine	0.3
Food	0.4
Avg error for the data set = (0.2 + 0.3 + 0.4) / 3 = 0.3	

Table 3.5: Average error for the entire fictional data set, containing 3 categories with made-up average errors.

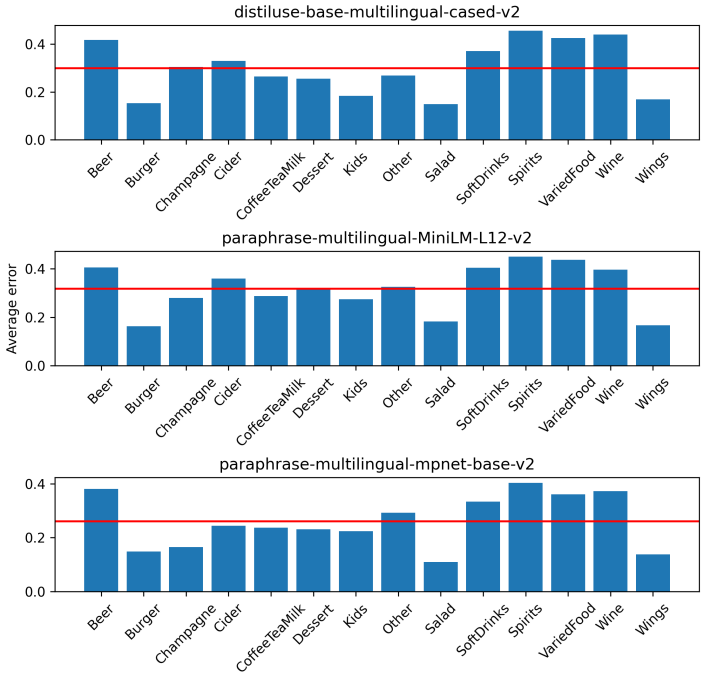

Figure 3.1: The average error per category for all 3 models, custom metric, raw data.

we got for our 3 multilingual sets of embeddings, for both raw and processed data. We present the average error per category for all 3 models, raw data, in Figure 3.1 (the red line denotes the average error for the entire data set). The average error per category for all 3 models, processed data, is shown in Figure 3.2 (the red line denotes the average error for the entire data set). The average error per model, raw data, is shown in Figure 3.3, and for the processed data in Figure 3.4.

We see that the para-base model slightly outperforms the other two, but the differences are very small. This confirms the results we got in the first part of the project where we were comparing the quality of the clusters after K-means and hierarchical clustering.

3.3 Custom fit and predict method for classification

3.3.1 Background

With this section, we are firmly moving into the domain of supervised learning. We will now be using the newly-classified data set to try and train a model to make predictions and automatically classify the items into one of the 14 categories. For this purpose, we need a fit method (which takes a set of training data and trains or fits the model to it) and a predict method (which takes a test data set and makes predictions on it)[leaj]. Our custom fit method consists of calculating the average embedding for every category, as described in the previous section. Our custom predict method we describe here.

To predict or automatically assign a label to an item, we take the item's sentence embedding, and loop through all the average category embeddings (one per category). We calculate the

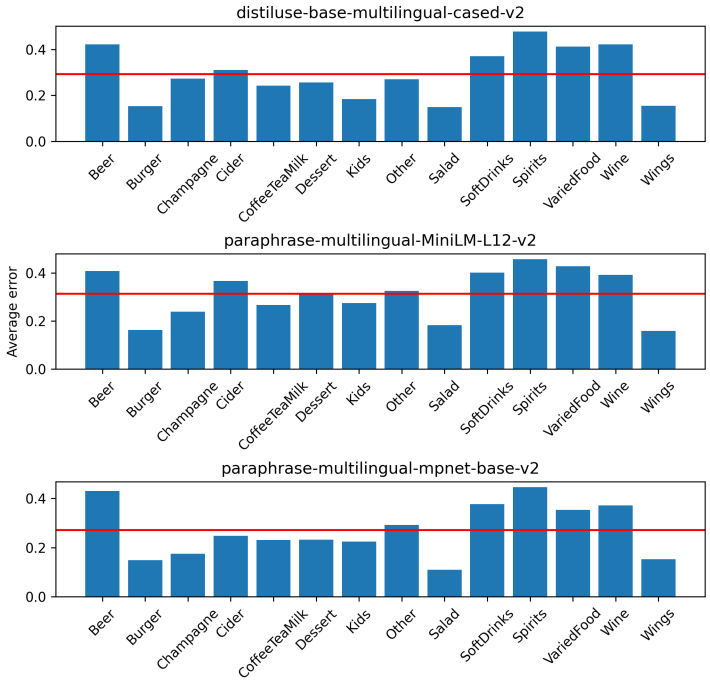

Figure 3.2: The average error per category for all 3 models, custom metric, processed data.

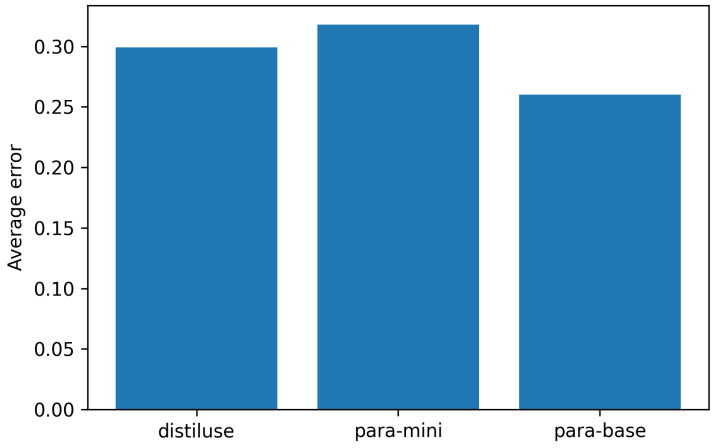

Figure 3.3: The average error per model, custom metric, raw data.

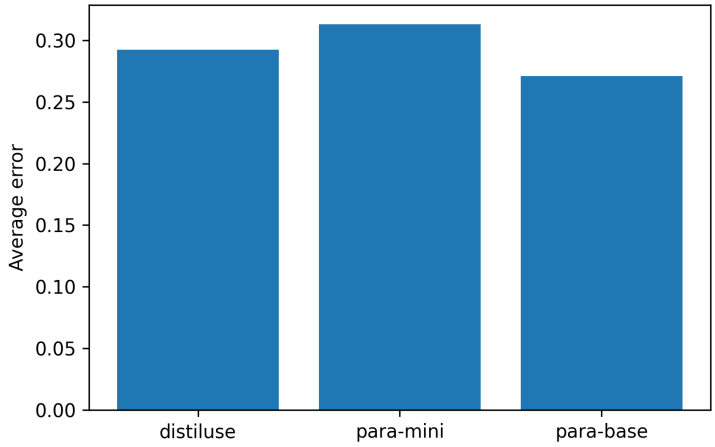

Figure 3.4: The average error per model, custom metric, processed data.

Item: 'Pepsi Max'	
Category	Error (item_emb, avg_cat_emb)
Beer	0.3
Soft Drinks	0.2
Food	0.4
Classified into: Soft Drinks	

Table 3.6: A fictional example of classification using our custom fit and predict methods.

error (defined exactly the same as in subsection 3.2.2), and we assign the item to the category where the error is the smallest. In other words, the item is assigned to the group whose average embedding has the greatest cosine similarity with the embedding of the item in question.

To illustrate this, let us imagine that we have three categories with three average embeddings, and an item called 'Pepsi Max' that we want to classify into one of the categories. We calculate the error between that item's embedding and all the average category embeddings. We assign the item to the category where the error is the smallest (Table 3.6).

3.3.2 Metrics

Before we inspect the results of our custom fit and predict methods, we discuss the metrics we use to measure how well the model performs. There are several options to choose from, e.g., accuracy, precision, recall, among others [Zuca]. In this project we use both accuracy and a lesser-known metric, called the F1 score. We present our reasoning below.

Both metrics use the following counts to calculate the score, namely 'True Positive' (TP: when a sample is predicted to be positive and its label is positive), 'True Negative' (TN: when a sample is predicted to be negative and its label is negative), 'False Positive' (FP: when a sample is predicted to be positive and its label is negative), 'False Negative' (FN: when a sample is predicted to be negative and its label is positive). These counts are often presented in a confusion matrix, essentially a table that presents the four values.

Accuracy is calculated using the following equation:

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Accuracy is bounded within $[0, 1]$. Its advantage is that it is a very intuitive and easy-to-understand metric. However, it does not work well with imbalanced data sets, as it tends to hide strong classification errors for classes with only a few units[GBV20].

In order to alleviate this problem, we use the F1-score to assess the performance per class. The F1-score is the harmonic mean of the precision and recall, given by the following formula:

$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

where precision is defined as: $\frac{\text{TP}}{\text{TP+FP}}$ and recall as $\frac{\text{TP}}{\text{TP+FN}}$. The F1-score is bounded by $[0, 1]$. We use it because it gives more weight to smaller classes and rewards models that have similar

precision and recall scores[GBV20]. It is also the preferred metric to use in NLP literature, despite some drawbacks[Der16].

To assess the performance on the entire data set we use the macro F1-score. The macro F1-score is given by the following equation:

$$\text{macro F1-score} = 2 \cdot \frac{\text{MacroAvgPrecision} \cdot \text{MacroAvgRecall}}{\text{MacroAvgPrecision}^{-1} + \text{MacroAvgRecall}^{-1}}$$

The advantage and the main reason why we use this score is that with the F1-score classes of different sizes have equal weight[GBV20]. This means that a high macro F1-score indicates a good performance on all the classes, regardless of their size, because the small classes have the same weight as the large classes.

3.3.3 Results

We use Scikit-learn's 'metrics' module[leaf] to calculate all the metrics that we present below. To train and test our custom fit and predict methods, we take the labelled data set and divide it into a training and a test set (20% of the data set is randomly assigned to the test set). After training the test set on our model (calculating the average embeddings of all categories) and running our custom predict function (assigning an item to the category whose average embedding is closest to the item in terms of cosine similarity), we present the results of the predictions.

We begin by showing the confusion matrices for all sets of embeddings, both raw and processed data. The confusion matrices are shown in Figure 3.5 (distiluse, raw data), Figure 3.6 (distiluse, processed data), Figure 3.7 (para-mini, raw data), Figure 3.8 (para-mini, processed data), Figure 3.9 (para-base, raw data) and Figure 3.10 (para-base, processed data).

We then present the accuracy score for all embeddings, raw and processed data, in Figure 3.11, as well as the macro F1-score in Figure 3.12. As expected and explained above, the macro F1-score is generally lower than the accuracy. Finally, we also present the F1-score per category for all embeddings, both raw (in Figure 3.13) and processed data (in Figure 3.14).

The results show that our classifier performs reasonably well and comparatively equally with all sets of embeddings. The macro F1-score is somewhere in the $[0.7, 0.8]$ range, which is rather good, considering how simple our train and predict functions are, and the fact that there are 14 categories to choose from. There is, however, a possibility that this result suffers from overfitting on the data set.[8]. An interesting extension of this project would be to test our model on other, unseen data, and compare results.

QUESTIONS: Should I present more mathematical details regarding the metrics? Should I extend the project by testing on new, unseen data?

[8]Overfitting in ML refers to the phenomenon when a model "fits exactly against its training data" but "cannot perform accurately against unseen data, defeating its purpose"[IBMb].

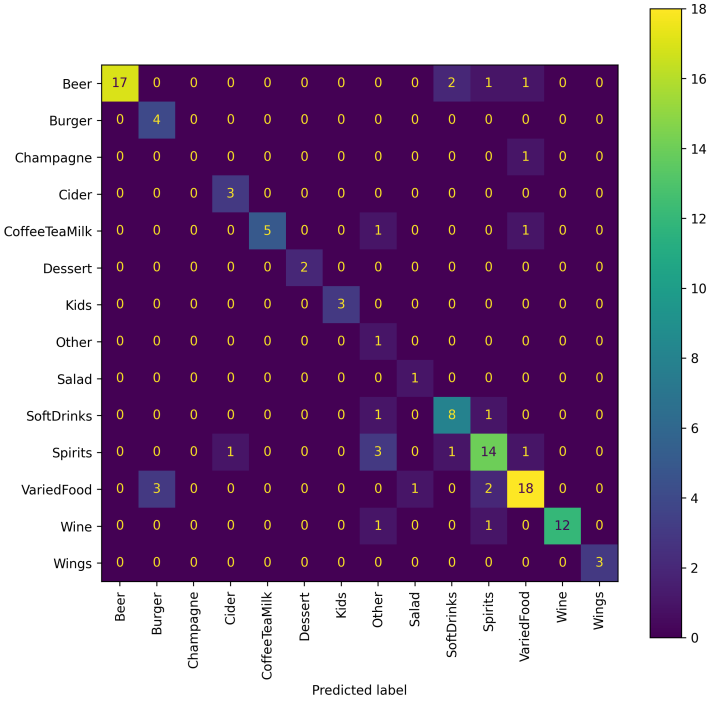

Figure 3.5: The confusion matrix after custom fit and predict, distiluse embeddings, raw data.

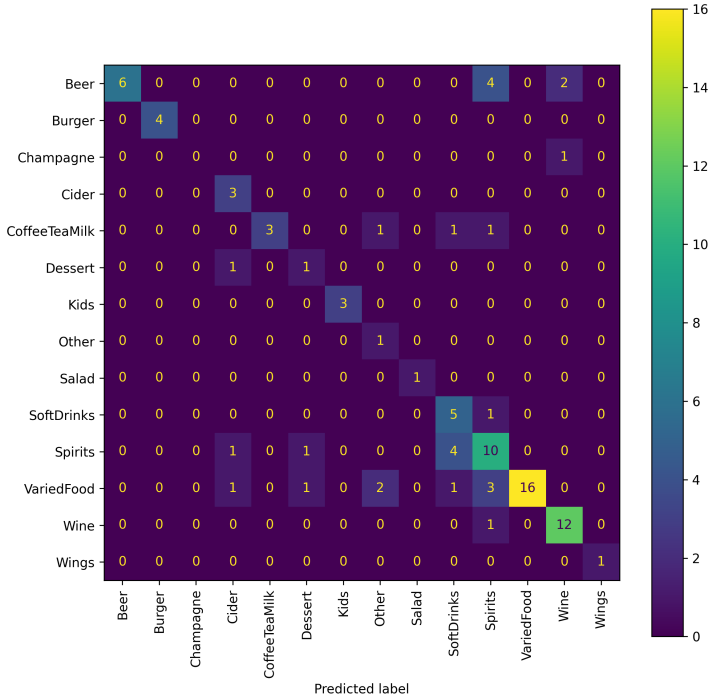

Figure 3.6: The confusion matrix after custom fit and predict, distiluse embeddings, processed data.

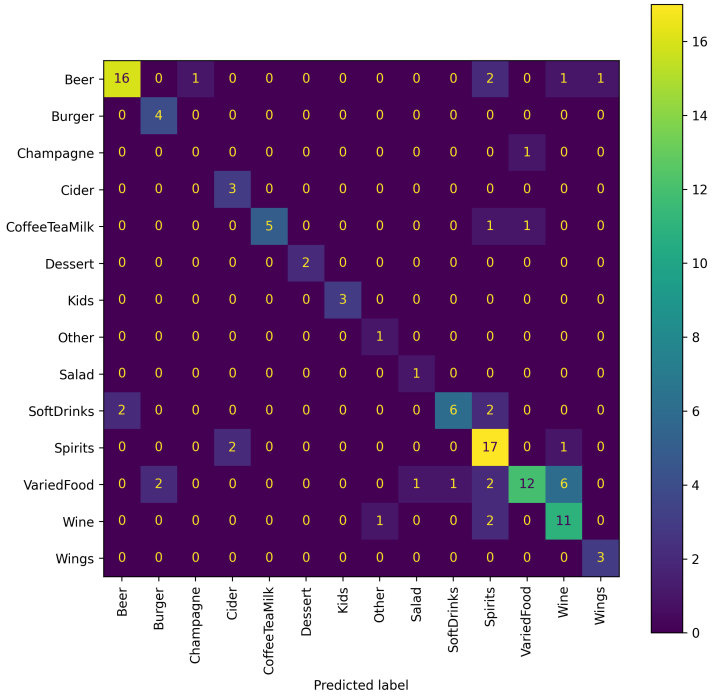

Figure 3.7: The confusion matrix after custom fit and predict, para-mini embeddings, raw data.

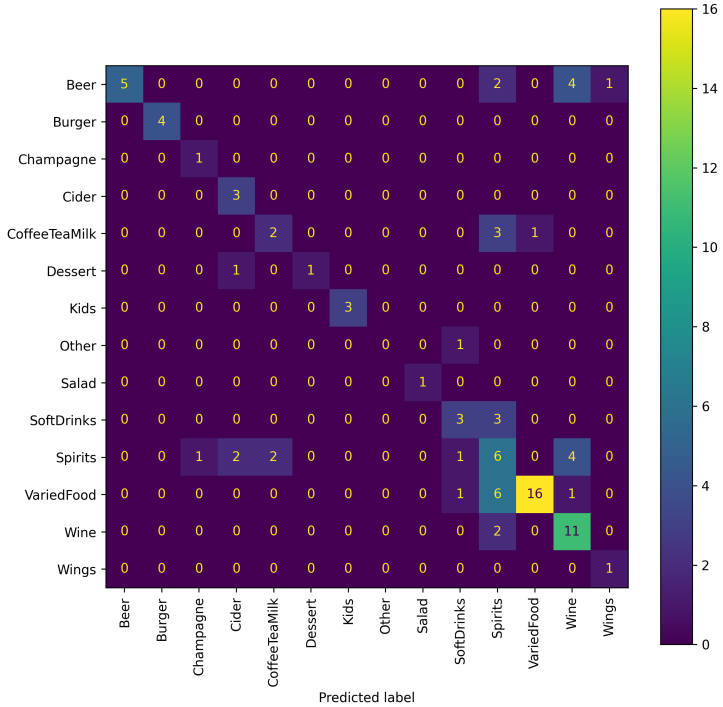

Figure 3.8: The confusion matrix after custom fit and predict, para-mini embeddings, processed data.

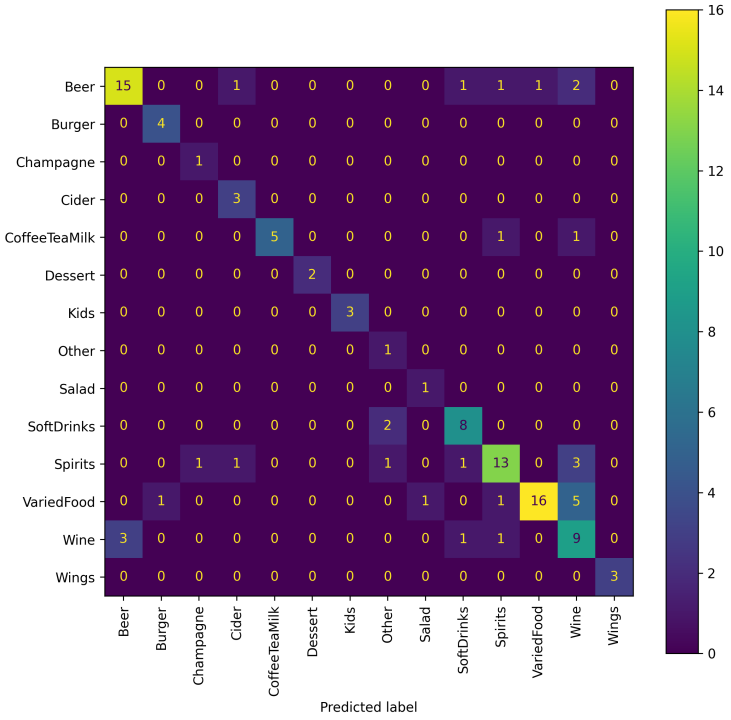

Figure 3.9: The confusion matrix after custom fit and predict, para-base embeddings, raw data.

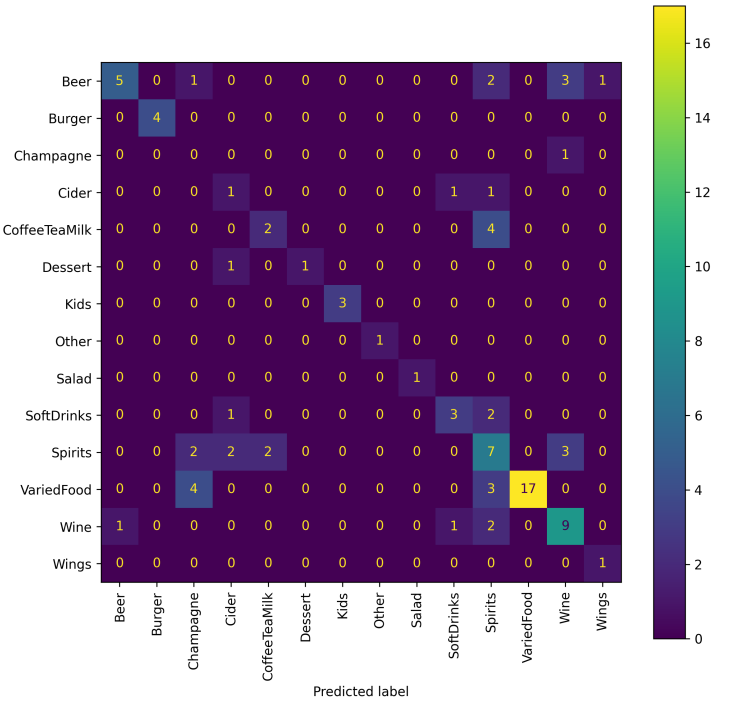

Figure 3.10: The confusion matrix after custom fit and predict, para-base embeddings, processed data.

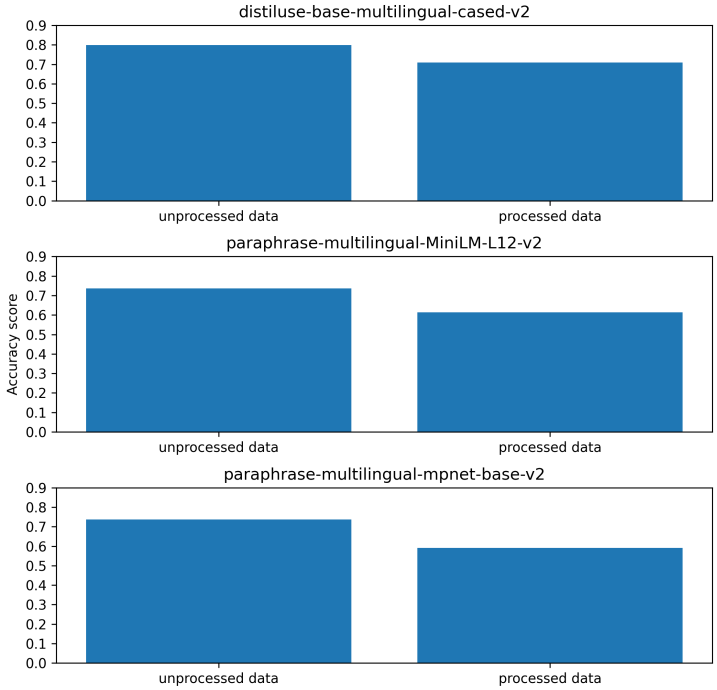

Figure 3.11: The accuracy score for all embeddings after custom fit and predict.

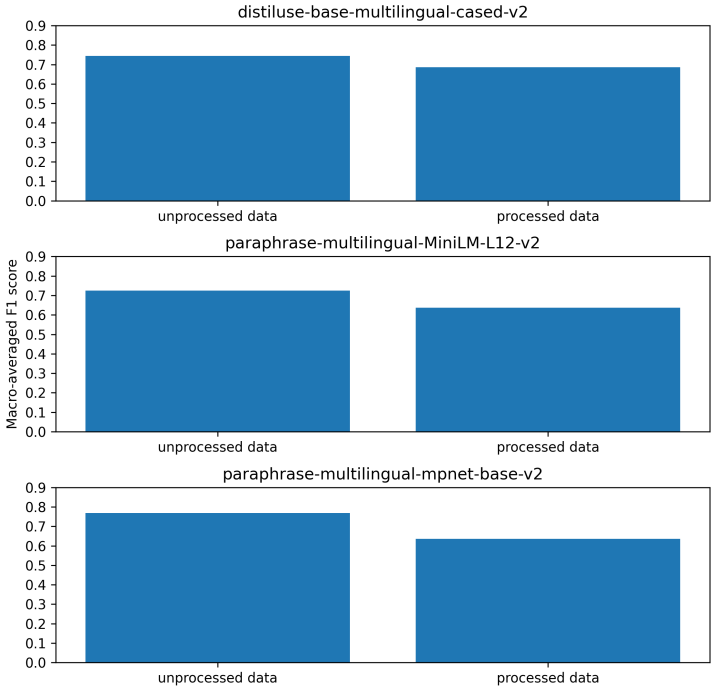

Figure 3.12: The macro F1-score for all embeddings after custom fit and predict.

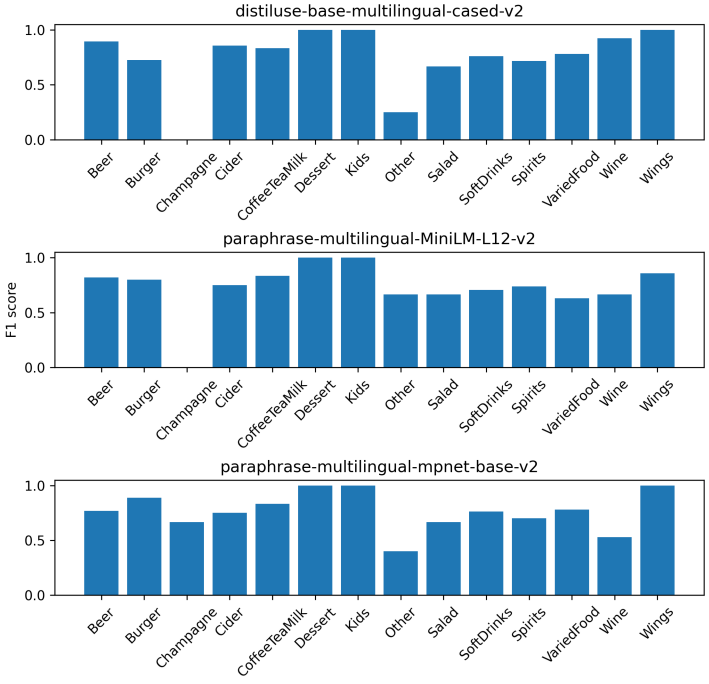

Figure 3.13: The F1-score per category for all embeddings after custom fit and predict, raw data.

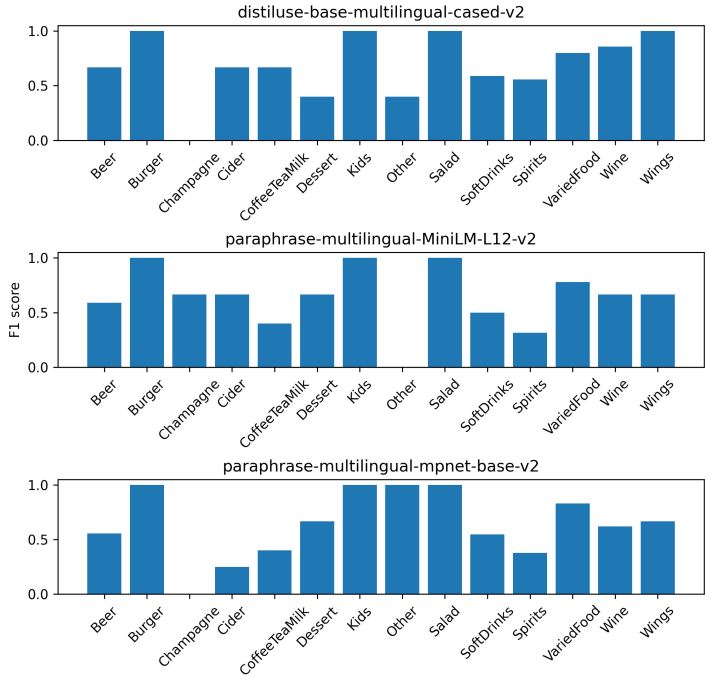

Figure 3.14: The F1-score per category for all embeddings after custom fit and predict, processed data.

3.4 K-nearest neighbors classifier

3.4.1 An overview

In this section we present the results of running the K-nearest neighbours algorithm (k-NN) on our labelled data set. We run the algorithm three times, first as-is, then with the help of Neighborhood Component Analysis (NCA), and finally after performing dimensionality reduction with NCA. Before running k-NN we do cross-validation on the data set in order to determine an appropriate value for the hyper-parameter k. We present the technical details and the results in the following subsections.

3.4.2 Background

K-NN is a classical classification algorithm for supervised learning. It labels a sample according to the majority label of the k-nearest neighbours. To this end, the algorithm needs to measure the distance between the unlabelled sample and the other, already labelled samples. It can then choose the k-nearest samples, look at their labels, and assign the new sample to the label that is most prevalent in the k-nearest samples[Stab]. To illustrate with an example, let us imagine that we have an uncategorised sample called 'Staropramen'.[9] Then let us say that our hyper-parameter k=*3*, which means that we will be looking at the 3 nearest neighbours to 'Staropramen'. We then calculate the distance from 'Staropramen' to all the other samples, and pick the 3 nearest to it. Let us say that the 3 nearest samples are 'Cappuccino', classified as 'CoffeeTeaMilk', 'Brooklyn Defender', classified as 'Beer', and 'Brooklyn Summer Ale', also classified as 'Beer'. Since the 'Beer' category receives the most votes, we correctly classify 'Staropramen' as a 'Beer'. This example also shows why it is a good idea to pick an odd number for k, so that we avoid ties in votes.

It is important to note that we can use several metrics to calculate the distance between the new, unlabelled sample, and the other, labelled samples[leai]. We use the Manhattan distance metric as it is more suitable than the Euclidean distance metric in high dimensional space[AHK01].

3.4.3 Choosing k with cross-validation

An important hyper-parameter to choose when running k-NN is the parameter k. It determines the number of neighbours the algorithm should look at when deciding how to label a sample. To determine the optimal value for k we use a technique called cross-validation (CV)[Jai]. Cross-validation is a technique that enables us to learn the parameters of the prediction function while avoiding overfitting on the training data[leab]. CV works by partitioning the training set into k smaller sets (folds). A model is then trained on k-1 folds, and the resulting model is validated on the remaining fold (essentially used as a test set). This procedure is performed k-times, so that each fold plays the role of the test set once[Jai].

[9] A type of beer.

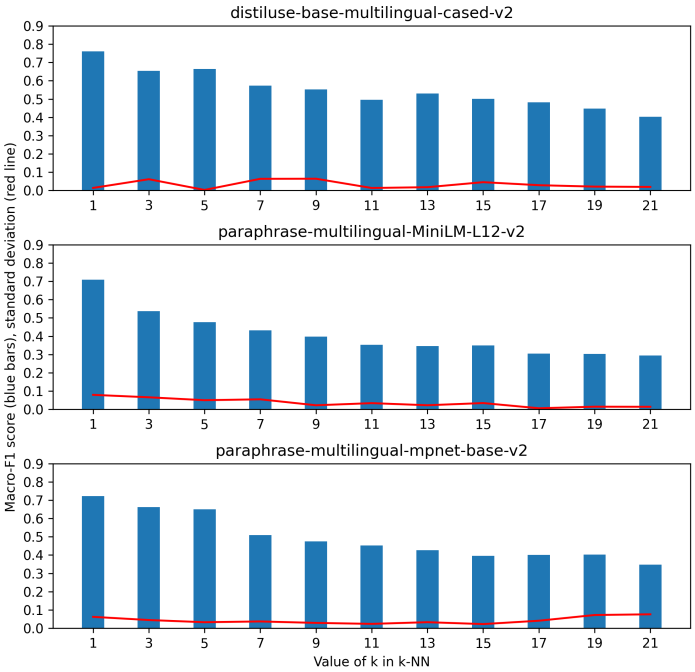

Figure 3.15: Macro F1-score and standard deviation for different values of k in k-NN, raw data.

We run k-NN in a loop, incrementing the hyper-parameter k by a step size of 2 (as mentioned before, we want to avoid even numbers, which could lead to ties in the votes for categories). We then cross-validate the model for each run, using a 3-fold splitting strategy, and we take the mean of the scores as our final evaluation of the performance. We also record the standard deviation of the 3 scores. We present the results for all embeddings, raw data in Figure 3.15, and processed data in Figure 3.16.

Looking at the results and considering the fact that our smallest categories in the raw data set, e.g., 'Salad' and 'Champagne', only contain 4 items, whereas our smallest category in the processed data set, 'Champagne', only contains 3 items, we might not want to consider k that is smaller than those values. Choosing a value for k that is much larger than either 3 or 4 would lead to those smaller categories consistently being outvoted[Stab]. Choosing $k=1$ might lead to overfitting[Agg], so we ignore that option. It would seem that for the unprocessed data, choosing $k=5$ will strike a balance between performance and a low standard deviation, whereas $k=3$ will be better for processed data. We therefore retrain all raw data sets with $k=5$ and all processed data sets with $k=3$ and then test for accuracy on the test set.

3.4.4 k-NN results

The results are presented as follows. The confusion matrices for the raw data set, distiluse model in Figure 3.17, para-mini model in Figure 3.19 and the para-base model in Figure 3.21. For the processed data, distiluse model in Figure 3.18, para-mini model in Figure 3.20 and the para-base model in Figure 3.22. The accuracy for all data sets and embeddings is presented in Figure 3.23. The macro F1-score for all data sets and embeddings can be observed in Figure 3.24. The F1-score per category for the raw data is in Figure 3.25, whereas for the processed data please refer

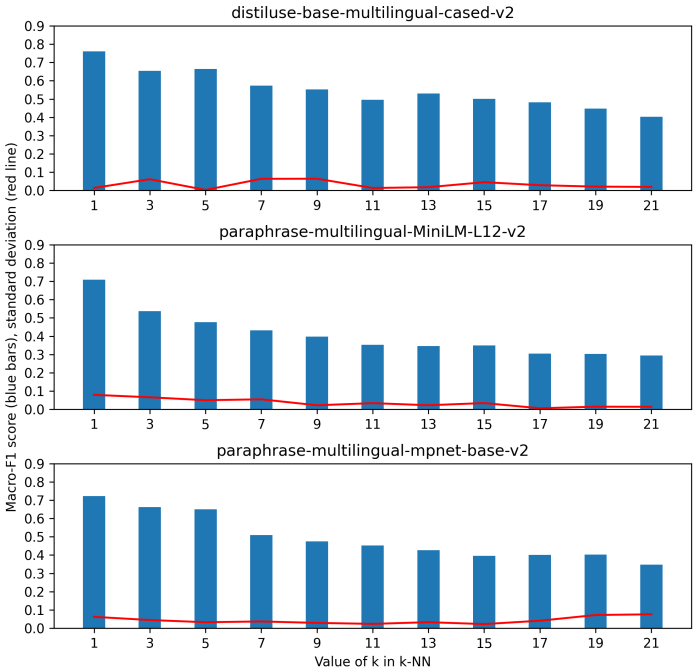

Figure 3.16: Macro F1-score and standard deviation for different values of k in k-NN, processed data.

to Figure 3.26.

Here we can see the importance of using the F1-score and the macro F1-score as our measuring stick, instead of accuracy. Looking at the accuracy score for the raw data (Figure 3.23), the distiluse model outperforms both the para-mini and the para-base models. However, looking at the confusion matrix for the distiluse model, raw data (Figure 3.17) and comparing it to the confusion matrix of the para-base model, raw data (Figure 3.21) we see that the distiluse model makes more errors on the smaller categories than the para-base model does. The F1-score and the macro F1-score penalise that. According to the macro F1-score for raw data (Figure 3.24), the para-base embeddings therefore greatly outperform both the para-mini and the distiluse embeddings.

3.5 K-nearest neighbors with Neighborhood component analysis

Neighborhood component analysis (NCA) is a distance metric learning algorithm, intended to improve the accuracy of k-NN classification[leag]. Its goal is to learn an optimal linear transformation matrix, which maximises the sum over all samples i of the probability p_i that i is correctly classified[leag].[10]

We use Scikit-learn's implementation of NCA and report the results here. The confusion matrices for the raw data set are in Figure 3.27 (distiluse), Figure 3.29 (para-mini) and Figure 3.31 (para-base). For the processed data, please refer to Figure 3.28 (distiluse), Figure 3.30 (para-mini),

[10]For a complete mathematical formulation, please refer to [Gol+04].

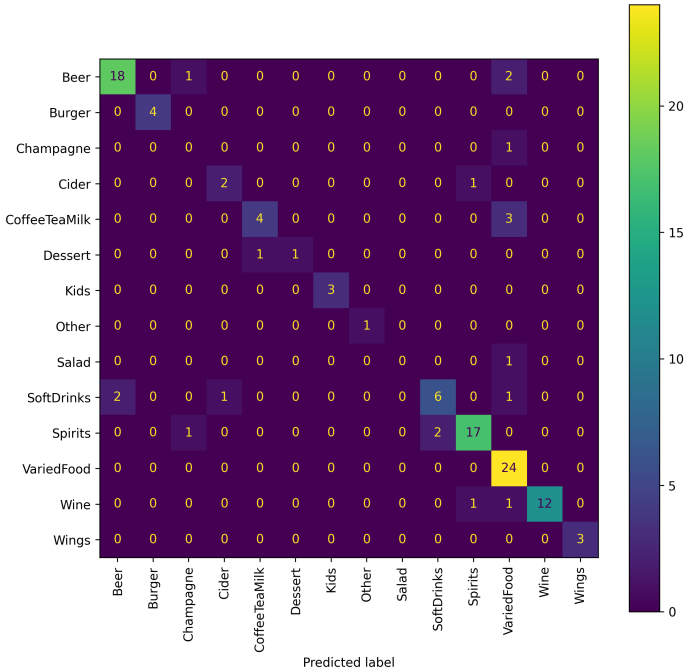

Figure 3.17: Confusion matrix after k-NN, distiluse embeddings, raw data.

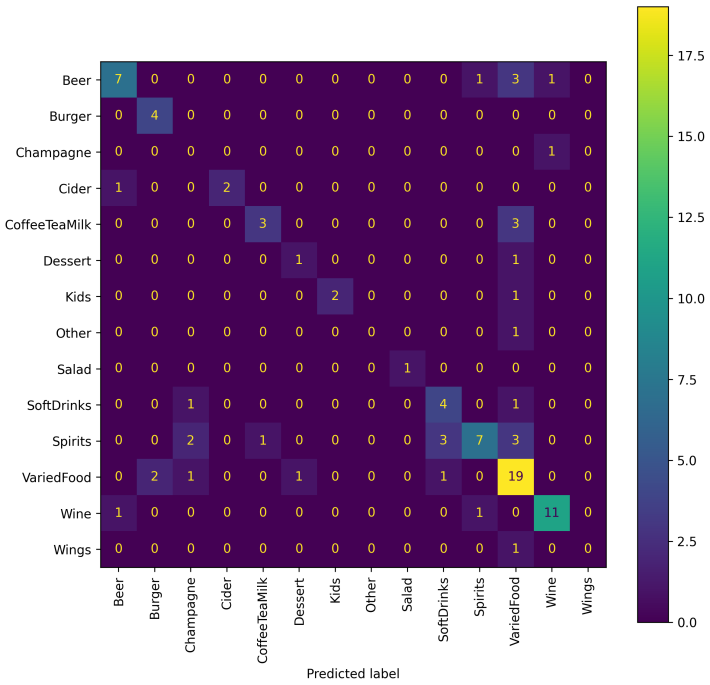

Figure 3.18: Confusion matrix after k-NN, distiluse embeddings, processed data.

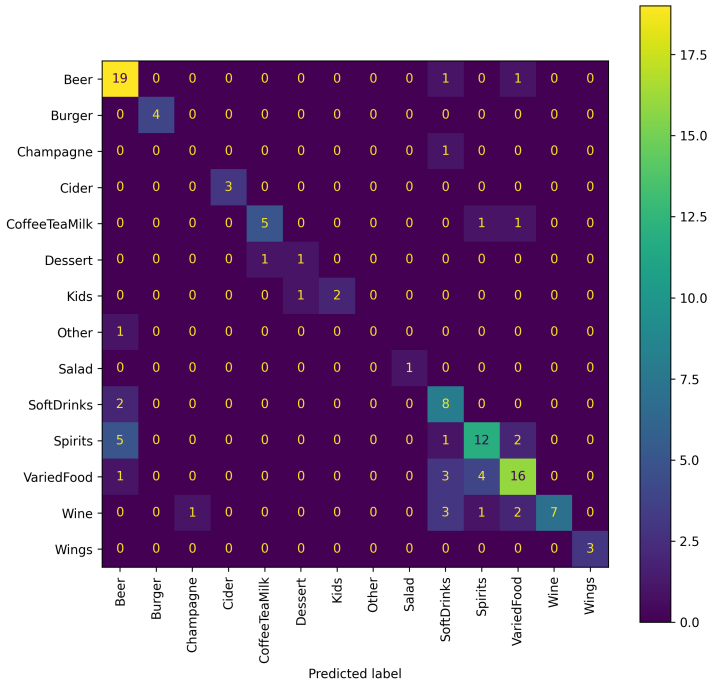

Figure 3.19: Confusion matrix after k-NN, para-mini embeddings, raw data.

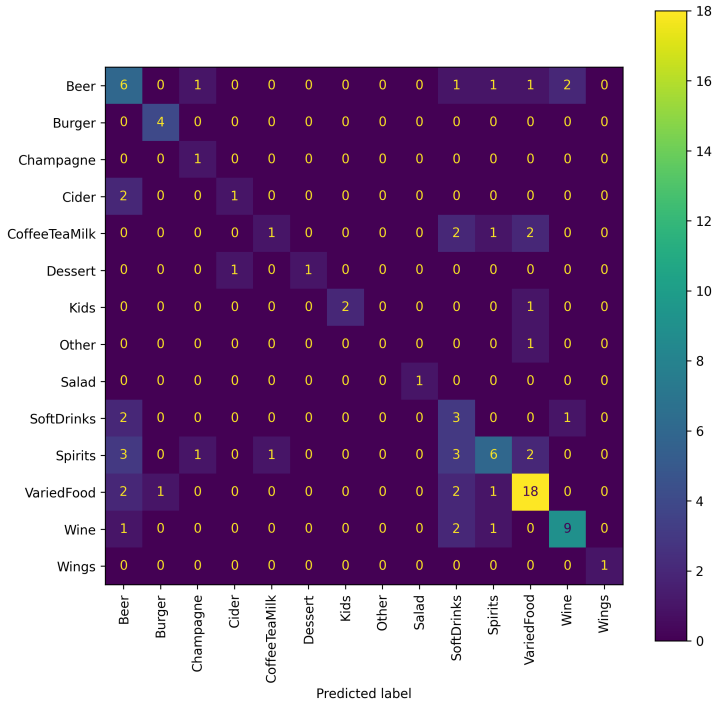

Figure 3.20: Confusion matrix after k-NN, para-mini embeddings, processed data.

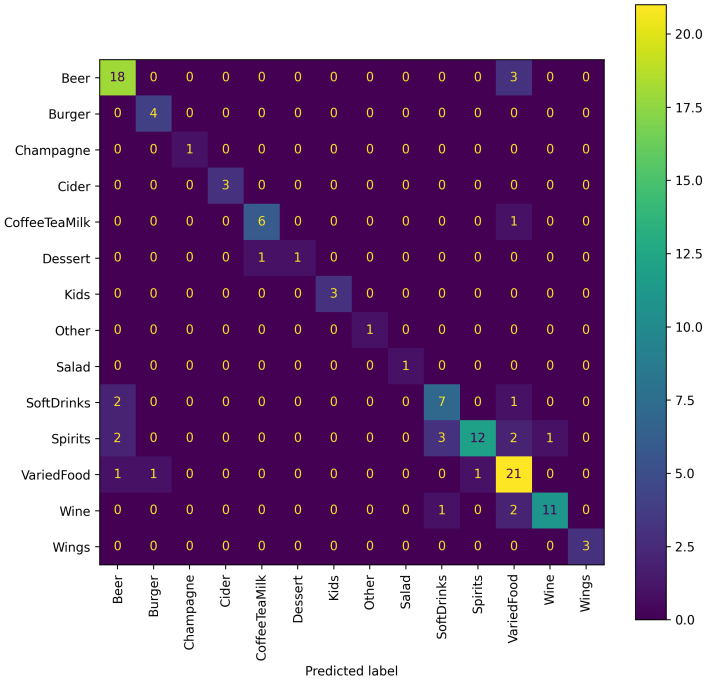

Figure 3.21: Confusion matrix after k-NN, para-base embeddings, raw data.

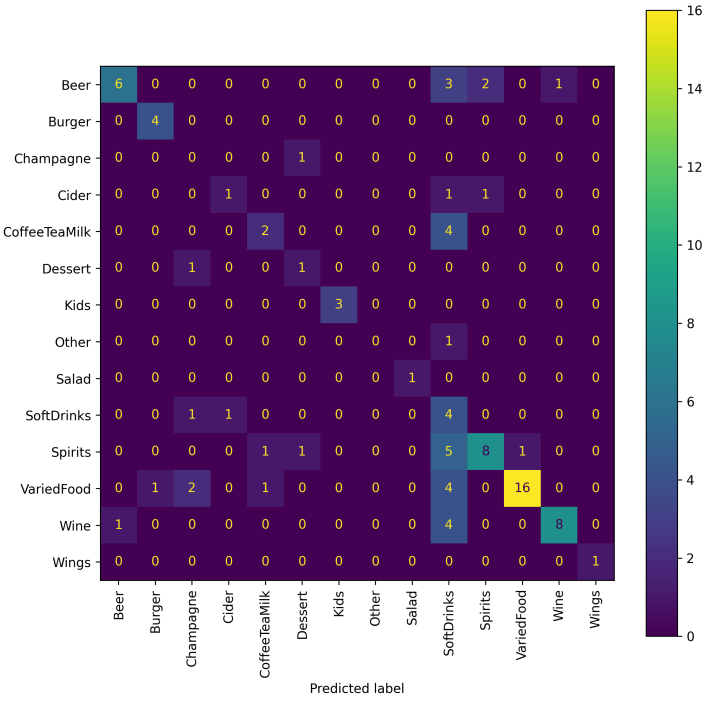

Figure 3.22: Confusion matrix after k-NN, para-base embeddings, processed data.

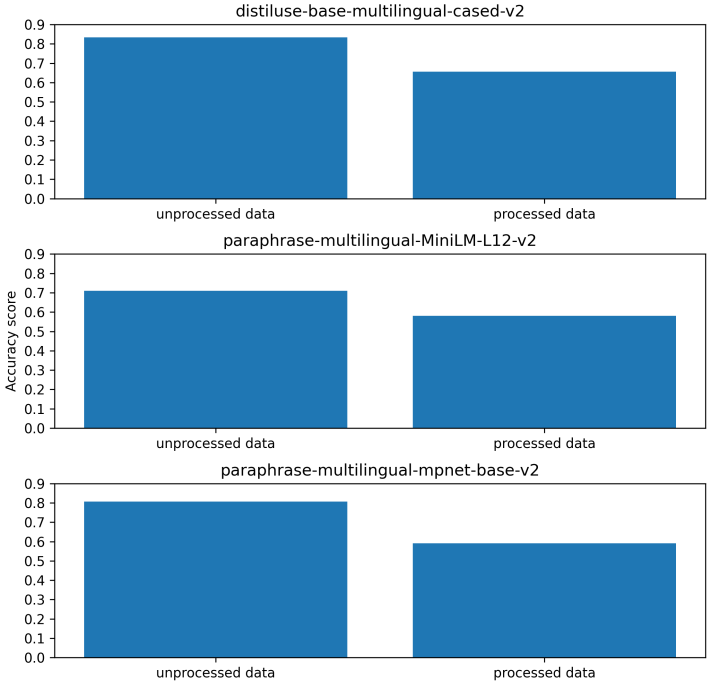

Figure 3.23: Accuracy score after k-NN.

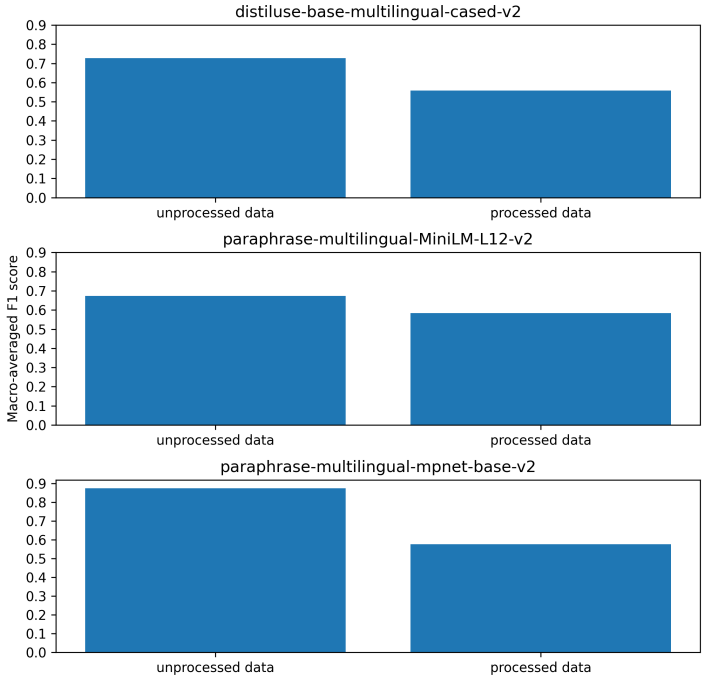

Figure 3.24: Macro F1-score after k-NN.

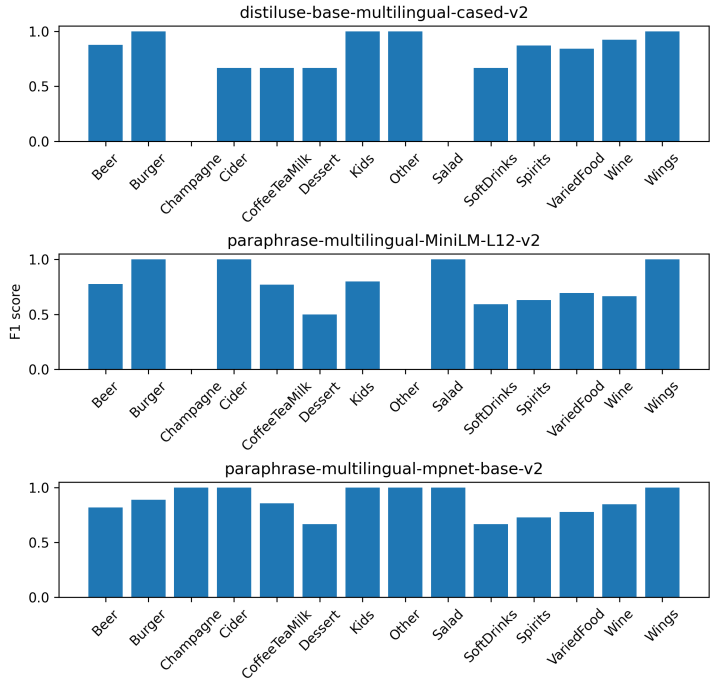

Figure 3.25: F1-score per category after k-NN, raw data.

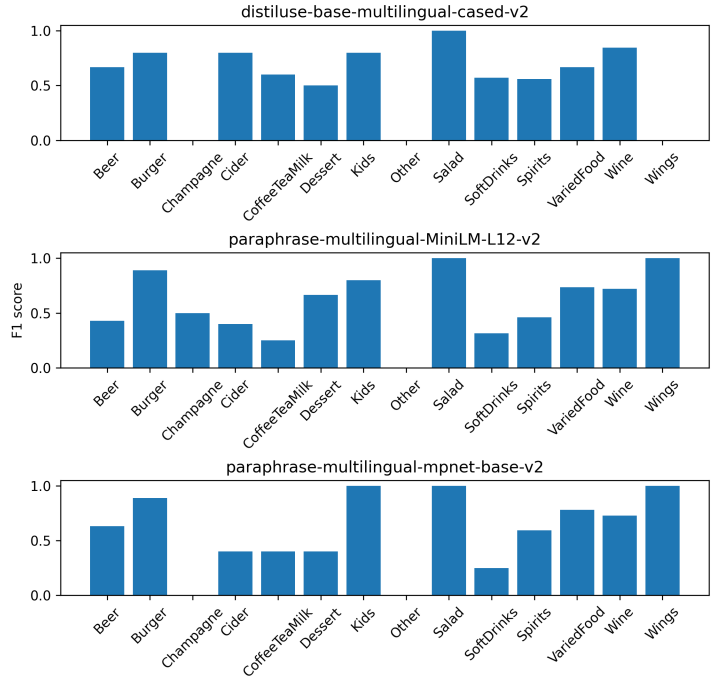

Figure 3.26: F1-score per category after k-NN, processed data.

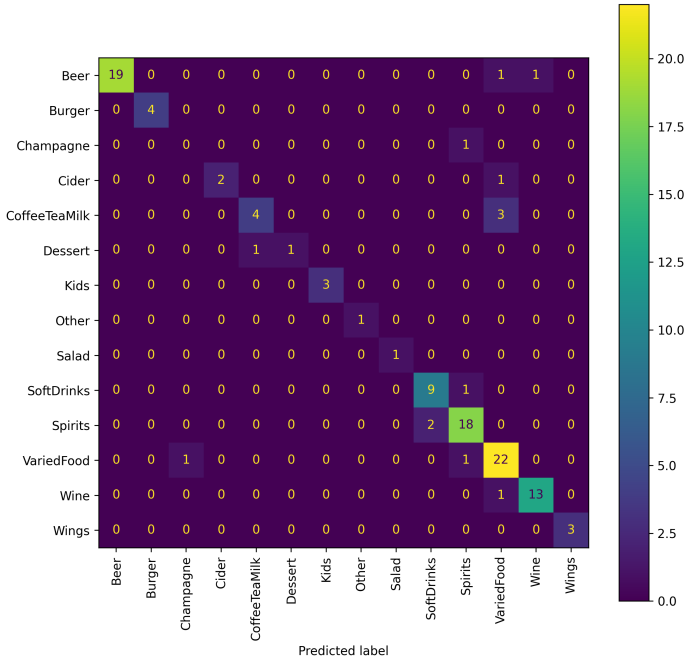

Figure 3.27: Confusion matrix after NCA and k-NN, distiluse embeddings, raw data.

and Figure 3.32 (para-base). The accuracy score for all embeddings, raw and processed data is in Figure 3.33. For the macro F1-score please refer to Figure 3.34. The F1-score per category for the raw data is presented in Figure 3.35, and for the processed data in Figure 3.36.

Comparing the accuracy score of k-NN with NCA (Figure 3.33) to the accuracy score of k-NN without NCA (Figure 3.23), we see a significant improvement for the distiluse and the para-mini embeddings, for both raw and processed data. For the para-base embeddings we see a significant improvement for the processed data only. The macro F1-score also shows a significant improvement for both the distiluse and the para-mini embeddings, raw and processed data alike, but a significant decrease in performance for the para-base embeddings (compare Figure 3.24 with Figure 3.34).

QUESTIONS: I don't know why this is the case. Should I try and make an educated guess? Should I explain NCA in more technical detail?

3.6 K-nearest neighbors with Neighborhood component analysis dimensionality reduction

The k-NN algorithm can suffer from the 'curse of dimensionality'[Kar] because it relies on the assumption that similar points lie close together. Namely, in high-dimensional spaces, this assumption usually does not hold[Uni]. We therefore perform dimensionality reduction before running k-NN again, and we perform this reduction using NCA. We use NCA instead of some other dimensionality reduction technique, e.g., PCA, because NCA lends itself well to classification problems thanks to its ability to handle multi-class problems. It also requires no additional parameters that

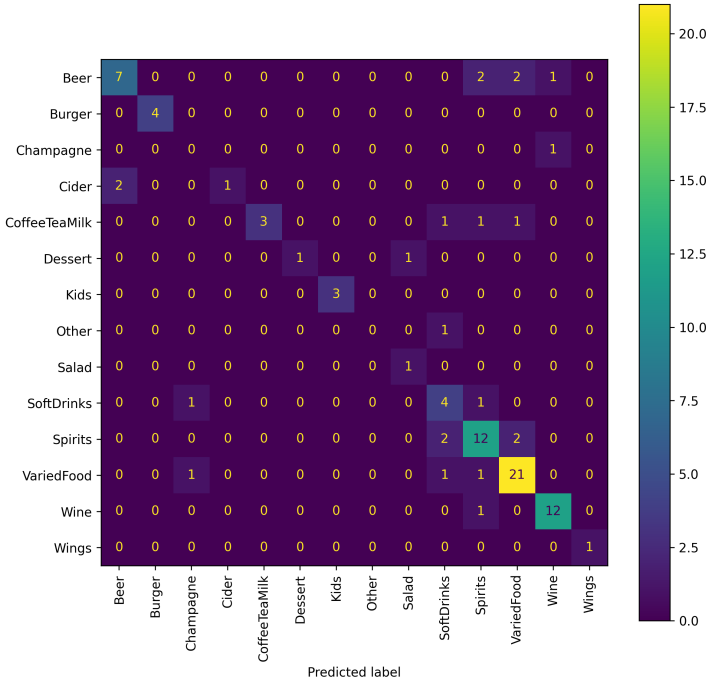

Figure 3.28: Confusion matrix after NCA and k-NN, distiluse embeddings, processed data.

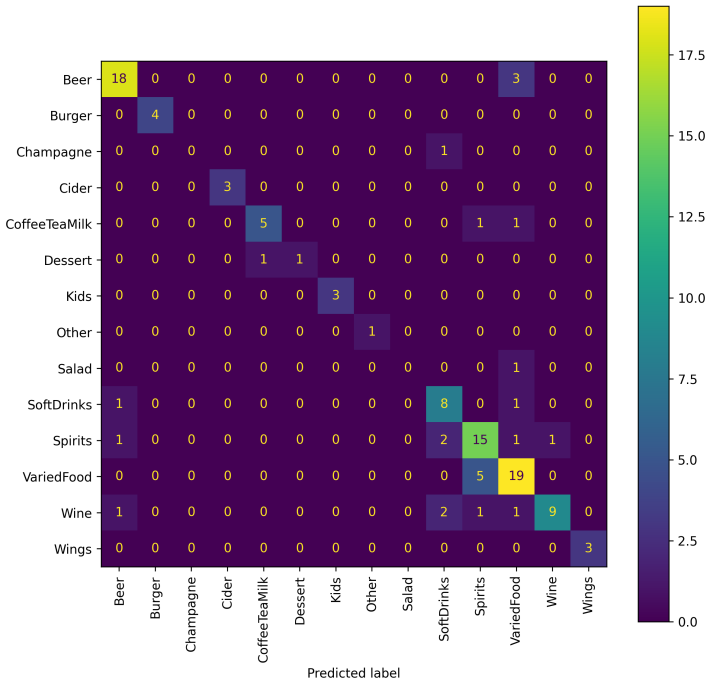

Figure 3.29: Confusion matrix after NCA and k-NN, para-mini embeddings, raw data.

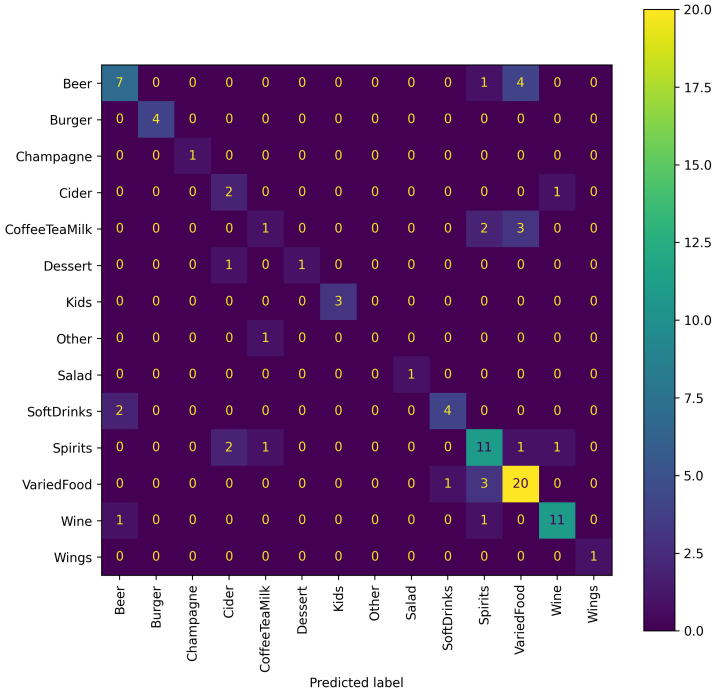

Figure 3.30: Confusion matrix after NCA and k-NN, para-mini embeddings, processed data.

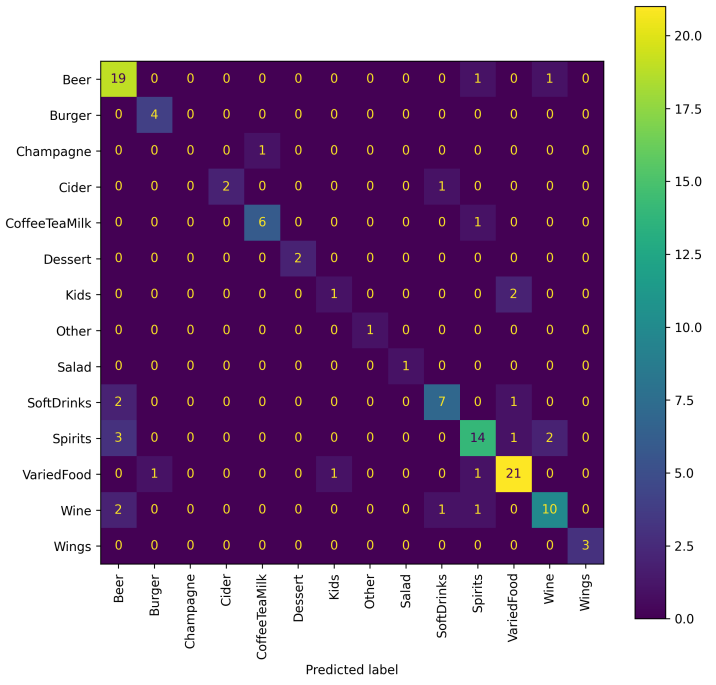

Figure 3.31: Confusion matrix after NCA and k-NN, para-base embeddings, raw data.

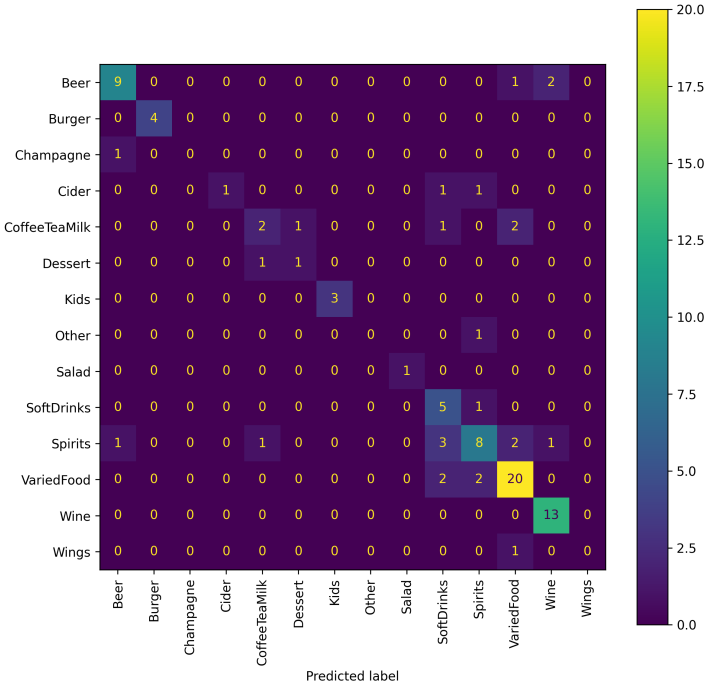

Figure 3.32: Confusion matrix after NCA and k-NN, para-base embeddings, processed data.

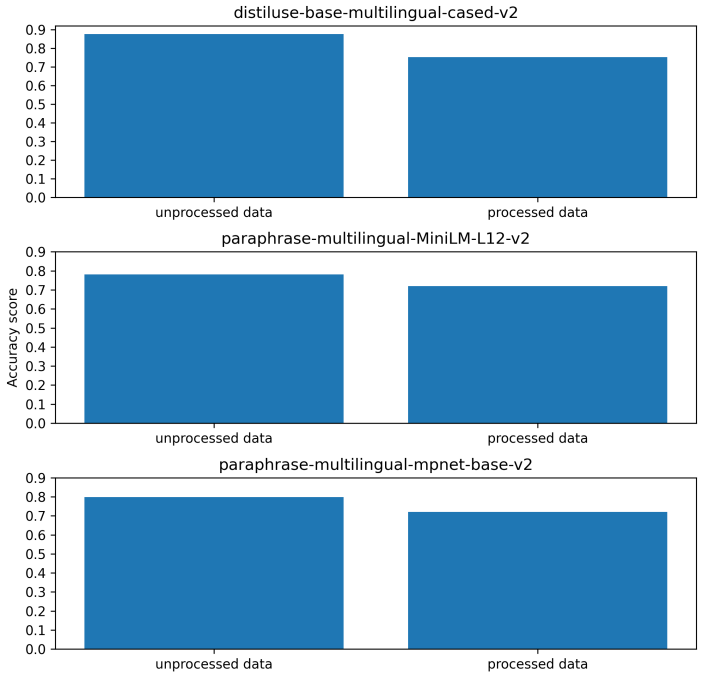

Figure 3.33: Accuracy score after NCA and k-NN.

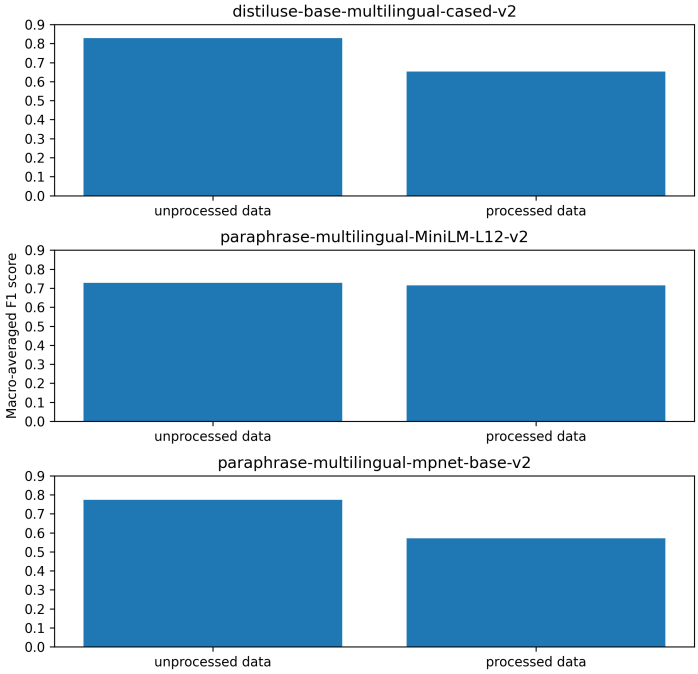

Figure 3.34: Macro F1-score after NCA and k-NN.

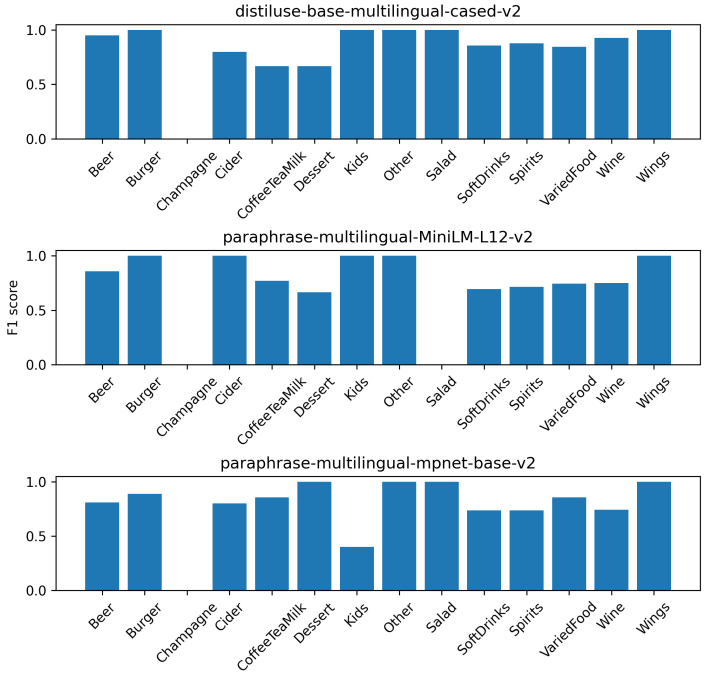

Figure 3.35: F1-score per category after NCA and k-NN, raw data.

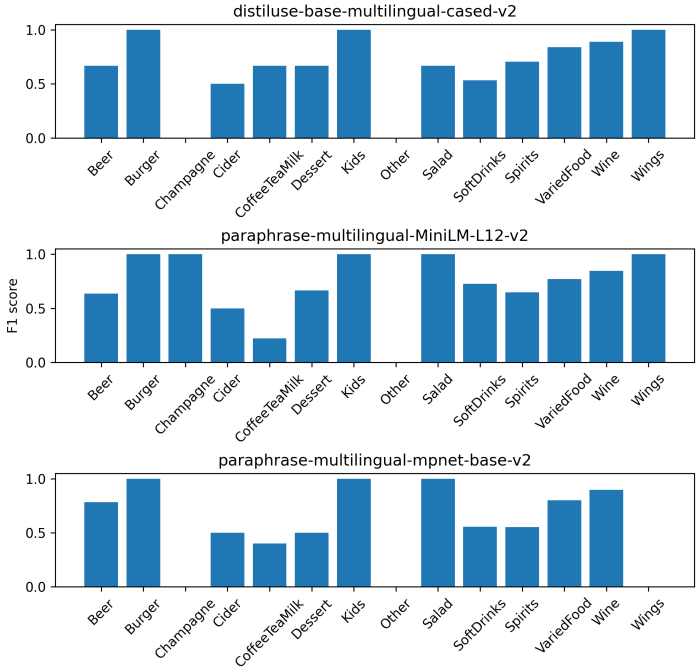

Figure 3.36: F1-score per category after NCA and k-NN, processed data.

would require fine-tuning by the user[leag]. Furthermore, research shows that dimensionality reduction with NCA, followed by k-NN, gives a significantly better result than dimensionality reduction with PCA, followed by k-NN[leac].

Before using NCA we would, however, like to know how many dimensions we would like to keep. To this end, we perform PCA. As mentioned briefly in subsection 2.2.4, PCA is a technique used to identify the orthogonal components of a data set that capture the maximum amount of the variance in the data[leah]. The principal components, produced by PCA, are eigenvectors of the data's covariance matrix[Wik22b]. After having produced the principal components, we can then decide how many of them we would like to keep, depending on how much of the total variance of the data set we would like to retain. In our case, we decide we would like to retain 99% of the information in each data set.[11] A quick analysis, the results of which are shown in Table 3.7, reveals how many dimensions we have to keep for every data set and how much variance is retained by those dimensions. We present the relationship between variance and the number of dimensions for the distiluse embeddings in Figure 3.37, for the para-mini embeddings in Figure 3.38 and for the para-base embeddings in Figure 3.39.

Now that we know how many dimensions we want to keep for each data set, we use NCA to perform the actual dimensionality reduction. Once we have reduced the dimensions with NCA, we run k-NN and present the results below.

The confusion matrices are shown in the following figures: distiluse embeddings, raw data in Figure 3.40, distiluse embeddings, processed data in Figure 3.41, para-mini embeddings, raw data in Figure 3.42, para-mini embeddings, processed data in Figure 3.43, para-base embeddings,

[11] We could have chosen to retain less of the variance, but we felt uncomfortable with losing that much information. Typical threshold values are around 90% - 95%.

	Nr. of components to keep		
Embeddings	Raw data	Processed data	Variance retained
distiluse	198	190	99.02%
para-mini	152	148	99.02%
para-base	179	174	99.02%

Table 3.7: The number of components we need to keep in order to retain the same amount of variance.

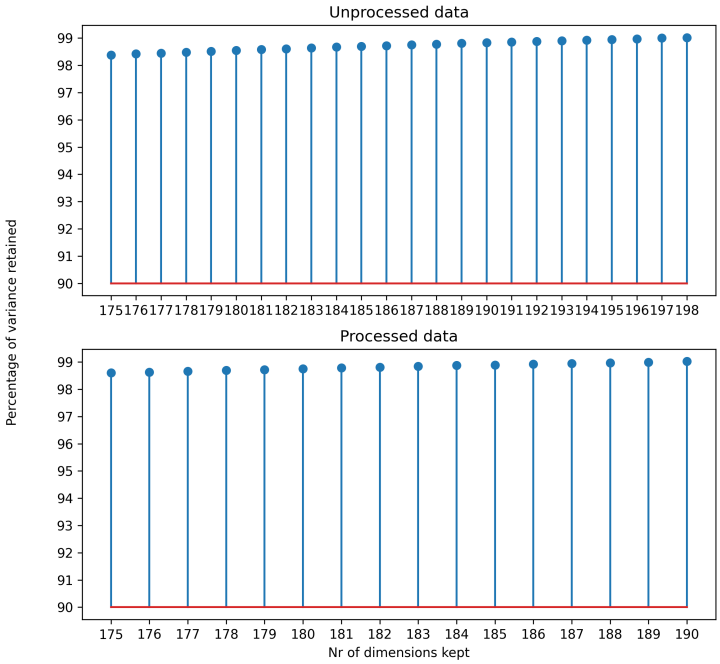

Figure 3.37: Variance in relation to the nr. of dimensions kept, distiluse embeddings.

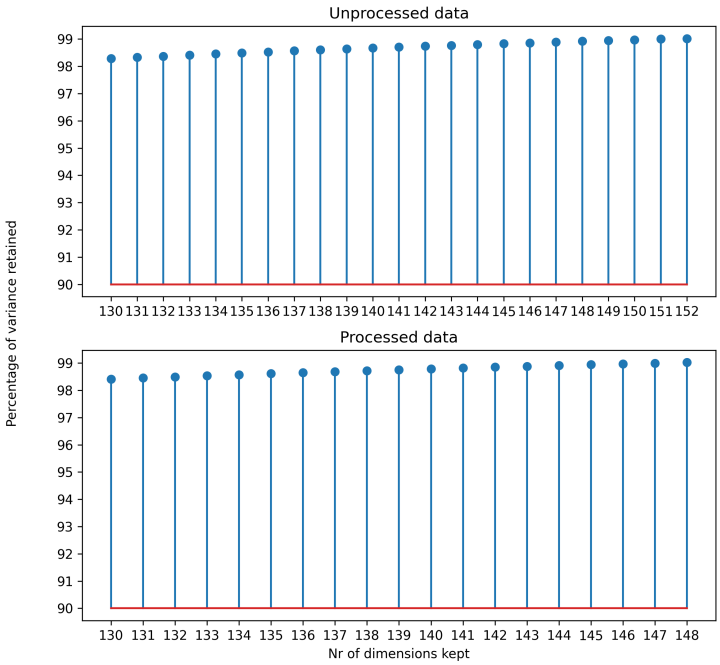

Figure 3.38: Variance in relation to the nr. of dimensions kept, para-mini embeddings.

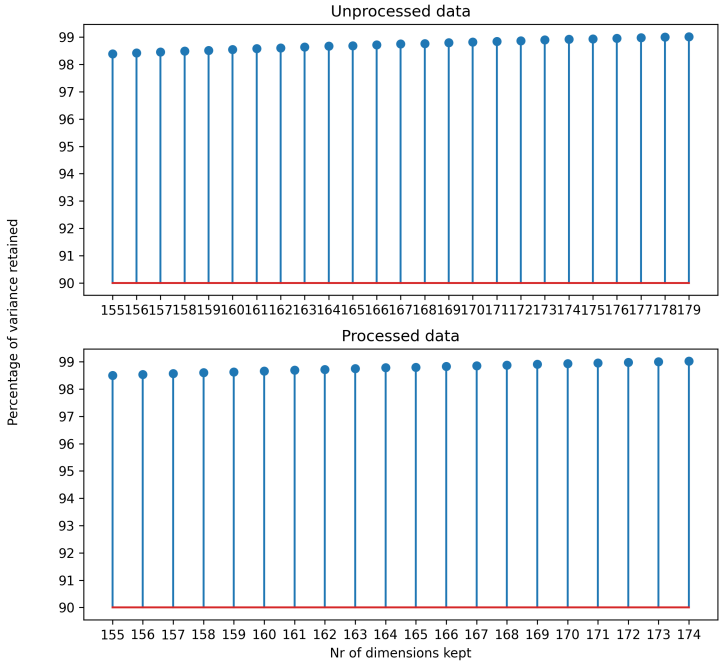

Figure 3.39: Variance in relation to the nr. of dimensions kept, para-base embeddings.

raw data in Figure 3.44 and para-base embeddings, processed data in Figure 3.45. The accuracy scores for all embeddings, raw and processed data, are presented in Figure 3.46. The macro F1-score for all embeddings, raw and processed data, is in Figure 3.47. The F1-score per category for the raw data is in Figure 3.48, and for the processed data in Figure 3.49.

Comparing these results to the results of running k-NN with NCA but without any dimensionality reduction, we see that the accuracy scores are almost the same (compare Figure 3.33 with Figure 3.46). The macro F1-scores tell a more complicated story. We see a significant improvement for the distiluse model, raw data, but a significant deterioration for the processed data. We also see a slight deterioration for the para-mini model, raw data and for the para-base model, raw data (compare Figure 3.34 to Figure 3.47).

A possible conclusion would be that reducing the number of dimensions did not produce a better result overall. On the contrary, after performing dimensionality reduction with NCA and then running k-NN, we have observed significantly lower performance for certain embeddings and data sets. Performing NCA without any dimensionality reduction before running k-NN, however, has led to a significant improvement in the performance of two sets of embeddings (distiluse and para-mini), and a significant loss of performance in the third set of embeddings (para-base) (see section 3.5).

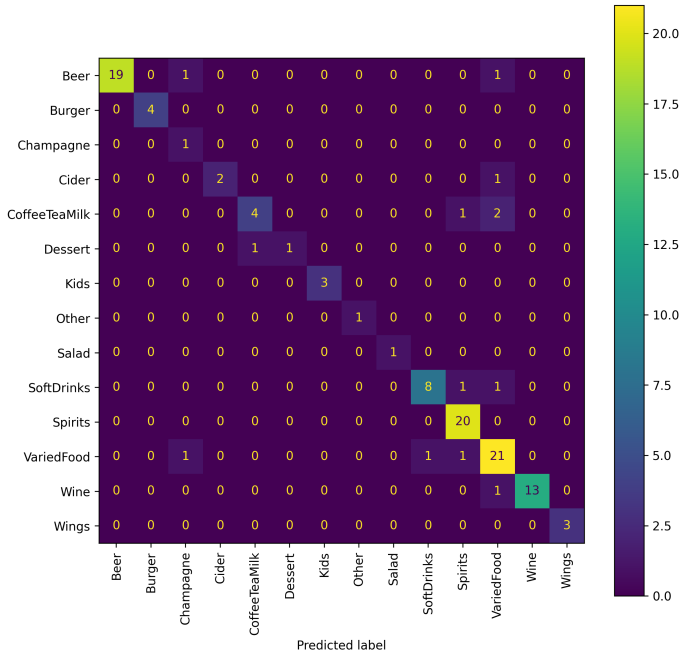

Figure 3.40: Confusion matrix after NCA dimensionality reduction and k-NN, distiluse embeddings, raw data.

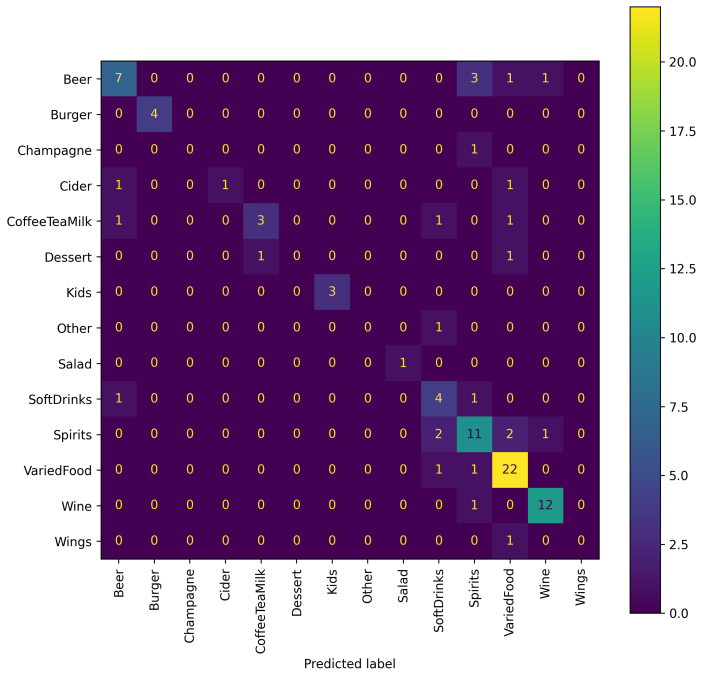

Figure 3.41: Confusion matrix after NCA dimensionality reduction and k-NN, distiluse embeddings, processed data.

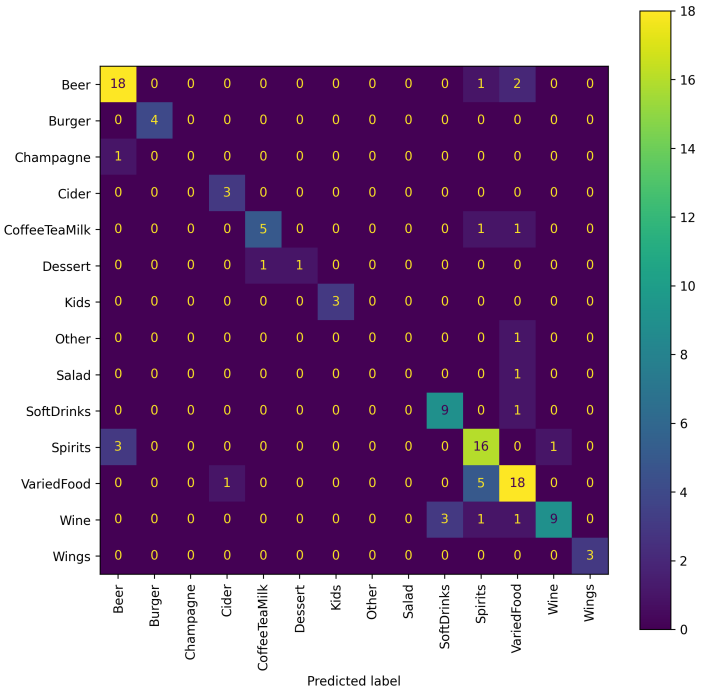

Figure 3.42: Confusion matrix after NCA dimensionality reduction and k-NN, para-mini embeddings, raw data.

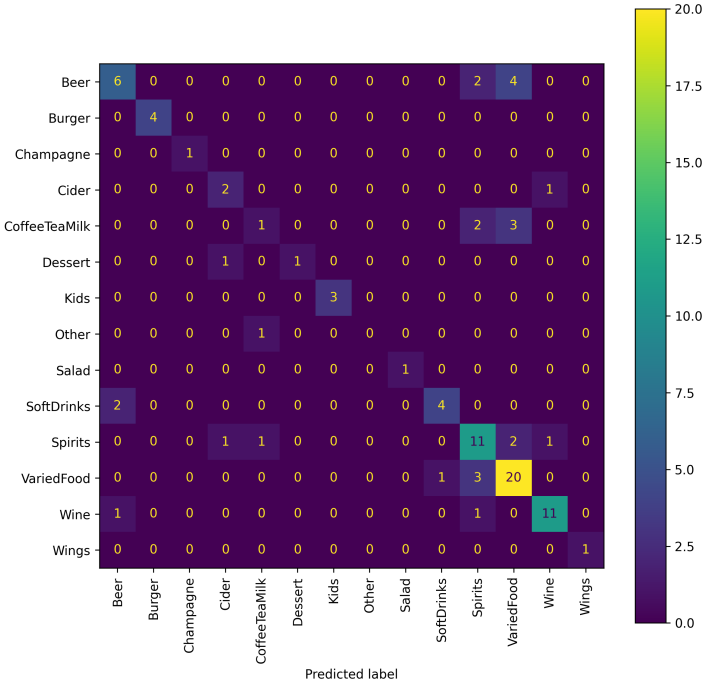

Figure 3.43: Confusion matrix after NCA dimensionality reduction and k-NN, para-mini embeddings, processed data.

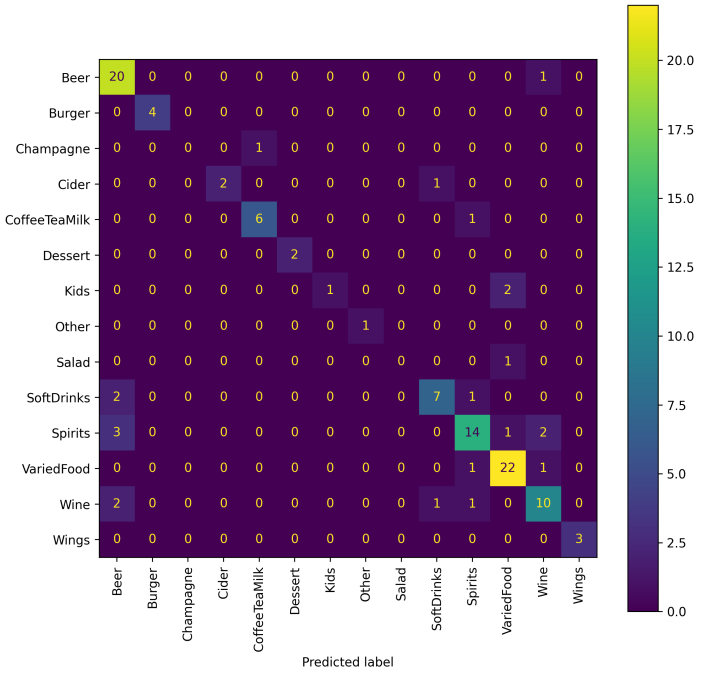

Figure 3.44: Confusion matrix after NCA dimensionality reduction and k-NN, para-base embeddings, raw data.

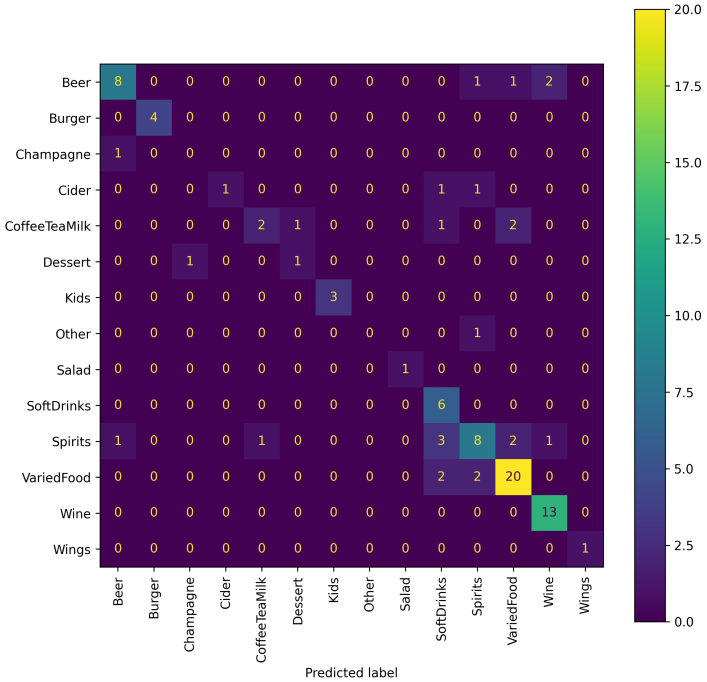

Figure 3.45: Confusion matrix after NCA dimensionality reduction and k-NN, para-base embeddings, processed data.

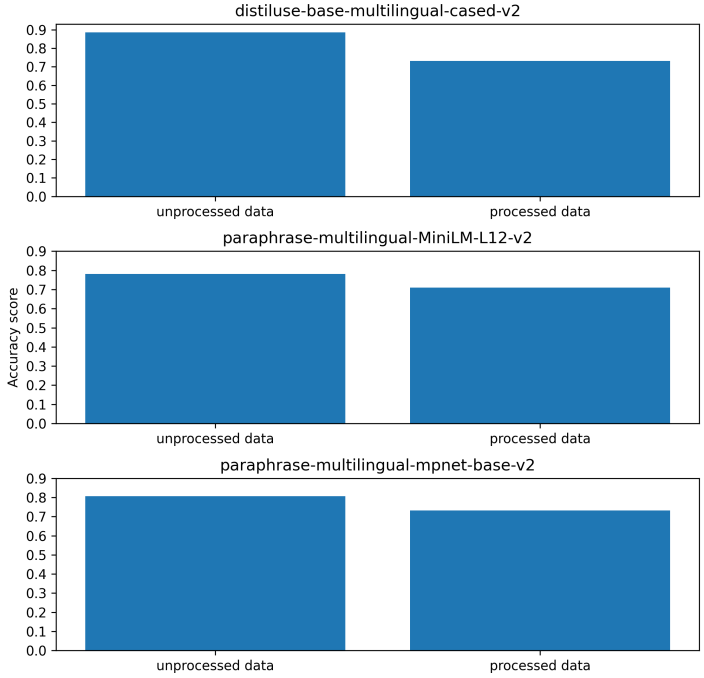

Figure 3.46: Accuracy score after NCA dimensionality reduction and k-NN.

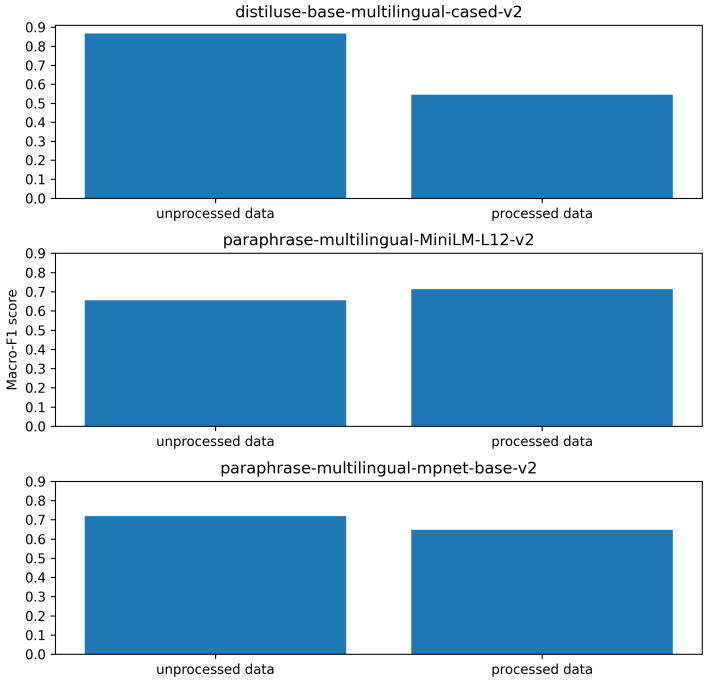

Figure 3.47: Macro F1-score after NCA dimensionality reduction and k-NN.

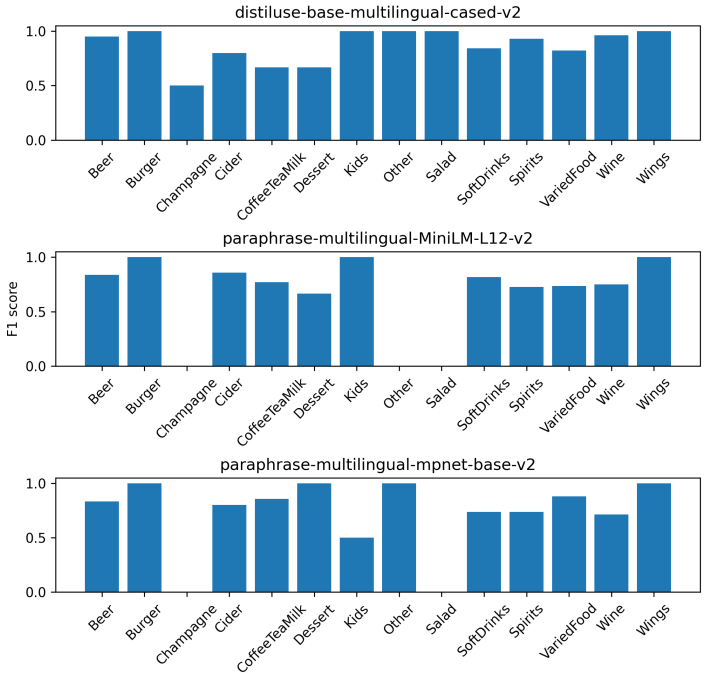

Figure 3.48: F1-score per category after NCA dimensionality reduction and k-NN, raw data.

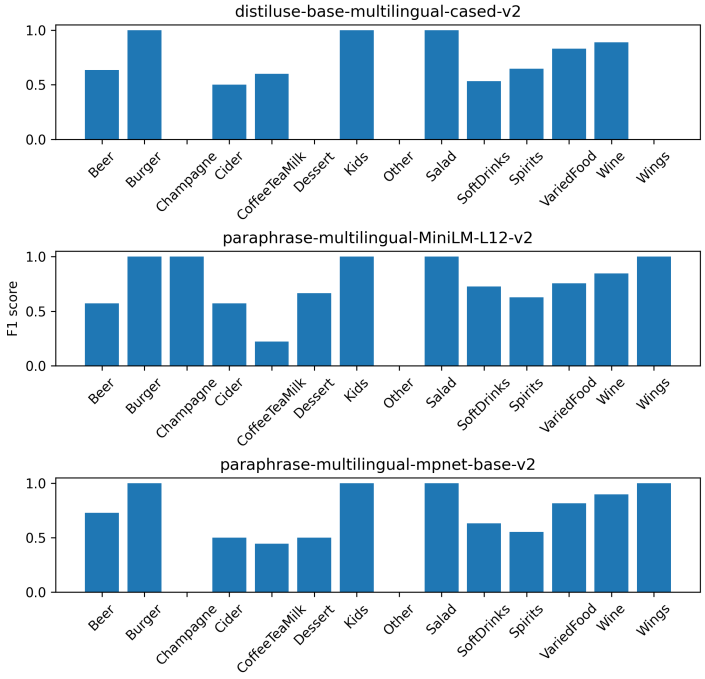

Figure 3.49: F1-score per category after NCA dimensionality reduction and k-NN, processed data.

Chapter 4

Related work

NLP is a field of machine learning that has been growing and with that the amount of literature on various research topics and problems within NLP is becoming larger every day. Here we present some of the research we have found most interesting and most closely related to our particular problem.

In 2017, a similar problem to the one we tried to solve in this project was tackled by a student at the Department of Information Technology at Uppsala University as a Bachelor Thesis research. That problem was also commissioned by Caspeco and it falls into the domain of NLP. The problem that Caspeco had at that time was that different restaurants could add their own articles to their checkout system. They wanted to find a way to map all the different versions of the same article to one common article, e.g., 'Carlsberg Hof 1L' and 'Carlsberg Hof Fat 50cl' are both the same article, i.e., 'Carlsberg Hof'. The author of the thesis used the Levenshtein distance and the NYSIIS distance to compare the different representations, and ran various clustering algorithms to see if the different representations could be grouped together[Hed17].

The same year, a researcher from the University of North Texas explored various ML approaches to classify common names in library metadata. The author of the paper used several classification algorithms, e.g., naive Bayes, random forest and support vector machines, and achieved a high accuracy of 99%[PC17].

Also in 2017, a group of researchers from Sogang University in South Korea developed a recurrent-neural-network-based model which predicts nationalities of personal names using feature extraction. They worked with a raw and cleaned data set throughout, which was a direct inspiration to our approach. They achieved accuracy scores ranging from 81.7% to 98.7%[Lee+17].

In 2021, a group of researches from various fields and universities built a classifier model to predict takeaway food outlets by cuisine type from the business name alone. They used a Long Short Term Memory variant of a recurrent neural network. They managed to get a correct prediction approximately three out of four times[Bis+21].

This is just a fraction of the relevant literature that deals with similar problems to ours but these papers have had the most direct and meaningful impact on our work.

Chapter 5

Discussion

In this chapter we summarise and discuss the results we arrived at and the difficulties we faced during the course of this project.

In the first part of the project we compared the quality of clusters after running K-means and hierarchical clustering on 3 different sets of sentence embeddings, produced by 3 different multilingual pre-trained SBERT models. We used the Silhouette Score, the Calinski-Harabasz Index and the Davies-Bouldin Index, as well as a visual, qualitative inspection of the clusters, to try and determine if any of the 3 sets of sentence embeddings would give us an outstanding result. The results were inconclusive and we were not able to pick a model that produced embeddings of a superior quality for our specific data sets, both raw and processed.

The difficulties we faced in this part of the project are mostly related to the fact that the results were inconclusive. The metrics we used all agreed on the fact that the clusters were overlapping and the qualitative inspection confirmed this, but we were hoping to find a pre-trained model that would stand out in terms of clustering performance.

In the second part of the project we attempted to automatically classify the items into predetermined groups. First we used a custom classifier. Then we ran k-NN and NCA, both before and after dimensionality reduction, and we achieved a good degree of accuracy on most data sets. There were some differences between the 3 different sets of embeddings, but they all performed well in one or another setting. There was a clear performance advantage with the raw data, and we were able to come to the conclusion that raw data should be preferred over processed data for classification purposes.

In this part of the project we faced the dilemma of how to classify items manually so that we establish a ground truth that is both reliable and has the potential to be generalized to other data from other restaurants. We are still unsure whether we succeeded on that front, as many different classification systems would be possible. We only worked with the data from one specific restaurant in Uppsala. The question of how our manual classification and the classifiers we implemented would generalise to other restaurants therefore remains unanswered.

Chapter 6

Conclusions and future work

At the end of this project, there are several conclusions we can draw, and some that we wish we could make but cannot. In the latter group, we have the inconclusive results from the first part of the project. We were unfortunately unable to find a pre-trained model that would give us sentence embeddings which would lead to outstanding clustering results. A possible extension of this project would therefore be to train our own model (or fine-tune one of the existing SBERT ones)[Reih]. The benefits and difficulties of this possible extension are discussed in subsection 1.2.5.

Another thing that we cannot be sure about is whether our work can be generalised to other data. We only used data from one particular restaurant in Uppsala and our manual classification was made to fit their particular menu. How would this fare with other data is a very important question and therefore a possible and desirable extension of this project.

Some of the conclusions we can make, however, are the following. Firstly, we have seen a tendency for the raw data to outperform the processed data in the classification part of the project. This we have seen with all the classification algorithms and with all sentence embeddings, so we can safely conclude that the extra bits of information that come from including the quantities alongside the names of the products are actually very beneficial to this downstream task. Secondly, we can safely say that building an automated classification system based solely on the item names for this specific restaurant is possible and could be implemented in practice. The classification results were rather good, some of them over 85% on the macro F1-score, and with more time we could probably fine-tune the classifier even further.

Another possible extension would be to prepare several different manual classifications, and compare the results of the predictions. How would the k-NN algorithm perform if we had fewer classes of menu items, or if the classes were more balanced in terms of value counts? What if we had more classes than the 14 we came up with for this particular data set? We mentioned the difficulties connected to the manual classification and ground truth several times in this thesis, and we mention it again as we cannot overstate the importance of this question.

Last but not least, there remains the question of how other classification algorithms would perform with this data. Yet another possible extension of this work could therefore be implementing a random forest classifier, or any other multi-class classifier for that matter, and compare the

performances. Perhaps the results we achieved here could be further improved. We leave this work to those that come after us.

FINAL COMMENTS

- I have loosely followed the MLA formatting guidelines.
- The report is bloated due to all the figures and tables I have included (there are only 32 pages of pure text). I could either remove some of the figures (for example, some of the confusion matrices), or remove an entire section (for example, hierarchical clustering)?
- What is the best course of action for the remaining weeks? Should I add more material, e.g., implement another classifier, test on more data? Or should I simply work on the quality of the text, adding more technical details, attempting to explain the results in a more thorough way?

Bibliography

[Agg] Sangeet Aggarwal. *K-Nearest Neighbors*. URL: https://towardsdatascience.com/k-nearest-neighbors-94395f445221. (accessed: 30.04.2022).

[AHK01] *On the Surprising Behavior of Distance Metrics in High Dimensional Spaces*. ICDT '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 420–434. ISBN: 3540414568.

[Bis+21] Tom R.P. Bishop et al. "Automatic classification of takeaway food outlet cuisine type using machine (deep) learning". In: *Machine Learning with Applications* 6 (2021), p. 100106. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100106. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000530.

[Cer+18] Daniel Cer et al. *Universal Sentence Encoder*. 2018. DOI: 10.48550/ARXIV.1803.11175. URL: https://arxiv.org/abs/1803.11175.

[Der16] *Complementarity, F-score, and NLP Evaluation*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 261–266. URL: https://aclanthology.org/L16-1040.

[Dev+18] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[Emm] Chris Emmery. *Euclidean vs. Cosine Distance*. URL: https://cmry.github.io/notes/euclidean-v-cosine. (accessed: 27.04.2022).

[GBV20] Margherita Grandini, Enrico Bagli, and Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. DOI: 10.48550/ARXIV.2008.05756. URL: https://arxiv.org/abs/2008.05756.

[Gol+04] L. Saul, Y. Weiss, and L. Bottou, eds. *Neighbourhood Components Analysis*. Vol. 17. MIT Press, 2004. URL: https://proceedings.neurips.cc/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf.

[Grb] Mihajlo Grbovic. *Listing Embeddings in Search Ranking*. URL: https://medium.com/airbnb-engineering/listing-embeddings-for-similar-listing-recommendations-and-real-time-personalization-in-search-601172f7603e. (accessed: 27.04.2022).

[Hed17] Filip Hedman. *Machine Learning For Automated Categorization of Product Articles*. 2017. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-347393.

[IBMa] IBM. *Machine Learning*. URL: https://www.ibm.com/cloud/learn/machine-learning. (accessed: 27.04.2022).

[IBMb] IBM. *Overfitting*. URL: httpshttps://www.ibm.com/cloud/learn/overfitting. (accessed: 29.04.2022).

[IBMc] IBM. *Supervised Learning*. URL: https://www.ibm.com/cloud/learn/supervised-learning. (accessed: 27.04.2022).

[IBMd] IBM. *Unsupervised Learning*. URL: https://www.ibm.com/cloud/learn/unsupervised-learning. (accessed: 27.04.2022).

[Jai] Deepak Jain. *Cross-validation using KNN*. URL: https://towardsdatascience.com/cross-validation-using-knn-6babb6e619c8. (accessed: 30.04.2022).

[Kar] Shashmi Karanam. *Curse of Dimensionality — A 'Curse' to Machine Learning*. URL: https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb. (accessed: 27.04.2022).

[Koz] Cassie Kozyrkov. *What is "Ground Truth" in AI? (A warning.)* URL: https://towardsdatascience.com/in-ai-the-objective-is-subjective-4614795d179b. (accessed: 27.04.2022).

[leaa] Scikit learn. *Clustering performance evaluation*. URL: https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation. (accessed: 27.04.2022).

[leab] Scikit learn. *Cross-validation: evaluating estimator performance*. URL: https://scikit-learn.org/stable/modules/cross_validation.html. (accessed: 30.04.2022).

[leac] Scikit learn. *Dimensionality Reduction with Neighborhood Components Analysis*. URL: https://scikit-learn.org/stable/auto_examples/neighbors/plot_nca_dim_reduction.html. (accessed: 30.04.2022).

[lead] Scikit learn. *Hierarchical clustering*. URL: https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering. (accessed: 28.04.2022).

[leae] Scikit learn. *K-means*. URL: https://scikit-learn.org/stable/modules/clustering.html#k-means. (accessed: 27.04.2022).

[leaf] Scikit learn. *Metrics and scoring: quantifying the quality of predictions*. URL: https://scikit-learn.org/stable/modules/model_evaluation.html. (accessed: 29.04.2022).

[leag] Scikit learn. *Neighborhood Components Analysis*. URL: https://scikit-learn.org/stable/modules/neighbors.html#nca. (accessed: 30.04.2022).

[leah] Scikit learn. *Principal component analysis (PCA)*. URL: https://scikit-learn.org/stable/modules/decomposition.html#pca. (accessed: 28.04.2022).

[leai] Scikit learn. *sklearn.metrics.DistanceMetric*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.DistanceMetric.html#sklearn.metrics.DistanceMetric. (accessed: 30.04.2022).

[leaj] Scikit learn. *Supervised learning: predicting an output variable from high-dimensional observations*. URL: https://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html. (accessed: 29.04.2022).

[Lee+17] *Name Nationality Classification with Recurrent Neural Networks*. IJCAI'17. Melbourne, Australia: AAAI Press, 2017, pp. 2081–2087.

[PC17] Mark Phillips and Jiangping Chen. "Machine learning for name type classification in library metadata". In: *Proceedings of the Association for Information Science and Technology* 54 (Jan. 2017), pp. 773–774. DOI: 10.1002/pra2.2017.14505401152.

[Reia] Nils Reimers. *Augmented SBERT*. URL: https://www.sbert.net/examples/training/data_augmentation/README.html. (accessed: 27.04.2022).

[Reib] Nils Reimers. *Multi-Lingual Models*. URL: https://www.sbert.net/docs/pretrained_models.html#multi-lingual-models. (accessed: 27.04.2022).

[Reic] Nils Reimers. *paraphrase-multilingual-mpnet-base-v2*. URL: https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2. (accessed: 27.04.2022).

[Reid] Nils Reimers. *Pretrained Models*. URL: https://www.sbert.net/docs/pretrained_models.html. (accessed: 27.04.2022).

[Reie] Nils Reimers. *SBERT Semantic Textual Similarity*. URL: https://www.sbert.net/docs/usage/semantic_textual_similarity.html. (accessed: 27.04.2022).

[Reif] Nils Reimers. *Sentence Transformers*. URL: https://huggingface.co/sentence-transformers. (accessed: 27.04.2022).

[Reig] Nils Reimers. *SentenceTransformers Documentation*. URL: https://www.sbert.net. (accessed: 27.04.2022).

[Reih] Nils Reimers. *Training Overview*. URL: https://www.sbert.net/docs/training/overview.html. (accessed: 27.04.2022).

[RG19] *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084.

[Staa] Josh Starmer. *StatQuest: K-means clustering*. URL: https://www.youtube.com/watch?v=4b5d3muPQmA. (accessed: 27.04.2022).

[Stab] Josh Starmer. *StatQuest: K-nearest neighbors, clearly explained*. URL: https://www.youtube.com/watch?v=HVXime0nQeI&t=69s. (accessed: 30.04.2022).

[Tha+20] Nandan Thakur et al. *Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks*. 2020. DOI: 10.48550/ARXIV.2010.08240. URL: https://arxiv.org/abs/2010.08240.

[Uni] Cornell University. *Curse of Dimensionality*. URL: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html. (accessed: 30.04.2022).

[Wik22a] Wikipedia contributors. *Cosine similarity — Wikipedia, The Free Encyclopedia*. [Online; accessed 27-April-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=1071254454.

[Wik22b] Wikipedia contributors. *Principal component analysis — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-April-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Principal_component_analysis&oldid=1070303429.

[Yan+19] Yinfei Yang et al. *Multilingual Universal Sentence Encoder for Semantic Retrieval*. 2019. DOI: 10.48550/ARXIV.1907.04307. URL: https://arxiv.org/abs/1907.04307.

[Zuca] Eugenio Zuccarelli. *Performance Metrics in Machine Learning — Part 1: Classification*. URL: https://towardsdatascience.com/performance-metrics-in-machine-learning-part-1-classification-6c6b8d8a8c92. (accessed: 29.04.2022).

[Zucb] Eugenio Zuccarelli. *Performance Metrics in Machine Learning — Part 3: Clustering*. URL: https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6. (accessed: 27.04.2022).

Appendix A

5 clusters after K-means, distiluse embeddings, raw data

Cluster 1

[’Kids Pancakes’, ’Grilled Chicken Quesadilla’, ’Fries’, ’Hot Crispy Burger’, ’Cheddar Cheeseburger’, ’Bacon Bbq Burger’, ’Buffalo Wings 5 pcs’, ’Parmesan Garlic Wings 5 pcs’, ’Fried Halloumi Bites’, ’Garlic And Cheese Bread’, ’Chili Cheeseburger’, ’Hot Cheddar’, ’Parmesan Garlic Wings 10 pcs’, ’Giant Double Cheeseburger’, ’Classic Steak Oxfilé’, ’Buffalo Wings 10 pcs’, ’Herb Butter Grilled Salmon’, ’Baconnaise Burger’, ’Truffle Oozing Burger’, ’Beef And Cheese Nachos’, ’Chili Mayo’, ’Kids Cheeseburger’, ’Kids Chicken Nuggets’, ’Truffle Mayo’, ’Parmesan Garlic’, ’Grilled Chicken Club Sandwich’, ’Jalapeno Oumph Burger (Vegan)’, ’Deep Fried Brownie’, ’Creamy Beef Tenderloin Pasta’, ’Classic Ceasar Salad’, ’Blue Cheese’, ’Byt till Sweet Potato Fries’, ’Bufflo Wings 15 pcs’, ’Classic Steak Ryggbiff’, ’Parmesan Garlic Wings 15 pcs’, ’Sweet Potato Fries’, ’Byt till Cheesy Fries’, ’Add Bacon’, ’Add Garlic Bread’, ’Parmesan And Garlic Ribs’, ’Camarones Rojos Fritos’, ’Tacos Del Mar’, ’Plato De Carne’, ’Tacos Vegetariano’, ’Ceviche de Pescado’, ’Carbon Frito’, ’Mozzarella Sticks 2 pcs’, ’Cheesy Fries’, ’Krossad potatis’, ’Onion Rings 2 pcs’, ’Chili Cheese Burger’, ’Cheddar Cheese Burger’, ’Caesar Salad’, ’Hot Buffalo Chicken Wings - 15 PCS’, ’Blue Cheese Dressing’, ’Bourbon BBQ Ribs’, ’Lightly Salted Fries’, ’Beef Cheese Nachos’, ’Parmesan & Garlic Sauce’, ’Bacon BBQ Burger’, ’Halloumi Fries’, ’Fish & Chips’, "O’Learys BBQ Sampler", ’Garlic Bread’, ’Hot Buffalo Chicken Wings - 5 PCS’, ’Classic Steak - Striploin’, ’Pulled Beef & Bacon Burger’, ’Grilled Chicken Quesadillas’, ’Classic Steak - Tenderloin’, ’Hot Buffalo Chicken Wings - 10 PCS’, ’Onion Ring - 2 PCS’, ’PKT Parmesan Garlic Wings 5pcs’, ’Giant Double Cheese Burger’, ’PKT Parmesan Garlic Wings 10pcs’, ’Side Tomato Salad’, ’Sub Fries to Loaded Fries’, ’Sub Fries to Sweet Potato Fries’, ’Salty Apple Crumble’, ’Parmesan Garlic Wings 15pcs’, ’Add Bacon - Burger’, ’Side Caesar Salad’, ’Mozzarella Stick - 2 PCS’, ’BBQ Grilled Chicken’, ’Jalapeno Oumph Burger’, ’Pickled CuCumber’, ’Sub Fried Chicken’, ’Crispy Halloumi Burger’, ’Bröd med Dip’, ’Hamburgerdressing’, "Kids Fish N’Chips", ’Kids Vegetables’, ’El Cerdo Maravilloso’, ’Parmesan & Garlic Ribs’, ’Dubble Up Your Burger’, ’Extra Cheese’]

Cluster 2

['Carlsberg Export 50 cl', 'Guinness 50cl', 'Staropramen 50 cl', 'Pepsi 50cl', 'Ramlösa 33 cl', 'Carlsberg Export 40 cl', 'Pepsi Max 50cl', 'Brooklyn Lager 50cl', 'Stam öl 50 cl', 'Brooklyn EIPA 50cl', 'Carlsberg Export 60 cl', 'Staropramen 60 cl', 'Stam öl 60 cl', 'Carlsberg Hof 50cl', 'Somersby Apple Cider Non Alc. 33 cl', 'Stam öl 40 cl', 'Zingo 40cl', 'Juice 40 cl', '7-Up 50cl', 'Brooklyn EIPA 40cl', 'Brooklyn Defender 50 cl', 'Mjölk 40 cl', 'Brooklyn EIPA 60cl', 'Pepsi 40cl', '1664 Blanc 33 cl', 'Zingo 50cl', 'Brooklyn Ale 40 cl', 'Pepsi Max 40cl', '7-Up 40cl', 'San Miguel 33cl', '1664 Blanc', 'Brooklyn Lager 40 cl', 'Kronenbourg Blanc 40 cl', 'Staropramen 40 cl', 'Carlsberg Hof 60cl', 'Pepsi Max 60cl', 'Brooklyn Lager 40cl', 'Falcon Export 50cl', 'Staropramen 50cl', 'Brooklyn Lager, 50cl', '7-up 50cl', 'Brooklyn Defender 50cl', 'Somersby Pear Cider 33cl', 'Kronenburg 1664 Blanc 33 cl', 'Ramlösa 33cl', 'Falcon Export 60cl', 'Falcon Export 40cl', 'Ramlösa citrus, 33cl', 'Brooklyn Lager, 40cl', 'Eriksberg 50cl', 'Staropramen 60cl', 'Stam 50cl', 'Staropramen, 50cl', 'Erdinger Hefe 50cl', 'Andiamo Red 75cl', 'Somersby Pear 33cl', 'Lagunitas IPA 35,5cl', 'Carlsberg Export 33 cl', 'Juice 50cl', 'Brooklyn Lager 60cl', 'Gnarly Head Old Vine Zinfandel, 75cl', 'Pepsi 40 cl', 'Carlsberg Hof 40cl', 'Andiamo White 75cl', 'Guinness 60cl', 'Grimbergen Blonde 33cl', 'Brooklyn Defender 60cl', 'Gnarley Chard 75cl', 'Cidraie Original Organic 33cl', 'Brooklyn Defender 40cl', 'Staropramen 40cl', 'Juice 40cl', 'Puycherich Syrah Rose 75cl', 'Pepsi Max 30cl', 'Brugal 1888', 'Brooklyn Bel Air 35,5 cl', 'Staropramen, 40cl', 'Ramlösa 80cl', '7up 50cl', 'Andiamo Rose 75cl', 'Eriksberg Karaktär 33 cl', '7up 40cl', 'Guinness 40cl', 'Kronenbourg Blanc 40cl', 'Kronenbourg Blanc 50cl', 'Kronenbourg Blanc 60cl', 'Juice 60cl', 'Carlsberg Export 50cl', 'Carlsberg Export 60cl', 'Carlsberg Export 40cl']

Cluster 3

['Erdinger Hefe', 'Bowling', 'Oumph', 'Aioli', 'Fish N´Chips', 'Kaffe', 'Popcorn', 'Combo Plate', 'Öppen Sprit', 'Nachos', 'Eriksberg', 'Stam Vitt', 'Stam Rött', 'Red Bull', 'Kaffe m. Mjölk', 'Ice Cream Sundae', 'Chocolate Brownie', 'Guacamole', 'Kids Quesadilla', 'Öppen Mat', 'KIds Brownie', 'Kids Sundae', 'Öppen dryck alkfri', 'Öppen läsk', 'Shuffle', 'Majonäs', 'Snacks', 'Öppen Vin', 'Stam Bowling', 'Öppen Kaffedrink', 'Espresso Dubbel', 'Cappucino', 'Bearnaise', 'Embutido Con Trufa', 'Coliflor Frita', 'Maiz a la Parrilla', 'Pepsi Max', 'Öppen Öl', 'Pepsi', 'Te', 'Rosquilla Frita', 'Nonno´s Favorito', 'El Cubano', 'Corona', 'Corona ', 'Kroketter', 'Pimientos De Padron', 'Vegetariano', 'Zingo', 'Brule De Crema', 'Cappuccino', 'Dubbel Espresso', 'Bryggkaffe', 'Espresso', 'Oumph Nahos', 'Milkshake ', 'The Ortiz', 'Cava GL', 'Helado', 'Stam ÖL', 'Manzana Fresca', 'Alex 1', 'Alex 2', 'Violeta', 'Kaffedrink', 'Stam VIN', 'Alkofri Drink', 'Caipirinha de Pîna', 'Stam DRINK', 'Espresso Martini', 'Öppen sprit', 'Coleslaw', 'Hot Shot', 'Läsk', 'Öppen Bowling', 'Stam Öl', 'Personalöl', 'Kampanjvin Glas', 'Öppen Event', 'Stam Cider', 'Öppen mat', 'Facundo Eximo', 'Tea', 'Coffee', 'Oumph Quesadilla', 'Espresso Ice Cream', 'Double Espresso', 'Kaffe Mjölk', 'Caffe Latte', 'Öppen Läsk', 'Alkoholfri drink', 'Mjölk', 'Tullamore Coffe', 'Kids Brownie', 'Sub Salmon', 'Chocolate Milkshake', 'Potatispure', 'Personal Shot', 'Jack Daniels', 'Mezcalita', 'Gräddfil', 'Café con chocolate', 'Patron', 'Bacardi Blanca', 'Öppet Vin', 'Macchiato', 'Side Sallad',

'Amargo Dulce', 'Strawberry Milkshake', 'Dubbel Macchiato', 'SNACKS', 'Taittinger', 'Öppen Rom', 'Absinthe', 'Mån Whisky', 'Chilibearnaise', 'Presentkort', 'Personal Cider', 'Cortado', 'Öppen Aktivitet']

Cluster 4

['Drink 4 cl', 'Drink 4 cl 119kr', 'Kalaspaket 1', 'Shot 6cl', 'Stam Shot 6 cl', 'Brooklyn Special Effects Non Alco 0,4%', 'Drink 6 cl 163kr', 'Drink 6 cl', '4 cl Boulard Grand/Knob Creek/Woodford Reserve', 'Stam Shot 4 cl', 'Drink 4 cl 129kr', 'Stam Drink 4 cl', 'Stam Drink 6 cl', 'Pripps Blå I (Lättöl 2,2% Alkohol)', 'Tullamore Irish Coffee 6 cl', '7up', '6 cl Ron Zacapa', 'Shot 4cl', 'Drink 1', 'El Mariachi 2', 'Drink 2', 'Shot 2cl', '6 cl Brugal 1888/Macallan Double Cask', '3 cl Brugal 1888/Macallan Double Cask', '4 cl Highland Park/Grönstedts VS/Laphroaig', '4 cl Baileys/Sambuca/Jim Beam/Famous Grouse', '4 cl Brugal 1888/Macallan Double Cask', 'Zacapa Centenario 23yo', 'Hernö Gin & Tonic 6 cl', 'Shots 4cl', 'Andiamo White 18cl', 'Drink 4cl', 'Stam Drink 4cl', 'Aperol Spritz 4cl', 'Redbull&Vodka 4cl', 'Andiamo Red 18cl', 'Tullamore Coffe 6cl', 'Gnarly Head Old Vine Zinfandel, 18cl', 'Puycherich Syrah Rose 18cl', 'Drink 6cl', 'Jamaican Coffee 4cl', 'Stammis Shot 4cl', 'Pripps Blå 1', 'Shots 6cl', 'Anejo Cuatro', 'Hernö Gin och Tonic 4cl', 'Mojito 4cl', 'Jäger-Redbull Dr 4cl', 'Laphroaig 10y', 'Redbull-Vodka 6cl', 'Personal Drink 4cl', 'Shot 2 cl', 'Espresso Martini de laysa 2', 'Gin ´n Tonic 6cl', 'Gnarley Chard 18cl', 'Andiamo Rose 18cl', '7-up Liten', 'Mojito 6cl', 'Ron Zacapa Centario 23', 'Jim Beam Sour 4cl', 'Lynchb.Lemonade 4cl', 'Gin ´n Tonic 4cl', 'Frozen Daiquiri 4 cl', 'Cava mas pere brut 1/1', 'Jäger-Redbull Dr 6cl', 'Margarita 4cl', 'Highland Park 12y', 'Moscow Mule 6cl', 'Moscow Mule 4cl', '1/2 Vitt vin', '1/2 Rött vin', 'Aperol Spritz 6cl', 'Espresso Martini 6cl', 'Lynchb.Lemonade 6cl', 'Hernö Gin och Tonic 6cl', 'Jamaican Coffee 6cl', 'Nicolas F Brut Resv 1/1', 'Jim Beam Sour 6cl', 'Personal Drink 6cl']

Cluster 5

['Ceasar Byt Ut Kyckling Till', 'Somersby Pear Cider', 'Cava Mas Pere Selecció Brut (ESP) FL', 'Fever Tree Indian Tonic', 'Eriksberg Karaktär', 'Husets RÖDA Andiamo (ITA) GL', 'Bourbon Bbq Ribs', 'Husets VITA Andiamo (ITA) GL', 'Bbq Plate', 'Centenero Ripasso Valpolicella (ITA) GL', 'Strawberry Lemonade Mocktail', 'Carlsberg Non Alc.', 'Husets ROSÉ Andiamo (ITA) GL', 'Cava Mas Pere Selecció Brut (ESP) GL', 'Gnarly Head Old Vine ZINFANDEL (USA) GL', 'Somersby Apple Organic', 'Carlsberg Export Pitcher', 'Husets RÖDA Andiamo (ITA) FL', 'Baron De Ley Club Privado Rioja (ESP) FL', 'Bolas De Flote', 'Lagunitas IPA', 'Castellanovo GL', 'Lomo Bajo (Biff)', 'Carnitas Con Aguacate', 'Samuel Adams Lager', 'Paso A Paso Verdejo GL', 'Husets VITA Andiamo (ITA) FL', 'Honoro Vera FL', 'Carlsberg Alcohol Free', 'Plato De Pescado', 'Somersby Apple NAB', 'Staropramen Pitcher', 'Paso a Paso GL', 'Lomo Alto (Entrecote)', 'Carajillo Cubano', 'Pisco Sour de Camila', 'Morgan Bay Cab FL', 'Boulard Cidre Biologique', 'Paso a Paso FL', 'Morgan Bay Cab GL', 'Margarita el diabolo', 'Paso A Paso Verdejo FL', 'Brooklyn Lager', 'Societa Chianti GL', 'Huber Terassen GL', 'Mi Abuela la Loca', 'Huber Terrassen FL', 'EXTRA CARNITAS 1ST', 'Morgan Bay Chard GL', 'Gris De Trop GL', 'Honoro Vera GL', 'Cours De Dame Blanc FL', 'Cours De Dame Blanc GL', 'Villa Conchi FL', 'Villa Conchi GL', 'Alkoholfritt Vin GL', 'Gnarly Head Old Vine

ZINFANDEL (USA) FL', 'Castellanovo FL', 'Brooklyn Defender Pitcher', 'Carlsberg Export', 'The Grinder Shiraz GL', 'Gnarly Head CHARDONNAY (USA) GL', 'Sub Oumph Caesar', 'Brooklyn Special Effects', 'Cava mas pere brut, glas', 'Jalapeno-Lime Mayo', 'Cotes De Rhone Villages FL', 'Brooklyn Naranjito', 'Estrella Galicia', 'Pina Pina Alex', 'Hunky Dory Sav FL', 'Carlsberg Non Alco', 'Somersby Non Alco', 'Falcon Export pitcher', 'Super Combo Plate', 'Boulard Cidre De Normandie', 'Staropramen pitcher', 'Garage Hard Lemon', 'Jalapeno Popper - 2 PCS', 'Sigtuna NAPA non alc', 'Kronenbourg Blanc NAB', 'Diplomatico Reserva', 'Brooklyn Summer Ale', 'Barolo Ratti FL', 'Ocean Spray Cranberry', 'Pepsi Maxi stor', 'Pepsi Stor GD', 'Barceló Imperial Onyx', 'Grönstedt VS CC', 'The Macallan Fine Oak 12y', 'Jim Beam Bourbon CC', 'Gentleman Jack Daniels', 'Carlsberg Hof pitcher', 'Brooklyn Lager pitcher', 'Galliano Martini', 'Tapiz Cabernet Sauvignon FL', 'Grönstedts Monopole VSOP', 'Maria Bonita GL', 'Morgan Bay Chard FL', 'ITA Prosecco GL', 'El Dorado 15yo', 'Long Island Iced Tea', 'Woodford Reserve', 'Societa Chianti FL', 'Brooklyn EIPA Pitcher', 'Zingo Stor GD', 'The Grinder Shiraz FL', 'Gris De Trop FL', 'EA TINTO FL', 'Boulard Grand Solage Calvados', 'Collet Chablis FL', 'EA TINTO GL', 'Grönstedts XO', 'Cerveza Alhambra', 'Knob Creek CC', 'Lander Jenkins Pinot FL', 'Alkoholfritt Vin FL', 'Personal Vitt Glas', 'Angostura 1919', 'Anima Negra FL', 'Martell VS', 'Martell VSOP', 'EXTRA TACO DEL MAR 1ST', 'EXTRA TAVO VEG 1ST', 'Les Cardounettes Blanc EKO (FRA) GL']

Appendix B

5 clusters after K-means, distiluse embeddings, processed data

Cluster 1

[’drink’, ’stam öl’, ’öppen sprit’, ’somersby pear cider’, ’somersby apple cider non alc.’, ’juice’, ’strawberry lemonade mocktail’, ’öppen dryck alkfri’, ’öppen vin’, ’stam drink’, ’öppen öl’, ’carlsberg alcohol free’, ’stam öl’, ’stam vin’, ’alkofri drink’, ’stam drink’, ’alkoholfritt vin gl’, ’öppen sprit’, ’stam öl’, ’stam cider’, ’ramlösa citrus’, ’redbull&vodka’, ’somersby non alco’, ’hernö gin och tonic’, ’alkoholfri drink’, ’personal drink’, ’lynchb.lemonade’, ’bacardi blanca’, ’öppet vin’, ’vitt vin’, ’rött vin’, ’cerveza alhambra’, ’lynchb lemonade’, ’alkoholfritt vin fl’, ’personal vitt glas’, ’absinthe’, ’mån whisky’, ’personal cider’]

Cluster 2

[’carlsberg export’, ’ceasar byt ut kyckling till oumph’, ’brooklyn lager’, ’brooklyn eipa’, ’fried halloumi bites’, ’carlsberg hof’, ’cava mas pere selecció brut (esp) fl’, ’fever tree indian tonic’, ’eriksberg karaktär’, ’husets röda andiamo (ita) gl’, ’bourbon bbq ribs’, ’husets vita andiamo (ita) gl’, ’centenero ripasso valpolicella (ita) gl’, ’carlsberg non alc.’, ’brooklyn defender’, ’brooklyn special effects non alco’, ’husets rosé andiamo (ita) gl’, ’cava mas pere selecció brut (esp) gl’, ’1664 blanc’, ’gnarly head old vine zinfandel (usa) gl’, ’somersby apple organic’, ’boulard grand/knob creek/woodford reserve’, ’husets röda andiamo (ita) fl’, ’baron de ley club privado rioja (esp) fl’, ’coliflor frita’, ’brooklyn ale’, ’carnitas con aguacate’, ’samuel adams lager’, ’paso a paso verdejo gl’, ’husets vita andiamo (ita) fl’, ’honoro vera fl’, ’somersby apple nab’, ’carajillo cubano’, ’pisco sour de camila’, ’morgan bay cab fl’, ’boulard cidre biologique’, ’kronenbourg blanc’, ’morgan bay cab gl’, ’paso a paso verdejo fl’, ’brugal 1888/macallan double cask’, ’societa chianti gl’, ’mi abuela la loca’, ’extra carnitas 1st’, ’morgan bay chard gl’, ’gris de trop gl’, ’honoro vera gl’, ’cours de dame blanc fl’, ’cours de dame blanc gl’, ’villa conchi fl’, ’villa conchi gl’, ’gnarly head old vine zinfandel (usa) fl’, ’highland park/grönstedts vs/laphroaig’, ’baileys/sambuca/jim beam/famous grouse’, ’zacapa centenario 23yo’, ’the grinder shiraz gl’, ’gnarly head chardonnay (usa) gl’, ’falcon export’, ’bourbon bbq ribs’, ’kronenburg 1664 blanc’, "o’learys bbq sampler", ’brooklyn special effects’, ’cava mas pere brut’, ’jalapeno-lime mayo’, ’cotes de rhone villages fl’, ’brooklyn naranjito’, ’pina pina alex’, ’hunky dory sav fl’, ’carlsberg

non alco', 'super combo plate', 'gnarly head old vine zinfandel', 'boulard cidre de normandie', 'puycherich syrah rose', 'garage hard lemon', 'lagunitas ipa 35', 'sigtuna napa non alc', 'grimbergen blonde', 'kronenbourg blanc nab', 'diplomatico reserva', 'brooklyn summer ale', 'barolo ratti fl', 'gnarley chard', 'ocean spray cranberry', 'pepsi maxi stor', 'barceló imperial onyx', 'cidraie original organic', 'grönstedt vs cc', 'the macallan fine oak 12y', 'jim beam bourbon cc', 'gentleman jack daniels', 'galliano martini', 'ron zacapa centario 23', 'tapiz cabernet sauvignon fl', 'brugal 1888', 'grönstedts monopole vsop', 'maria bonita gl', 'morgan bay chard fl', 'el dorado 15yo', 'brooklyn bel air', 'long island iced tea', 'woodford reserve', 'societa chianti fl', 'the grinder shiraz fl', 'gris de trop fl', 'boulard grand solage calvados', 'collet chablis fl', 'highland park 12y', 'grönstedts xo', 'knob creek cc', 'lander jenkins pinot fl', 'angostura 1919', 'nicolas f brut resv', 'jim beam sour 6cl', 'anima negra fl', 'extra taco del mar', 'les cardounettes blanc eko (fra) gl']

Cluster 3

['erdinger hefe', 'guinness', 'staropramen', 'pepsi', 'ramlösa', 'bowling', 'kalaspaket', 'pepsi max', 'fries', 'aioli', 'fish n´chips', 'popcorn', 'combo plate', 'nachos', 'eriksberg', 'stam vitt', 'stam rött', 'red bull', 'zingo', 'guacamole', 'chili mayo', 'öppen mat', 'kids sundae', '7-up', 'bbq plate', 'öppen läsk', 'shuffle', 'shot', 'stam shot', 'majonäs', 'snacks', 'stam bowling', 'espresso dubbel', 'pripps blå i', 'bearnaise', 'embutido con trufa', 'maiz a la parrilla', 'bolas de flote', 'lagunitas ipa', 'tacos del mar', '7up', 'castellanovo gl', 'te', 'rosquilla frita', 'nonno´s favorito', 'el cubano', 'ceviche de pescado', 'lomo bajo (biff)', 'corona', 'kroketter', 'pimientos de padron', 'vegetariano', 'ron zacapa', 'brule de crema', 'dubbel espresso', 'plato de pescado', 'bryggkaffe', 'oumph nachos', 'the ortiz', 'cava gl', 'helado', 'san miguel', 'paso a paso gl', 'el mariachi &2', 'lomo alto (entrecote)', 'paso a paso fl', 'manzana fresca', 'margarita el diabolo', 'alex &1', 'alex &2', 'huber terassen gl', 'violeta', 'huber terrassen fl', 'caipirinha de pîna', 'onion rings', 'castellanovo fl', 'coleslaw', 'hot shot', 'läsk', 'hernö gin & tonic', 'öppen bowling', 'sub oumph caesar', 'halloumi fries', 'shots', 'andiamo white', 'personalöl', 'kampanjvin glas', 'öppen event', 'aperol spritz', 'corona', 'öppen mat', 'estrella galicia', 'facundo eximo', 'tea', 'andiamo red', 'onion ring', 'oumph quesadilla', 'stammis shot', 'pripps blå 1', 'anejo cuatro', 'somersby pear', 'öppen läsk', 'mojito', 'jäger-redbull dr', 'laphroaig 10y', 'jalapeno popper', 'sub salmon', 'potatispure', 'gin´n tonic', 'personal shot', 'pepsi stor gd', 'jack daniels', 'andiamo rose', 'jim beam sour', 'mezcalita', 'gräddfil', 'gin ´n tonic', 'ita prosecco gl', 'patron', 'frozen daiquiri', 'zingo stor gd', 'margarita', 'ea tinto fl', 'macchiato', 'side sallad', 'ea tinto gl', 'moscow mule', 'amargo dulce', 'dubbel macchiato', 'snacks', 'taittinger', 'öppen rom', 'chilibearnaise', 'presentkort', 'cortado', 'martell vs', 'martell vsop', 'extra tavo veg', 'öppen aktivitet']

Cluster 4

['kaffe', 'kaffe m. mjölk', 'ice cream sundae', 'chocolate brownie', 'kids brownie', 'mjölk', 'öppen kaffedrink', 'cappucino', 'tullamore irish coffee', 'cappuccino', 'espresso', 'milkshake', 'kaffedrink', 'espresso martini', 'coffee', 'tullamore coffe', 'espresso & ice cream', 'double espresso', 'kaffe & mjölk', 'jamaican coffee', 'caffe latte', 'chocolate milkshake', 'espresso

martini de laysa', 'café con chocolate', 'strawberry milkshake']

Cluster 5

['kids pancakes', 'grilled chicken quesadilla', 'hot crispy burger', 'cheddar cheeseburger', 'bacon bbq burger', 'buffalo wings', 'parmesan & garlic wings', 'garlic and cheese bread', 'chili cheeseburger', 'hot cheddar', 'giant double cheeseburger', 'classic steak oxfilé', 'herb butter grilled salmon', 'baconnaise burger', 'truffle oozing burger', 'beef and cheese nachos', 'kids cheeseburger', 'kids chicken nuggets', 'kids quesadilla', 'truffle mayo', 'parmesan & garlic', 'grilled chicken club sandwich', 'jalapeno oumph burger (vegan)', 'deep fried brownie', 'creamy beef tenderloin pasta', 'classic ceasar salad', 'blue cheese', 'sweet potato fries', 'classic steak ryggbiff', 'cheesy fries', 'add bacon', 'add garlic bread', 'parmesan and garlic ribs', 'camarones rojos fritos', 'plato de carne', 'tacos vegetariano', 'carbon frito', 'mozzarella sticks', 'krossad potatis', 'chili cheese burger', 'cheddar cheese burger', 'caesar salad', 'hot buffalo chicken wings', 'blue cheese dressing', 'lightly salted fries', 'beef & cheese nachos', 'parmesan & garlic sauce', 'bacon bbq burger', 'fish & chips', 'garlic bread', 'classic steak - striploin', 'pulled beef & bacon burger', 'grilled chicken quesadillas', 'classic steak - tenderloin', 'pkt parmesan garlic wings', 'giant double cheese burger', 'side tomato salad', 'sub fries to loaded fries', 'sub fries to sweet potato fries', 'salty apple crumble', 'parmesan garlic wings', 'add bacon - burger', 'side caesar salad', 'mozzarella stick', 'bbq grilled chicken', 'jalapeno oumph burger', 'pickled cucumber', 'sub fried chicken', 'crispy halloumi burger', 'bröd med dip', 'hamburgerdressing', "kids fish n'chips", 'kids vegetables', 'el cerdo maravilloso', 'parmesan & garlic ribs', 'dubble up your burger', 'extra cheese']