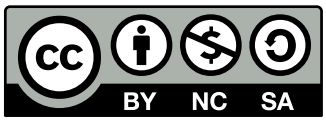
Aalto University
School of Science

Master's Programme in Mathematics and Operations Research

Clustering hierarchical purchasing categories for procurement benchmarking using sentence embeddings

Lukas Olenborg

Master's Thesis
2024



Aalto University
School of Science

Author Lukas Olenborg

Title Clustering hierarchical purchasing categories for procurement benchmarking using sentence embeddings

Degree programme Mathematics and Operations Research

Major Systems and Operations Research

Supervisor Dr. Jukka Kohonen

Advisor M. Sc. Sergey Zakrytnoy

Collaborative partner Sievo Oy

Date 31 July 2024 **Number of pages** 66+7 **Language** English

Abstract

The purpose of this thesis is to assess the feasibility of mapping similar textual purchasing categories for novel analytics and benchmarking in procurement. These categories exist in organisation hierarchies of large enterprise customers. Language models are used to represent text labels meaningfully, and clustering is used to group similar categories. Transformer-based sentence embedding (SBERT) models MPNet, and MiniLM proved highly effective in capturing textual similarities, resulting in high-quality mappings and outperforming traditional word embedding models. HDBSCAN was identified as a suitable clustering algorithm, detecting outlier points and effectively processing high-dimensional data with clusters of various shapes. The mappings demonstrated high accuracy and increasing coverage had a greater impact on the results and remains the key aspect to address in future work. This work proposes an automated solution to a crucial data mapping task, significantly reducing the required manual effort.

Keywords hierarchical text, language models, sentence embeddings, semantic textual similarity, clustering, supply chain, procurement, spend analysis

Preface

This thesis marks the end of my studies in mathematics and data science at Aalto University. I am happy to complete this challenging yet rewarding journey with a thesis I am proud of. I am grateful for the support and encouragement from the people around me, which made this thesis possible.

Firstly, I would like to thank my advisor, Sergey Zakrytnoy, for proposing this topic and steering the research towards business relevance. Thank you for the guidance and encouragement throughout this process.

I am also thankful to my supervisor, Jukka Kohonen, whose enthusiastic interest in the topic has been clear since our first meeting. Your mentorship and insightful feedback during the writing process have significantly enhanced this thesis.

I extend my sincere thanks to Sievo for providing this exciting opportunity and for the funding that made this research possible. Working on a project that is both relevant to the company and academically interesting has been truly motivational. To the team at Sievo – Anton, Ilia and Alireza – thank you for your active collaboration and support. Your insights and expertise have been a great help during this research.

Finally, I thank my friends and family for their encouragement and support from my school days through my university studies. A special mention goes to my roommate and great friend, Jan, for your patience and for always lifting my spirits during this process.

Otaniemi, 31 July 2024

Lukas Olenborg

Contents

Symbols and abbreviations

$A_{\texttt{1/0}}$	`1/0` Accuracy 33, 34
C	Coverage 33, 34
M	Number of correctly clustered categories 33, 34
N	Number of considered categories 33
$\bar{n}_C$	Mean cluster size 33, 34, 40
$\bar{t}_C$	Mean taxonomies per cluster 33, 34, 40
$\cos(\theta)$	Cosine similarity 27
$\mathbf{X}$	Weighted embedding vector 27
$\mathbf{x_{CL}}$	Category embedding vector 27
$\mathbf{x_{FP}}$	Flat path embedding vector, context vector 27
$\mathbf{x_{HP}}$	Hierarchical path embedding vector, context vector 27
$\tilde{n}_C$	Median cluster size 33, 34
$\tilde{t}_C$	Median taxonomies per cluster 33, 34
k	Number of clusters 33, 34, 40
w_C	Category weight 27, 49–52
w_P	Path weight, context weight 27
AvgComp	Average compactness 33, 34, 39, 40, 46, 68
AvgSep	Average separation 33, 34, 39, 40, 46, 49, 67
BERT	Bidirectional Encoder Representations from Transformers 11, 14, 15, 29, 30
CH	Calinski-Harabasz index 33, 34, 39, 40, 45, 46, 49, 50, 67
CL	Category label 11, 26, 27
DBI	Davies-Bouldin index 33, 34, 39, 40, 46, 48–51, 67
ELMo	Embeddings from Language Models 14
ERP	Enterprise resource planning 10
FPL	Flat path label 25–27, 35, 36, 38, 43, 44, 55
GDP	Gross domestic product 12
GPT	Generative pre-trained transformer 11
HPL	Hierarchical path label 25–27, 36, 43
ISE	Independent sentence embeddings 15, 27
LLM	Large language model 14, 30
LM	Language model 11, 17, 22, 23, 28, 29, 35, 36
MECE	Mutually exclusive and collectively exhaustive 10, 19, 22
MLM	Masked language modelling 14, 15, 30
MTEB	Massive Text Embedding Benchmark 14, 15
NLI	Natural language inference 14
NLP	Natural language processing 9, 13–15, 17, 58

PCA	Principal component analysis 31, 32
PLM	Permuted language modelling 15, 30
SBERT	Sentence-BERT 15, 59
SI	Silhouette score 33–35, 39, 40, 46, 48, 67
SOM	Self-Organizing Maps 17
STS	Semantic textual similarity 9, 13, 27–29, 59
TF-IDF	Term frequency-inverse document frequency 13, 17
WE	Word embeddings 13

1 Introduction

Recent advancements in the field of natural language processing (NLP) have enabled the processing of diverse and extensive textual data, leading to significant improvements in sophisticated tasks such as text generation, machine translation and text classification. Novel methods and language model architectures efficiently capture and represent meaning, context, and similarity of text. In this research, we explore semantic textual similarity (STS) of textual procurement data organised in complex hierarchies.

This work is conducted in collaboration with Sievo, a procurement analytics company that provides analytics for large enterprise customers with complex organisations and large amounts of spend. Spend data is organized into hierarchical purchasing categories with textual labels. These hierarchies (taxonomies) vary significantly across customers, being customized to meet their unique needs and requirements. This work adopts a high-level perspective, exploring the purchasing categories across customers.

The primary objective is to map similar categories across customers for high-level analytics and market benchmarking purposes. This research problem is particularly exciting for several reasons. The dataset is unique and challenging, with diverse textual categories in customer hierarchies from different industries. The proposed methodology utilizes state-of-the-art language models and is fully unsupervised, with clustering as a downstream task. This work presents an automated solution for a crucial data mapping task that would otherwise require extensive manual effort. It is a novel approach to a complex problem in the field of spend analysis.

Section 2 provides context to the field of procurement analytics and outlines the motivation and objectives of this thesis. A comprehensive literature review on text embedding models and text clustering is presented in Section 3. Section 4 describes the dataset used in our experiments, including data collection, data scoping, and preprocessing steps. The methodology in Section 5 details the processing of text labels and the use of embedding models, clustering algorithms and presents evaluation metrics. The design of experiments is outlined in Section 6. We review the results and present our findings and their business impact in Section 7. Finally, Section 8 summarizes this work and suggests future research.

2 Background

2.1 Sievo

Sievo is a leading Finnish procurement analytics company. One of Sievo's core products is spend analysis. A typical implementation of the spend analysis solution for a large multinational enterprise requires proper organisation and processing of procurement transactions extracted from scattered enterprise resource planning (ERP) systems and varying in quality, practices, and even language. In spend classification, this raw data is organised into hierarchical categories, also referred to as *taxonomies*. This classification is the backbone for various analytics and insights provided by Sievo, allowing for a high-level view of large enterprise spend and detailed analytics on specific products.

Sievo offers certain recommendations and best practices for taxonomies, but generally, the taxonomies are provided by the customer. This is important, as the taxonomy often represents the overall organisation and functions of a large enterprise customer. Taxonomies, therefore, vary greatly across industries and organisation sizes, and different conventions among customers.

A common split of spend data, appearing high-up in the taxonomy tree, is the split into *direct* and *indirect* spend. Direct spend refers to the money an organization invests in goods and services directly associated with the production of its primary products or services. This includes items like raw materials and components which are essential for the core manufacturing operations. Direct categories tend to be more carefully managed as they have a bigger impact on the overall profitability of the business; also, one may expect that direct categories are industry-specific. For example, a medical devices company would be procuring some electric components, while a food conglomerate would be buying sugar, beans and crops.

Indirect spend refers to the expenditures on goods and services that are not directly tied to the production process but are necessary for the overall functioning of the business. This category encompasses a broad range of items such as office supplies, utilities, marketing services, and IT support. Unlike direct categories, indirect categories are industry agnostic: most enterprises need to buy office supplies, financial services etc.

Categories in an efficiently designed product taxonomy should be mutually exclusive and collectively exhaustive (MECE), a principle introduced by Rasiel (1999). This means that the correct product category should be unique and unambiguous for each procurement transaction (mutually exclusive). At the same time, the taxonomy should provide a suitable category for any given transaction (collectively exhaustive). In reality, taxonomies are not optimal, despite being carefully planned and reviewed regularly. Small changes to taxonomies are common, such as the addition of new product categories, but large revamps also occur when, e.g., a customer's core business or data strategy evolves.

The customer taxonomies used in this thesis are anonymized to protect the privacy and confidentiality of Sievo customers. The Sievo Demo taxonomy, which closely resembles typical customer taxonomies is used to provide detailed examples throughout this thesis. Figure 1 displays a subtree of Demo taxonomy categories related to packaging.

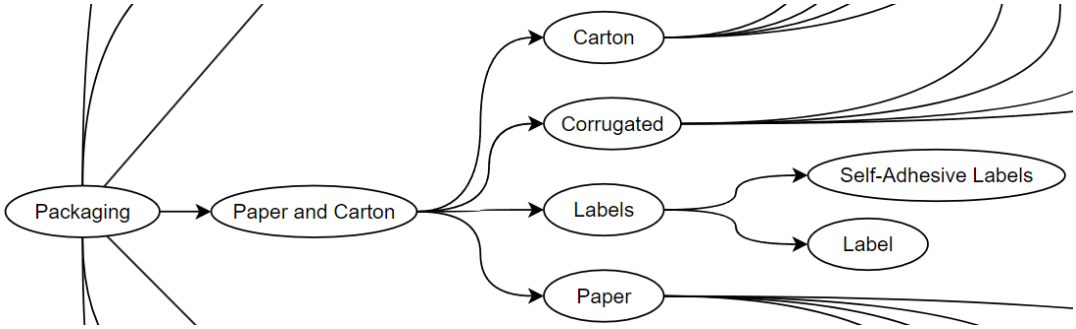


Figure 1: Sievo Demo taxonomy. Example subtree of packaging materials.

2.2 Problem statement

The purpose of this work is to assess the feasibility of mapping similar categories across customer taxonomies using language models (LMs) and clustering algorithms. This gives rise to several key objectives. Firstly, we need to identify and implement LMs that are well suited for processing hierarchical and textual procurement *category labels (CLs)*. State-of-the-art pre-trained LMs, such as BERT and GPT (Devlin et al., 2019; OpenAI, 2024) generate rich text representations, which capture context and meaning in addition to superficial aspects such as length and syntax. Text representations are numerical encodings of text, typically at token or word level. We need to implement computationally efficient models suitable for encoding varying length and hierarchical category labels. A key component in this work is the language model generating text representations, which naturally form clusters of similar product categories.

Secondly, we aim to cluster overlapping categories with sufficient coverage, using solely the category labels from taxonomies as source data. This requires suitable clustering algorithms that can handle high-dimensional text encodings and provide high-quality clusters. The datasets can be large and challenging as diverse customer taxonomies create imbalance and noise. Achieving meaningful clusters requires several data scoping and preprocessing decisions. Thirdly, we assess whether the context provided by the hierarchy, i.e., the hierarchical parent categories can be used to improve the accuracy of mappings. We construct *path labels* by recursively concatenating parent labels for each category across the hierarchy.

Finally, we will review and benchmark the methodology in different settings and aim to optimize it in terms of the quantity and quality of mappings. The lack of a labelled dataset is a key challenge in this work. Additionally, the methodology outputs clusters of categories instead of, e.g., structured results of data points in predetermined classes. Optimizing the methodology requires various experiments and

a rigorous evaluation strategy with suitable metrics. We leverage descriptive statistics and unsupervised clustering metrics in *grid searches* and combine this unsupervised approach with manual validation in an iterative approach to evaluate the results.

2.3 Business opportunity

Traditional spend analysis relies on structured internal and external data assets crawled from public sources or acquired from third parties. Sievo is well-positioned in the market, with its extensive internal datasets and advanced analytics platform. As Sievo continues to grow, it now processes around 2% of global GDP annually, creating opportunities for extensive community data benchmarks. Notable benchmarks in procurement include payment terms recommendations and proprietary commodity price indices.

A gap exists in spend analysis between high-level, publicly available global market averages and detailed per-customer analytics available on the Sievo platform. To achieve accurate industry and category-level community benchmarks, a system for mapping customer organizations – whether centralized or decentralized – is essential. This mapping task, referred to as *category harmonization*, is crucial and is already planned for implementation. Mapping customer hierarchies is essential for bridging the gap and enabling community benchmarks. However, given the scale of the taxonomies and complexity of many-to-many mappings, manual efforts for a comprehensive mapping are estimated to require hundreds of full-time employee days. This thesis proposes a methodology for automatically generating decentralized mappings, which can significantly support manual review and mapping of purchasing categories, thereby reducing the current labour-intensive process.

3 Literature review

3.1 Evolution of embedding models

Raw textual data is unsuitable for many NLP applications, such as sentiment analysis, machine translation, text generation, and clustering. Statistical and machine learning methods often require numerical data with fixed dimensions or categorical data, whereas textual data is unstructured, sparse and high-dimensional. This motivates the need for rich text representations. These representations should be numerical fixed-length vectors. Classical methods include one-hot encoding, bag of words, and term frequency-inverse document frequency (TF-IDF) (Manning et al., 2008). These methods define a text corpus containing all n unique words appearing in a dataset. In one-hot encoding, each sentence or document is encoded into a binary vector of length n, indicating the presence of words with 1s. Bag of words builds on this approach, capturing the number of occurrences for each word and adding frequency to the representation. A drawback of this method is that common words may not be descriptive, and they are over-represented. TF-IDF considers both the local frequency of a word and the global infrequency, giving more weight to rare words. Novel methods based on neural networks have since been popularized, discussed below.

The emergence of word embeddings (WEs) was a significant advancement in the NLP field, popularised by efficient embedding models based on shallow neural networks. A missing piece in classical methods is the notion of semantic similarity between words. Mikolov et al. (2013) introduced Word2Vec, a neural network model architecture that efficiently generates high-quality continuous word embeddings. The efficiency of the model allowed for increased training over large datasets, resulting in improved quality in various STS tasks. Word2Vec proposed two architectures: a continuous bag of words model predicting a word based on the context of the surrounding words and a continuous skip-gram model trained on a classification task. Pennington et al. (2014) introduced Glove, inspired by global matrix factorization from earlier methods and the use of local context from Word2Vec. Glove also outperformed earlier methods and considered global characteristics of a word corpus in addition to the local context in Word2Vec. In addition to improving semantic richness in embeddings, these models greatly reduced dimensionality. Bojanowski et al. (2017) improved on Word2Vec by considering sub-word information with the fastText architecture, allowing the generation of embeddings to words not seen during training. Arora et al. (2016) describe a robust and highly performant methodology for various NLP tasks, using these WE models.

Transformer-based models brought another leap in advancement, enhancing context-awareness in language models with the attention mechanism, initially proposed by Vaswani et al. (2017) in *Attention Is All You Need*. The architecture moved from using recurrent and convolutional neural networks to only using the attention mechanism. The attention mechanism uses the most relevant parts from the entire input sequence, whereas earlier models fixed context windows. The transformer also brought improvements with a deeper neural network, parallelism and improved training schemes. A simple diagram of the transformer architecture can be seen in

Figure 2. The transformer initially achieved state-of-the-art performance in language translation while also laying the groundwork for today's advanced generative models, such as GPT-4 developed by OpenAI (2024). Before transformers, Peters et al. (2018) introduced Embeddings from Language Models (ELMo), a deep bidirectional language model, which improved context awareness and achieved state-of-the-art performance in several tasks. Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), perhaps the most widely used model, combined the architectures of bidirectional models and transformers. BERT is trained with masked language modelling (MLM), where the model is tasked to predict some masked words in a sentence. BERT's architecture also allowed finetuning a single output layer to various tasks in NLP. Liu et al. (2019) trained BERT on larger datasets introducing RoBERTa, and DeBERTaV3 (He et al., 2022), a state-of-the-art model in various natural language inference (NLI) tasks. ALBERT, ELECTRA, and MiniLM brought improvements in memory, size and efficiency (Lan et al., 2019; Clark et al., 2019; Wang et al., 2020).

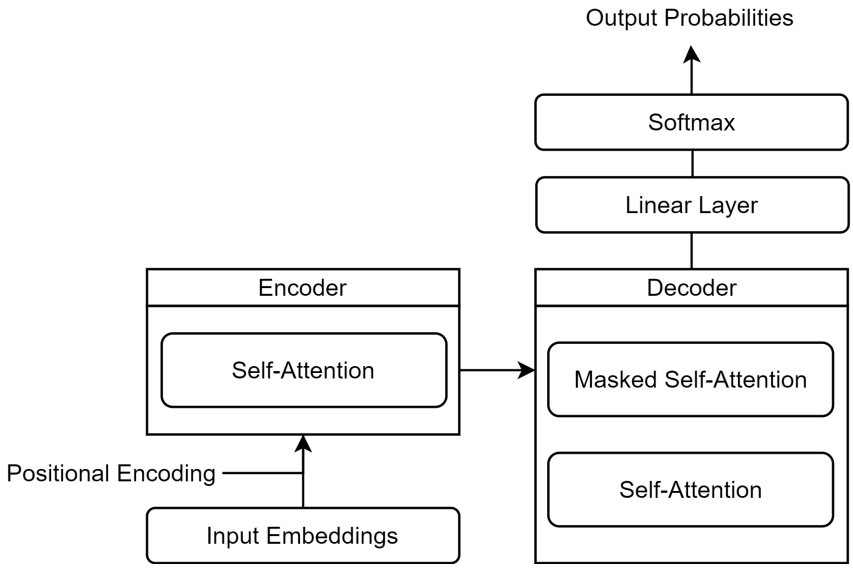


Figure 2: The transformer architecture. For more details, see Vaswani et al. (2017).

As the demand and use of language models increased significantly, the need for reliable evaluation and comparison of these models emerged. SentEval (Conneau and Kiela, 2018) and GLUE (Wang et al., 2018) aim to universally evaluate the quality of sentence embeddings on various NLP tasks. Perone et al. (2018) carried out a comprehensive evaluation of popular sentence embeddings on downstream tasks. In their findings, new models such as ELMo showed promise, while no model performed best across all tasks. They also demonstrated that models typically perform best on tasks resembling those used in pre-training, suggesting that models have not generalized that well. Muennighoff et al. (2023) introduced the Massive Text Embedding Benchmark (MTEB), a popular benchmark over various datasets and NLP tasks. Freestone and Santu (2024) review whether embeddings of recent large language models (LLMs), such as GPTs, provide improvements in embeddings, finding similar or slightly improved performance compared to BERT.

3.2 Sentence embedding models

The evolution of embedding models is crucial for advancing natural language processing. Models like BERT revolutionized the field by leveraging deep neural networks and extensive pretraining, improving context awareness and capturing meaning and similarity. This resulted in significant improvements in various NLP tasks compared to previous models. Its success enabled the development of sentence embedding models, which specialize in representing entire sentences.

Reimers and Gurevych (2019) modified the BERT architecture with siamese and triplet network structures to efficiently generate universal and independent sentence embeddings (ISEs) with Sentence-BERT (SBERT). Embeddings in BERT are token-level and dependent on the specific context, inputs and task used in training. SBERT applies a pooling strategy on token-level outputs of BERT to directly generate embeddings for whole sentences. The SBERT architecture has since expanded and supports various models designed for sentence-level tasks. Considering our specific task and recent advancement in the field, MiniLM, MPNet, and RoBERTa are promising models (Wang et al., 2020; Song et al., 2020; Liu et al., 2019). These models are compatible with the SBERT architecture and rank highly in general embedding benchmarks and clustering tasks in MTEB. Additionally, they are of moderate size in terms of memory and high speed while outperforming most older models and other models of similar size. MiniLM is based on deep self-attention distillation, i.e., compressing large transformer models by training a smaller student model to mimic the larger model. MPNet combines techniques from MLM in BERT and permuted language modelling (PLM) in XLM while addressing their limitations, outperforming both in various tasks.

While embedding dimensions were initially reduced with Word2Vec and GloVe, the more recent advanced models have increased dimensions, aiming to capture more richness in context and semantics. BERT is available as a base and large model, where embeddings are output in 786 and 1 024 dimensions, respectively. High dimensions can lead to the "curse of dimensionality", where the feature space grows disproportionately in comparison to the number of data points, resulting in sparsity and significant challenges in many applications. Wang et al. (2023) showed that the output dimensions of today's embedding models are often unnecessarily high and could be reduced. However, SBERT models typically provide embeddings in lower, e.g., 384 or 512 dimensions.

3.3 Clustering text representations

Clustering is a method to group similar data points. Unlike classification, clustering detects naturally occurring patterns and groups in an unsupervised setting, with no prior notion of classes. Clustering is commonly used in exploratory data analysis when visualizing data, but it is also an important downstream method in many applications, such as data mining (Tan et al., 2018). Clustering methods are commonly divided into *hierarchical* and *partitioning* methods. Partitional methods divide the data into distinct groups, whereas hierarchical clustering includes subclusters.

K-means is a popular and efficient partitional clustering algorithm which splits a dataset into k groups. It works by first selecting k initial centroids and assigning data points to the closest centroid. These centroids are then repeatedly updated using the mean location of data points in the cluster. However, this approach has several limitations: the number of clusters is fixed and must be specified by the user, the clusters are assumed to be spherical and of similar density, and the algorithm is non-deterministic and sensitive to initial centroid locations (MacQueen, 1967).

Agglomerative clustering starts with all points in separate clusters and then merges clusters based on a distance measure. This process is repeated until all points are in a single cluster, resulting in a hierarchy of clusters. Divisive clustering starts with all points in a single cluster and iteratively splits the clusters. Both methods result in hierarchies of clusters which relaxes some limitations in non-hierarchical clustering algorithms, such as k-means. There is no need to specify the number of clusters in advance, and the granularity of clusters can be chosen using a suitable cut-off. It can also be very useful when the underlying data is hierarchical (Tan et al., 2018).

Density-based clustering relaxes the strict geometric, centroid-based clusters of k-means by defining clusters simply as regions of high density. This flexible approach allows finding clusters of varying shapes and sizes, even in higher dimensions. A density-based algorithm visits all points in the data and begins by defining points in high-density regions as core points. Surrounding points are either set as additional *core points* or *neighbouring* points. Points not reached from any cluster are finally set as outliers, a characteristic unique to density-based algorithms. HDBSCAN (Campello et al., 2013) combines the approach from hierarchical and density-based clustering, resulting in a versatile and efficient algorithm suitable for high-dimensional and hierarchical data and irregular clusters. An example of k-means clustering and HDBSCAN, a density-based, agglomerative, and hierarchical clustering algorithm, can be seen in Figure 3.

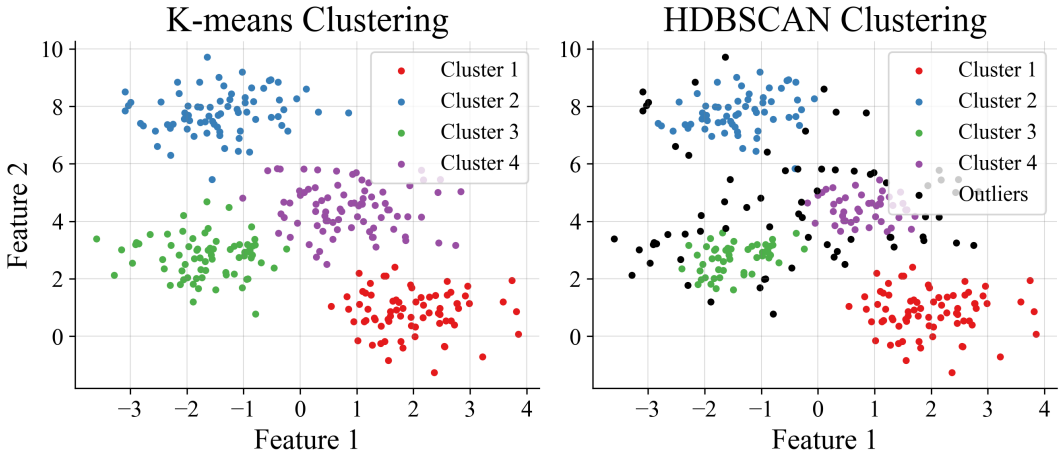


Figure 3: Illustration of k-means and HDBSCAN clustering. Synthetic data points sampled around 4 centroids are clustered with k-means and HDBSCAN algorithms. Note the outlier points in HDBSCAN.

Clustering single words has been extensively explored using traditional methods like TF-IDF and k-means (Manning et al., 2008). However, clustering hierarchical *sentences* using embeddings from modern LMs is a less explored area, particularly beyond visualisation purposes. Clustering appears in NLP as a downstream task in, e.g., text summarization (Haider et al., 2020), topic modelling (Weißer et al., 2020) and information retrieval (Reimers et al., 2019; Zoupanos et al., syyskuu 9, 2022). Haider et al. (2020) use word2vec and k-means in a sentence-based document summarization task. Saha (2023) shows that (H)DBSCAN outperforms k-means and single linkage clustering in clustering e-commerce customer reviews.

Tache et al. (2021) found Self-Organizing Maps (SOM), proposed by Kohonen (2001), to improve clustering language compared to k-means, resulting in a distribution closer to Zipf's law. Zipf's law states that the frequency of an item is inversely proportional to its rank in the frequency table, and it describes the distribution of words in natural language (Powers, 1998). Argyrou (2009) utilizes SOMs in clustering hierarchical data in a graph-based approach. Uma and Santhi Thilagam (2023) cluster hierarchical and structured JSON data and discuss the tradeoffs between using content and structure. de Knijff et al. (2013) and Li et al. (2013) discuss methods to generate domain or topic taxonomies from text documents. Aggarwal et al. (2001) discuss the curse of dimensionality in the context of clustering and distance measures.

Sheela et al. (2023) use fuzzy clustering for text summarization. In fuzzy clustering, points can belong to several clusters with a continuous membership function taking values 0 to 1, characterizing the degree of belonging of each data point to each of the clusters. This can be beneficial in an open setting, where the textual data and potential clusters are ambiguous, with overlapping themes.

As clustering is unsupervised and often used for visualization purposes, cluster evaluation is not always a requirement (Tan et al., 2018). However, it is crucial in tasks where clusters represent the final output. Evaluation metrics for clustering can be divided into internal and external evaluation metrics. Internal, i.e., unsupervised metrics review the general structure and shape of the resulting clusters. These metrics often consider how tight the clusters are (cohesion) and how far away clusters are from each other (separation). On the other hand, external evaluation metrics are used when a labelled dataset exists, indicating the desired clusters. Many benchmarks in the NLP field use V-measure as an external evaluation metric. V-measure assesses the balance between each output cluster containing only members from a single labelled cluster (homogeneity) and all members of a given labelled cluster belonging to the same output cluster (completeness, Rosenberg and Hirschberg 2007).

This thesis will not include a comparison of various clustering algorithms, as we focus on data strategy and embedding models. We have opted to utilize HDBSCAN (Campello et al., 2013) as our primary clustering algorithm. HDBSCAN can identify clusters of various shapes and sizes, providing flexibility and handling high-dimensionality well. Density-based algorithms' ability to detect noise and outlier points is well-suited for our task, as we expect some imbalance in our datasets. Additionally, this capability simplifies our workflow as we would otherwise require a separate step to remove bad clusters. Finally, HDBSCAN is hierarchical and computationally efficient, which is beneficial with high dimensional datapoints.

4 Data

To assess the feasibility of mapping similar categories between customer taxonomies, we have used anonymized taxonomies of industry partners – customers of Sievo. As discussed in Section 2.1, Sievo organises data by classifying procurement transactions of customers into categories. Moreover, the categories are organised in purchasing taxonomies. This section briefly reviews the Sievo Demo taxonomy in Section 4.1. We then present and explore datasets used in this work and discuss data scoping in Section 4.2. Finally, in Section 4.3, we review the required preprocessing and introduce two label generation strategies.

Some taxonomies reflect a heavier focus on indirect purchasing categories, while others include a comprehensive hierarchy of direct spend. We curated several anonymized and generalized taxonomy datasets (see Table 1): industry-specific datasets, a dataset with mostly indirect spend and one large, global dataset including the majority of Sievo customers. Indirect spend tends to have very similar categories, as this spend is standardized and industry-agnostic. In some taxonomies, most matches may seem trivial, but the methodology described in this work remains useful, as it automates the process and highlights differences in such taxonomies. Direct spend is varied, but we can expect similarities in industry-specific datasets. However, the organisation and size of such taxonomies are typically diverse. The global dataset provides an excellent foundation for identifying numerous matches and benchmarking the methodology on a large scale. On the other hand, it may be challenging due to excessive repetition and noise.

Table 1: Taxonomy datasets selected for exploratory data analysis. **Categories** refers to the total number of categories before any processing or scoping.

Industry	Description	Taxonomies	Categories
Telecom	Mostly indirect categories	5	3 587
Machinery	Mixture of indirect and direct categories	8	7 089
Pharmaceuticals	Detailed direct categories representing chemical components	7	3 904
Global	Large, diverse datasets with specifics of multiple industries	106	140 614

The numbers of taxonomies and total categories vary across the datasets. As seen in Table 1, the global dataset includes 106 taxonomies and over 140 thousand categories. The other datasets are much smaller subsets of this dataset, with 5–8 taxonomies and 3 500–7 100 total categories.

4.1 Exploratory data analysis

We begin the data exploration by reviewing the Sievo Demo taxonomy. The Demo taxonomy is a 4-level taxonomy, and a subtree can be seen in Figure 4 (see also Figure 1). The taxonomy has three categories at the highest level: "Indirect", "Raw Materials", and "Packaging", which expand into 29 level 2 categories and 137 level 3 categories. The lowest level has 481 categories, accumulating to 650 categories in the taxonomy. "Packaging" is the smallest subtree with 78 categories, while the "Indirect" and "Raw Materials" subtrees cover over 200 categories. On average, parent categories have 3.83 *direct* child categories at the subsequent level. The category "Additives" has the most children, with 35 direct child categories, appearing at level 3 in the taxonomy path `Raw Materials -> Seasoning and Additives -> Additives`. Table 2 presents statistics for the Demo taxonomy.

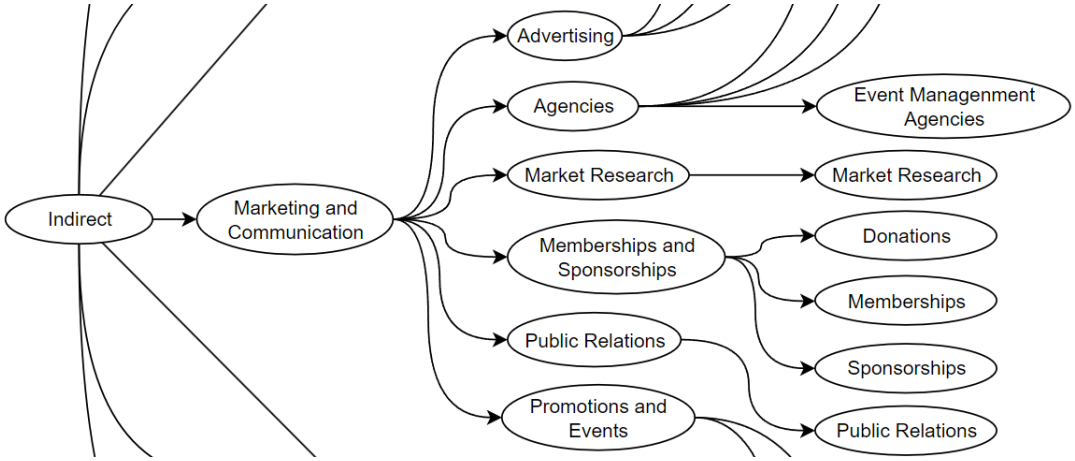


Figure 4: Sievo Demo taxonomy. Example subtree with categories related to marketing.

The Demo taxonomy is designed to resemble typical customer taxonomies in size, structure and relevance of categories. However, it strictly adheres to the MECE principle and contains little noise and clean labels, which is not always true in customer taxonomies. Nonetheless, the Demo taxonomy exhibits repetition in the form of duplicate categories, which is also typical in customer taxonomies. There are no duplicates in the first two levels, but within level 3, the "Labels" category is duplicated, appearing in `Packaging -> Plastic -> Labels` and `Packaging -> Paper and Carton -> Labels`. Level 4 includes five duplicated categories. There are a total of 64 duplicated categories, where most of the duplication appears across different hierarchy levels, such as `Public Relations -> Public Relations` in Figure 4.

We now explore the datasets in Table 1. The distribution of taxonomy depths, i.e., the number of levels in the hierarchies for the global and machinery datasets, can be seen in Figure 5. Almost half of the taxonomies in the global dataset have a 4-level hierarchy (47 out of 106). Over 90% of taxonomies have 3–5-level hierarchies. These numbers are in line with the distribution of all 150+ taxonomies in Sievo. The minimum number of levels is 2, with a single taxonomy, and the maximum number

of levels is 7, with four taxonomies. The machinery dataset includes two taxonomies with 3-level hierarchies, two with 4-level- and four with 5-level hierarchies.

Table 2: Statistics on Sievo Demo taxonomy.

Statistic	Value
Num categories, Level 1	3
Num categories, Level 2	2
Num categories, Level 3	137
Num categories, Level 4	481
Num total categories	650

Statistic	Value
Avg num children	3.83
Max num children	35
Min num children	1
Num within level duplicates	6
Num total duplicates	64

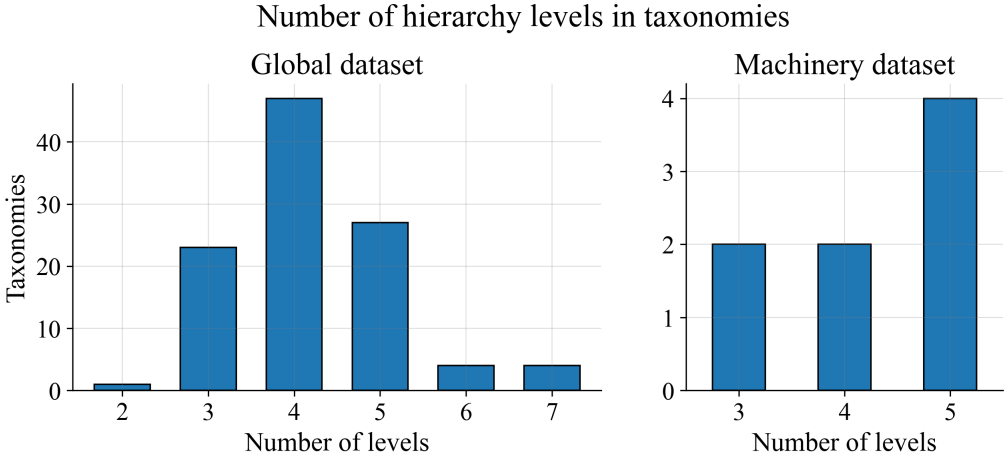


Figure 5: Distribution of taxonomy depths. The global dataset is on the left, and the machinery dataset is on the right.

Figure 6 displays the number of categories at the highest and lowest levels of the hierarchies for the global dataset taxonomies. The minimum number of categories at the highest level is 1, and the maximum is over 20. Around half of the taxonomies, 55 out of 106, have 1–5 categories at the highest level. Many of these are examples of the indirect/direct split, and some include packaging as a separate category. 83% of taxonomies have no more than 10 categories at the highest level. The second bar plot indicates that most Sievo taxonomies have less than 500 categories at the lowest level, with 62 taxonomies. 83% of taxonomies have no more than 1 000 categories at the lowest level. The 18 largest taxonomies have over 1 000 categories, including an outlier with over 10 000 categories at the lowest level.

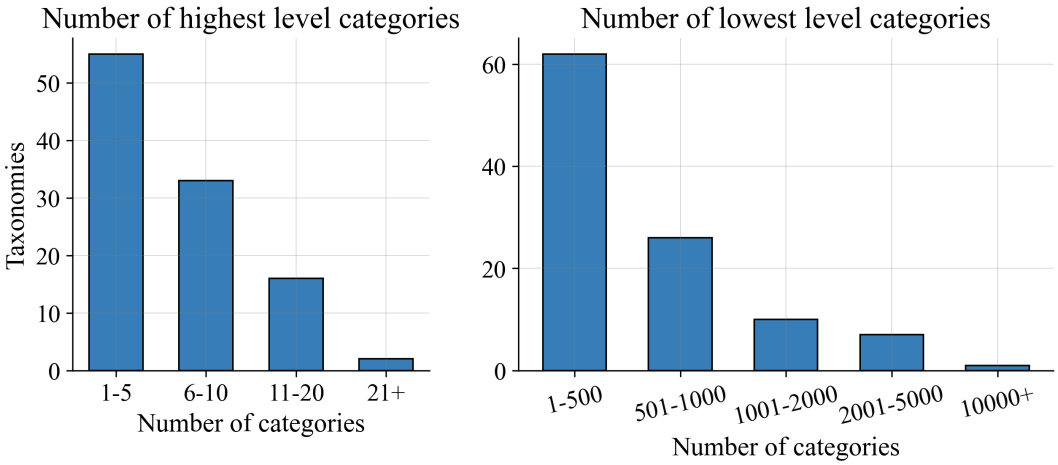


Figure 6: Number of categories in the global dataset. Categories in the highest (left) and lowest (right) levels of the hierarchy.

Table 3: Example data point from Demo taxonomy.

Feature	Value
Category ID	1
Level 1 Parent	Raw Materials
Level 2 Parent	Nuts and Seeds
Level 3 Parent	Nut
Category	Hazelnut

An example data point can be seen in Table 3. Each data point consists of a unique identifier `Category ID`, the category label (`Category`) and the labels of all succeeding parent categories until the highest level (`Level N Parent`). The label of this category, "Hazelnut", and the labels of its parents are short and concise 1 to 3-word descriptions. The longest label, "Nuts and Seeds", has 14 characters, while the direct parent, "Nut", has only three characters.

Figure 7 shows the word and character count distributions across 140 614 categories in the global dataset. The category labels are often a single word. 1–3-word category labels cover 72% of all categories, and 1–5-word category labels cover 89% of all categories. Less than 4% of categories have labels with over 10 words, but this still corresponds to over 5 400 categories. The character count distribution shows that most category labels have 11–20 total characters, which aligns well with the common number of 1–3 words per label. 82% of categories have 2–30 characters, and category labels with 2–40 characters cover 90% of categories. This bar plot also shows 1 037 very short category labels with no more than two characters. However, this corresponds to less than 0.8% of the dataset. Some of these are problematic, such as single letters, which may correspond to some company conventions or a single dash symbol (-), but most are short abbreviations, such as "HR" or "IT".

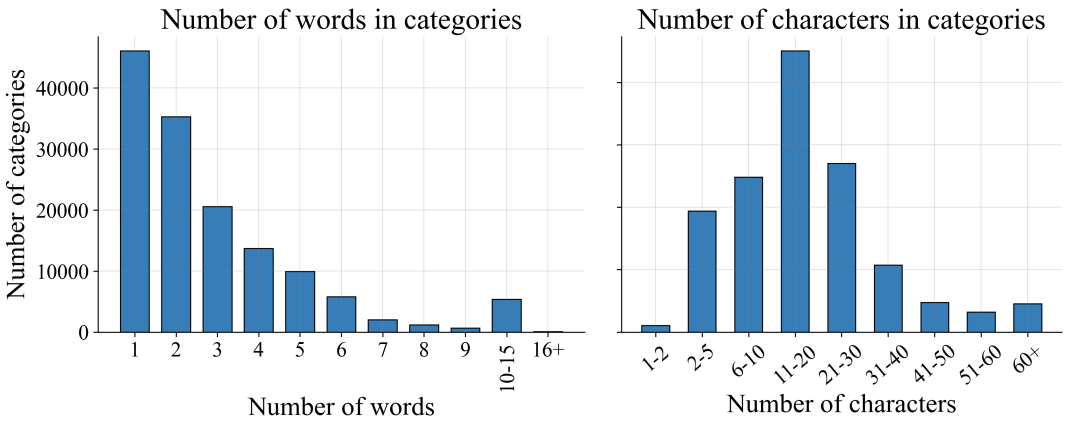


Figure 7: Word and character count distributions of categories. Includes all categories from the global dataset before data scoping.

4.2 Dataset scoping

We investigated the smaller datasets more thoroughly by visualizing taxonomy structures, sampling categories, and parent categories. We observed that high-level categories are short, generic, and often quite broad regarding business relevance. Diving deeper into the hierarchy, the categories become increasingly specific at lower levels, often reflecting highly industry-specific details and sometimes very granular distinctions, such as the smallest components and parts. The aforementioned observations and differences in level of detail are most apparent in large taxonomies with deep hierarchies. Very specific low-level categories can be difficult to map to common categories in a smaller taxonomy. This analysis indicates that striking the right balance in the level of detail for business relevance and minimising noise and imbalance is challenging and crucial in obtaining high-quality mappings. This issue will be discussed later in the section.

We identified some noise in the labels and encountered a lot of repetition. The noise includes special symbols such as `&()+,-./` and numbers. Some taxonomies capitalize everything, and we observed other company-specific conventions, such as codes. While it can be argued that some modern LMs represent common symbols meaningfully, they likely do not bring substantial semantic value to the labels.

Repetition was present in many directions: repeating words within a category label, repeating words across parent categories, repeating identical categories across parent categories and, finally, repeating identical or very similar categories at the same hierarchy level. Repeating words within a category are not that common, but it is common for a word such as "Packaging" or "Services" to repeat multiple times across parent category labels. Repeating identical categories across parent categories was quite common. This is especially prevalent in taxonomies where paths in the entire taxonomy or subtrees of the taxonomy have been enforced to a uniform depth. This is a relatively common practice, which goes against the MECE principle if considering the taxonomy as a whole. However, the lowest level categories, often used in, e.g., classification, remain MECE. Finally, repeating categories at the same hierarchy level

often occurs with slight product variations such as colour or other attributes. Another cause is vague words or *homonyms*, i.e., words with multiple meanings in different contexts, where a generic example is "Part".

The issue of repetition, again, compounds in large taxonomies in terms of deep hierarchies with many categories at lower levels. Examples of repetition in Sievo Demo taxonomy are shown in Table 4. The Demo taxonomy had no repeating categories at the same level or repeating words within single category labels, although these were found in customer taxonomies.

Table 4: Examples of repeating words and identical categories across parent categories in Sievo Demo taxonomy. The last is an example of a potential homonym.

Parent Categories	Category
Indirect > Professional Services > Financial Services >	Investment Services
Indirect > IT and Telecom > IT Consulting >	IT Consulting
Packaging > Packaging Adhesives > Tape >	Tape
Raw Materials > Gas, Nitrogen >	Nitrogen
Raw Materials > Nuts and Seeds >	Nut

Considering the issue of repetition, changes in the level of detail of categories, and high variability of taxonomy size, we decided to scope the datasets to a single level. We investigated the hierarchy levels of taxonomies in the smaller datasets and picked a single best level for each. We made this choice so that the categories are detailed enough while maintaining numerous categories, and the categories are business-relevant. To remain consistent, we implemented the following simple logic in automatically choosing this single level:

Depth of Hierarchy ≤ 2 $\rightarrow$ Select Level 2
Depth of Hierarchy $3-5$ $\rightarrow$ Select Level 3
Depth of Hierarchy 6+ $\rightarrow$ Select Level 4

This selection process ensures that considered categories have a similar level of detail. This is more interesting from a business perspective and advantageous in inputting balanced categories into the LMs. It also greatly reduces the issues created by excessive repetition and moderates the differences between large, complex taxonomies and smaller taxonomies. We select the lower level (level 2) for small taxonomies with two levels to maximize detail and the number of categories. We also choose the lowest level for 3-level taxonomies. For taxonomies with 4–5 levels, we maintain the selection at level 3. For large taxonomies with 6+ levels, we select level 4 as we observed that the level of detail becomes irrelevant at levels any lower than this. As most taxonomies have 3–5 levels (90%, see Figure 5), we will most often select level 3 categories. The levels handpicked in our initial investigation of customers of the smaller datasets also fit the selection process above.

The scoped datasets introduced earlier are listed in Table 5. With the described selection process, only 15–30% of categories are kept in the smaller datasets. In

8 machinery taxonomies, 1 314 categories are selected from over 7 000 original categories. Manually validating most clusters in smaller datasets is also more feasible, given that the number of points is in the hundreds instead of thousands. The global dataset size decreased from 140 thousand to 23 thousand categories during scoping.

This selection process partially discards the hierarchical structure and flattens the dataset to categories of a single level. However, we can still use the parent categories of each category or even sample lower-level categories as part of the data point. In addition to the issues discussed earlier, this scoping seeks to prevent single taxonomy clusters. The clustering algorithm knows nothing about the customers, and imbalanced, noisy datasets would lead to large taxonomies dominating the clusters. While this may solve another problem, namely refining and simplifying a taxonomy, it is not as interesting in this work, which aims to map categories across customers.

Table 5: Scoped taxonomy datasets.

Industry	Taxonomies	Total Categories	Selected Categories
Telecom	5	3 587	698
Machinery	8	7 089	1 314
Pharmaceuticals	7	3 904	1 146
Global	106	140 614	23 552

4.3 Preprocessing and label generation

As mentioned in the above sections, the categories in customer taxonomies contain some noise. This section presents the preprocessing steps taken to reduce this noise and discusses category path label generation.

Numbers, special symbols and common stopwords such as "and", "or" and "the" are removed. We also remove very generic categories, such as "Other". Whitespaces are converted into a single space, and all letters are lowercased. Repeating words are removed so that the first occurrence of the word is kept and the order is otherwise preserved. Lemmatization of words, a common text preprocessing method where words are transformed into their root form by removing plural forms or inflexions, is not used, as advanced language models can represent these relationships and similarities.

The updated word and character count distributions after data scoping and preprocessing are shown in Figure 8 (see also Figure 7). Two-word categories are now more common than single-word labels. The relative number of 1–3 word categories has also increased, from 72% to 81%. There is a visible shift to the left in the distribution, resulting in lower frequencies in high word counts of 5 and above. Less than 4% of categories have labels with over 6 words. Regarding character counts, the distribution has become more normal, symmetric and narrower, with a higher peak in 11–20 characters and smaller frequencies at tail-ends. 82% of categories have 6–30 characters. Overall, these changes indicate successful data scoping and preprocessing, resulting in labels that are detailed, concise and of suitable length.

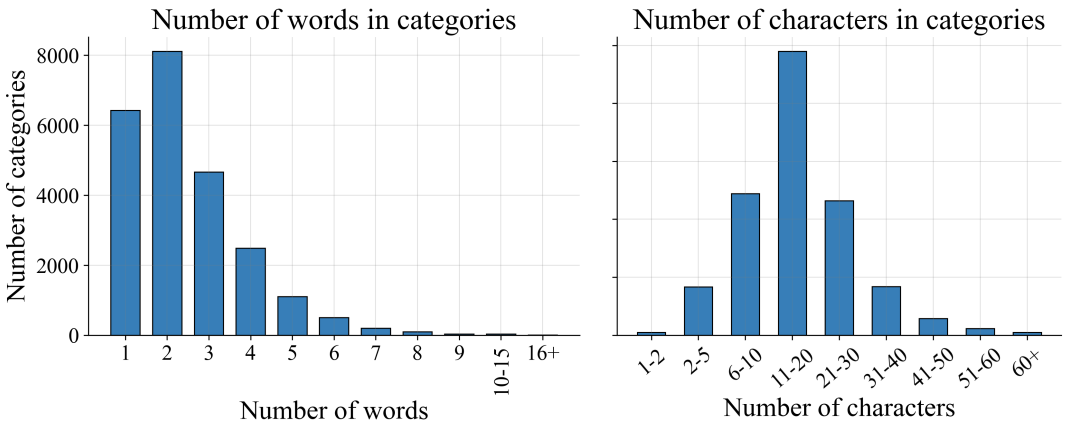


Figure 8: Word and character count distributions of categories. The global dataset after scoping and preprocessing.

The path label is constructed after preprocessing of categories such that the parent nodes for each category are recursively concatenated with a separator character. We considered two versions with different levels of preprocessing. The hierarchical path label (HPL) represents the hierarchical structure with the ">" separator symbol and includes no additional preprocessing. The flat path label (FPL) discards the hierarchical structure; repeating words are removed across the whole path, and a single space is used as a separator symbol. If a repeating word appears in the category itself, it is preserved in the category and removed from the parents, otherwise the first appearance of the repeating word is kept. Examples of HPLs and FPLs can be seen in Table 6. Both path labels include the category, as sometimes parent categories are short or ambiguous. This allows the category to be seen in its context instead of separating the category and context. The cleaned category label and path labels of the example datapoint in Table 3 are shown in Table 6.

The level of preprocessing impacts the tradeoff between removing noise and losing information. We decided to take a liberal approach in cleansing the data, as the longer path labels contain noise and differences in taxonomy size and structure create a substantial imbalance in the data, partially addressed in data scoping (see Section 4.2). Heavy processing removes some of this noise and decreases the role of superficial attributes such as length or company-specific conventions. For example, removing repeating words is very useful in long paths, where a word such as "Services" may appear multiple times, increasing the label length unnecessarily and gaining excessive weight. Not removing this repetition leads to large clusters of different types of services. Lowercasing all words may cause information loss, as certain brand names or abbreviations are more commonly capitalized. However, not all taxonomies follow the same conventions in capitalization, so lowercasing is useful in normalizing the data.

Removing generic categories is done sparingly, as categories like "Component" or "Part" may gain substantial meaning as provided by the path label. However, avoiding large clusters of "Part"s may be challenging and requires significant weighting on the

provided context label. This is a repeating tradeoff in this work, as sometimes the context provided by the path is crucial. Sometimes the category is descriptive enough, and the path is unnecessary noise. Despite the preprocessing and data scoping, path labels from deep taxonomies are longer on average, which makes it difficult to map to similar categories with much simpler path labels.

This section concludes with the presented datasets being reviewed, scoped and preprocessed, preparing for the core methodology. The cleaned datasets comprise the category and the hierarchical and flat path labels for all considered categories.

Table 6: Example data points from Sievo Demo taxonomy and the preprocessed category and path labels.

Feature	**Value**
Category ID	1
Level 1 Parent	Raw Materials
Level 2 Parent	Nuts and Seeds
Level 3 Parent	Nut
Category	Hazelnut
Preprocessed CL	hazelnut
HPL	raw materials > nuts seeds > nut > hazelnut
FPL	raw materials nuts seeds nut hazelnut
Category ID	2
Level 1 Parent	Indirect
Level 2 Parent	Logistics
Level 3 Parent	Logistics Services
Category	Packaging Services
Preprocessed CL	packaging services
HPL	indirect > logistics > logistics services > packaging services
FPL	indirect logistics packaging services

5 Methodology

The previous section detailed the data collection, scoping and preprocessing steps, including path label generation. This section focuses on the core methods in this work, namely the sentence embedding models (ISE) and HDBSCAN clustering. Additionally, we discuss dimensionality reduction and introduce metrics, which ensure reliable review and evaluation of this methodology. The methodology, including the data scoping and preprocessing, is shown in the diagram in Figure 9.

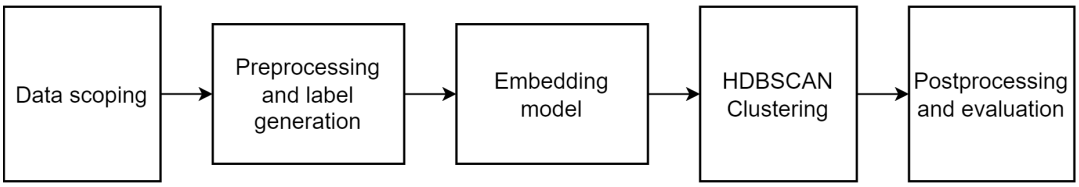


Figure 9: Data collection and methodology.

5.1 Embedding models

The cleansed textual category and path labels are embedded into numerical vectors using language models:

$$\mathbf{x_{CL}} = \mathrm{LM}(\mathrm{CL}) \in \mathbb{R}^d \tag{1}$$

$$\mathbf{x_{FP}} = \mathrm{LM}(\mathrm{FPL}) \in \mathbb{R}^d \tag{2}$$

$$\mathbf{x_{HP}} = \mathrm{LM}(\mathrm{HPL}) \in \mathbb{R}^d \,, \tag{3}$$

where $\mathbf{x_{CL}}$ is the category embedding vector, $\mathbf{x_{FP}}$ and $\mathbf{x_{HP}}$ are the flat and hierarchical path embedding vectors, respectively, LM is the language model, and d is the output dimensionality of the language model. These embedding vectors are then normalized, and finally, a weighted average is taken over the category and path embedding vectors:

$$\mathbf{X} = w_C \left(\frac{\mathbf{x_{CL}}}{\|\mathbf{x_{CL}}\|} \right) + w_P \left(\frac{\mathbf{x_{\{F,H\}P}}}{\|\mathbf{x_{\{F,H\}P}}\|} \right) \in \mathbb{R}^d \,, \tag{4}$$

where $w_C, w_P \in [0, 1]$ are the scalar weights for the category and path, respectively, and $w_C + w_P = 1$. These weights can be adjusted to balance the emphasis between individual categories and the context provided by path embeddings. This process results in the final weighted embedding vector $\mathbf{X}$, used as input for the clustering algorithm.

While cosine similarity ($\cos(\theta)$), which measures the angle between vectors, is commonly used in STS tasks (Mikolov et al., 2013; Reimers and Gurevych, 2019), we use Euclidean distance as the internal distance metric of HDBSCAN in this work.

The cosine similarity and Euclidean distance between two vectors $\mathbf{u}$ and $\mathbf{v}$ are defined as:

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \tag{5}$$

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} \tag{6}$$

For normalized vectors, where $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$, the cosine similarity simplifies to $\cos(\theta) = \mathbf{u} \cdot \mathbf{v}$ and the Euclidean distance simplifies to $d(\mathbf{u}, \mathbf{v}) = \sqrt{2 - 2(\mathbf{u} \cdot \mathbf{v})}$. In this case, the Euclidean distance is related to cosine similarity by the equation:

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{2 - 2\cos(\theta)} \tag{7}$$

This indicates that Euclidean distance and cosine similarity are closely related when vectors are normalized. In this context, Euclidean distance measures the straight-line distance between vectors on the surface of a hypersphere in high dimensions. It is the default measure in HDBSCAN and a natural fit for the density-based approach.

Table 7: Considered LMs for generating sentence embeddings. The Speed is the encoding speed of sentences/sec on a V100 GPU. The Perf. is a performance metric over 14 datasets in STS tasks. The statistics are gathered from the SBERT website[1](Reimers and Gurevych, 2019).

Language Model	Dims.	Size	Speed	Perf.
fastText, trained from scratch	100	-	-	-
fastText, trained from scratch	300	-	-	-
fasttext-wiki-news-subwords-300	300	6 400 MB	-	-
allMiniLM-L6-v2	384	80 MB	14 200	68.06
paraphrase-MiniLM-L6-v2	384	80 MB	14 200	64.82
allMiniLM-L12-v2	384	120 MB	7 500	68.70
paraphrase-MiniLM-L12-v2	384	120 MB	7 500	66.01
all-mpnet-base-v2	768	420 MB	2 800	69.57
paraphrase-mpnet-base-v2	768	420 MB	2 800	67.97
all-distilroberta-v1	768	290 MB	4 000	68.73
all-roberta-large-v1	1 024	1 360 MB	800	70.23
stsb-roberta-base-v2	768	-	-	-
distilusebase-multilingual-cased-v2	512	480 MB	4 000	60.18
sentence-t5-base	768	210 MB	2 500	67.84

All considered embedding models are listed in Table 7. The models in bold are most suitable for our task and promising regarding performance in STS benchmarks. The size and speed of the models indicate their computational efficiency in embedding categories, while the embedding dimensions impact the clustering speed. All models, excluding fastText, are pre-trained models available in the SBERT repository.

[1]See also the SBERT website for performance metrics and model statistics: https://www.sbert.net/docs/sentence_transformer/pretrained_models.html#original-models

For fastText, we use a model pre-trained on Wikipedia and a news dataset. The large model size of 6 400 MB is due to the pre-trained model storing the word and subword vectors for a very large corpus, unlike current models, which store model weights. Additionally, we will evaluate training two fastText models from scratch using our data, one outputting 300 dimensions and the other outputting simpler 100-dimensional embeddings. Generally, word embedding models are lightweight and faster than recent models. The low dimensions additionally improve clustering speed.

For MiniLM and MPNet, we will use two types of models: models prefixed with "all" are general purpose models suitable for various STS and semantic search tasks, and models prefixed with "paraphrase" are finetuned for paraphrasing tasks. While the general-purpose models are much more popular, the paraphrasing models may be more suitable for clustering. The MiniLM-L6 models are the smallest at 80 MB and the fastest encoders at 14 200 sentences per second. The general purpose model has a score of 68.06 in the performance benchmark. The performance gains in using the larger 12-layer ("-L12") models are negligible compared to the loss in speed and memory efficiency. The MPNet models are around five times slower than MiniLM models and output embeddings in 768 dimensions, which may slow down clustering. Still, these models provide high quality in various tasks and score highly on the performance benchmarks, scoring 69.57 for the general purpose model.

For BERT-based models, we consider all-distilroberta-v1 as it is a fast general-purpose model, scoring 68.73 in the performance benchmark. Additionally, we included a USE-based model as Ajallouda et al. (2022) showed that it outperformed SBERT on STS tasks with *noun phrases*, relevant for purchasing categories. However, this model has a significantly weaker performance benchmark, with a score of 60.18. The final model we consider is sentence-t5-base, a T5-based general-purpose model, which is slightly faster than MPNet and has a high score of 67.84.

We considered several aspects when selecting suitable models: performance in STS and clustering tasks, size and efficiency, output dimension size, established model performance and reliability, and the licenses under which models have been released. All SBERT models are released under the Apache 2.0 license, and fastText is released under the MIT license, allowing open use, modification and distribution, even in commercial settings.

5.1.1 fastText

fastText is based on a shallow neural network architecture similar to Word2Vec's skip-gram model but extends it with subword information. Each word is a unique vector in Word2Vec, whereas fastText represents words as bags of character n-grams. Bojanowski et al. (2017) highlight the importance of handling out-of-vocabulary words in generalization and considering the internal structure of words in morphologically rich languages such as Finnish. This breakdown into subwords resembles the tokenization process of modern LMs. The architecture includes an input layer, a hidden layer that learns the embeddings, and an output layer that predicts context words. fastText is designed for speed and scalability.

5.1.2 MiniLM

The MiniLM model is designed to distil, i.e., mimic the self-attention mechanism of a larger transformer model. The model aims to alleviate the speed and memory requirements of pretraining LLMs with billions of parameters. In training, the K-L divergence between the self-attention distributions of the teacher and student model is minimized. Previous work attempted to minimize this loss for all transformer blocks in the teacher model, while the methodology in MiniLM pinpoints the crucial self-attention in the last transformer block. Wang et al. (2020) also showed that an iterative process including middle-sized models was beneficial when creating small student models. A diagram illustrating an M-layer MiniLM student model mimicking an L-layer teacher model is shown in Figure 10. The queries, keys and values are components of the attention mechanism in transformers. During training, the models calculate attention scores using dot products by comparing queries from the current token with keys of other tokens to weight the relevant input sequence token representations (values) highly (Vaswani et al., 2017).

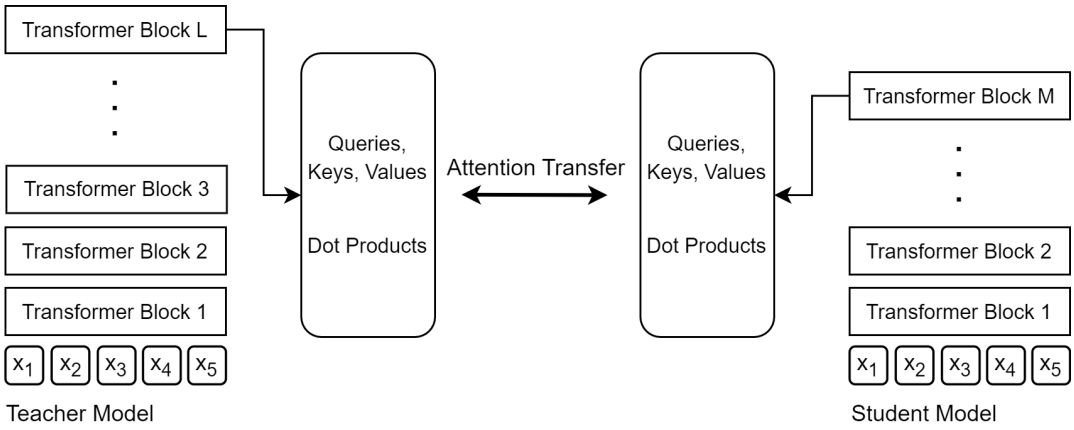


Figure 10: The MiniLM architecture. The student model with M layers mimics the self-attention mechanism of the last layer of the teacher model with L > M layers. Please refer to the original paper for more details (Wang et al., 2020).

5.1.3 MPNet

MPNet aims to address problems in BERT and XLM while preserving their benefits. In MLM used in BERT, the model predicts masked words based on visible words. MPNet enhances this process by separating the positional information and content of words, thus keeping track of the positions of the masked words. XLM uses an auto-regressive model, which reads the input sequence from left to right when predicting the masked words. PLM addresses this limitation by permuting the input sequence so that the words are predicted based on many permutations of the input sequence. MPNet uses an auto-encoding model that can use full contextual information and instead leverages permutation to vary which words are masked and in which order they are predicted.

The MPNet architecture can be seen in the diagram in Figure 11, displaying masked tokens, permutation and positional encoding.

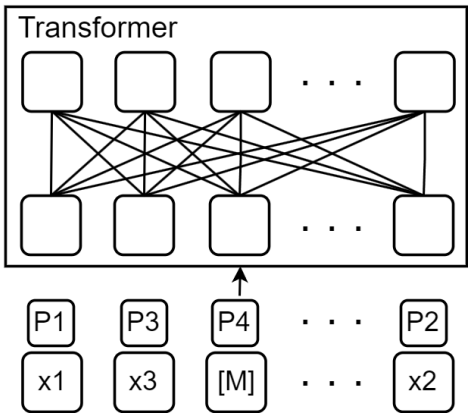


Figure 11: The MPNet architecture. x_i are permuted input tokens, P_i are positional encodings and `[M]` are masked tokens. Please refer to the original paper for more details (Song et al., 2020).

5.2 Dimensionality reduction

Dimensionality reduction algorithms are commonly used to visualize high-dimensional data. They can also be useful in preprocessing, removing noise and complexity from data, and making identifying patterns easier. However, information loss is inherent when reducing dimensions, and improper techniques may change or warp the data substantially. Principal component analysis (PCA) is a widely used linear dimensionality reduction algorithm (Abdi and Williams, 2010). PCA outputs orthogonal principal components, linear combinations of the original dimensions. Essentially, the algorithm preserves dimensions explaining most of the variance while dropping redundant dimensions that explain little to no variance. The principal components are ranked, allowing users to decide how many to use.

t-SNE is a nonlinear dimensionality reduction method mostly used for visualizing data in 2D or 3D. Unlike PCA, it focuses on preserving local structures, such as small clusters and local distances, but the global structure may be distorted. UMAP (McInnes et al., 2020) is another nonlinear dimensionality reduction algorithm based on manifold learning. UMAP can preserve both global and local structures effectively in some settings. It assumes that data has a manifold structure in high dimensions, which can be projected onto lower dimensions. For example, a curved surface of a ball in 3D can be flattened to a sphere in 2D. UMAP has two main parameters: `n_neighbors` and `min_dist`. The `n_neighbors` parameter controls the size of the local neighbourhood considered when approximating the data structure. A smaller value focuses on local details, while a larger value balances local and global structure preservation. The `min_dist` parameter controls how tightly UMAP packs points together, with a smaller value resulting in a more clustered and tighter representation

of points. While UMAP may be more versatile and balanced than PCA and t-SNE, it has drawbacks. UMAP is highly sensitive to its parameters, requiring careful selection to balance the preservation of local and global structures. Additionally, UMAP does not preserve density variations reliably, which may result in inconsistencies in density-based clusters provided by HDBSCAN.

The embedding dimensions in Table 7 can be unnecessarily large for our relatively short textual data points, especially the category labels. The SBERT architecture is meant to be used with natural language sentences, which tend to be longer than the category labels. Additionally, as discussed in the literature review, some studies indicate that embedding dimensions of modern language models are unnecessarily large (Wang et al., 2023). However, we decided to scope out dimensionality reduction from this work. As described above, the global approach in PCA may not be suitable for this task, and t-SNE results in excessive reduction to low dimensions. UMAP, on the other hand, requires extensive testing and tuning of parameters. Instead, we use UMAP to visualize clusters.

5.3 HDBSCAN Clustering

HDBSCAN (Campello et al., 2013) is the hierarchical extension of the DBSCAN algorithm. This density-based clustering algorithm groups points in high-density regions while setting points in low-density regions as outliers. The `epsilon` parameter, set by the user in DBSCAN, defines the maximum allowed distance between two points, i.e., the minimum distance for high-density regions. HDBSCAN enhances the approach by creating a hierarchy of clusters, allowing for clusters of different density levels and optimizing `epsilon` automatically to keep the most robust clusters. HDBSCAN is a versatile algorithm that is robust to varying density levels and noisy data, and it is computationally efficient, even with high-dimensional data. Setting points as outliers is also very beneficial in some datasets. In our problem, we consider this a natural cutoff between high-quality, evident mappings of categories and outlier categories.

Campello et al. (2013) describe the algorithm as follows. HDBSCAN initially stores the data in a graph, where each data point is connected with weighted edges. These weights are based on mutual reachability distance, which is the maximum of two values: the core distance of each point and the distance between the points. The core distance is the distance to the kth nearest neighbour, where k corresponds to the parameter `min_samples`, set by the user.

The algorithm then builds a minimum spanning tree from this graph. This tree connects all points with the smallest total edge weights without forming cycles. Sorting the minimum spanning tree by mutual reachability distance results in the hierarchy of clusters, also called the dendrogram. This hierarchy is then condensed by iterating through mutual reachability distances and applying cluster splits. The clusters at each iteration are kept or discarded based on the fixed parameter `min_cluster_size`. Finally, the algorithm selects the most stable clusters from the hierarchy during this process.

5.4 Evaluation metrics

We collect five descriptive statistics of the clusters to understand their characteristics and underlying patterns. Key statistics include the number of clusters k, mean cluster size $\bar{n}_C$ and median cluster size $\tilde{n}_C$. We also collect the mean and median for the number of categories from distinct taxonomies per cluster, denoted by $\bar{t}_C$ and $\tilde{t}_C$, respectively. These numbers are an early indication of whether the clustering was successful and can be compared to expectations, considering the dataset size and number of taxonomies included.

A custom manual validation process is also implemented to ensure the business relevance of the output clusters. We carry out the validation such that each category confirmed by a human validator to belong to its cluster in terms of business relevance is marked with a `1`. In contrast, mismatching categories are marked with `0`. `1/0` Accuracy is the ratio between the total number of correctly clustered categories and the number of considered categories. If only a subset of clusters is validated, the ratio between the total number of correctly clustered categories and the number of validations can be used to estimate this accuracy. The coverage metric is the ratio between the number of categories in clusters and the number of considered categories. The missing points are either outliers or belong to clusters with a single taxonomy, which are filtered out. Combining the `1/0` Accuracy and coverage, we get the total number of correctly clustered categories M:

$$M = NCA_{1/0} \tag{8}$$

$$\hat{M} = NC\hat{A}_{1/0}\ , \tag{9}$$

where N is the number of considered categories, C is the coverage, $A_{1/0}$ is the `1/0` Accuracy and $\hat{A}_{1/0}$ is the estimated `1/0` Accuracy. M/N is the proportion of correctly clustered categories. Ultimately, these are the most important business metrics.

The `1/0` validation process is somewhat limited as it only checks whether a category fits the given cluster without considering if it would fit better in another cluster or if clusters should be merged or split. Clusters that should be split receive low scores. This introduces bias in the accuracy metric, potentially favouring more granular clusterings. However, we complement this by using other statistics, metrics and a thorough review process. This issue is further addressed in the results section.

In addition to the coverage and accuracy metrics, we use five unsupervised metrics to evaluate clustering performance automatically. These internal (see Section 3.3) clustering metrics provide insights into the clusters' quality regarding cohesion and separation without considering the ground truth of cluster labels. The Silhouette score (SI) measures the similarity of a point to its own cluster compared to its dissimilarity to the nearest neighbouring cluster. The Davies-Bouldin index (DBI) measures the similarity ratio of clusters to their most similar cluster, considering within-cluster scatter and between-cluster separation. The Calinski-Harabasz index (CH) measures the between-cluster and within-cluster dispersion ratio. The average separation (AvgSep) measures the average distance between cluster centroids. The average compactness (AvgComp) measures the average distance between points in clusters. Formulations and references for the unsupervised clustering metrics can be

found in Appendix A.1. All metrics presented in this section and their output ranges and optimal values are summarized in Table 8.

Table 8: Cluster evaluation metrics including descriptive statistics, manual validation statistics and unsupervised metrics.

Metric	Description	Range	Opt. Value
k	Number of clusters	-	-
$\bar{n}_C$	Mean cluster size	-	-
$\tilde{n}_C$	Median cluster size	-	-
$\bar{t}_C$	Mean taxonomies/cluster	-	High
$\tilde{t}_C$	Median taxonomies/cluster	-	High
C	Coverage	[0, 1]	High
$A_{\texttt{1/0}}$	`1/0` Accuracy	[0, 1]	High
M	Num. correctly clustered categories	-	High
M/N	Propotion of M	[0, 1]	High
SI	Silhouette score	[-1, 1]	High
CH	Calinski-Harabasz index	[0, ∞]	High
AvgSep	average separation	[0, ∞]	High
DBI	Davies-Bouldin index	[0, ∞]	Low
AvgComp	average compactness	[0, ∞]	Low

6 Design of experiments

We begin the experiments by running 10 embedding models from Table 7 once in a default setting to set baseline benchmarks. Baseline runs are discussed in Section 6.1. In Section 6.2, we present a detailed analysis of label types, outlier points, hyperparameters and evaluation metrics. We also validate runs manually and review the clusters in detail. This analysis additionally provides the foundation for implementing automated grid searches, discussed in Section 6.3.

All experiments begin with the automated data scoping, preprocessing and label generation. We then embed the category and path labels with the selected LM and cluster these embeddings with HDBSCAN. The resulting clusters are post-processed to remove outliers and clusters with a single taxonomy. Finally, we collect the descriptive statistics and unsupervised metrics, and in some experiments, we validate the results manually. The clusters are sorted based on per cluster SI for manual validation.

6.1 Baseline

Machinery is a suitable industry for the baselines and detailed analysis as it provides complex categories while also having many expected trivial matches. There is no need for a domain expert in validation, and there are many taxonomies (8) but a moderate number of categories (1 314). The moderate number of categories is important to limit the effort required in manual validation.

To set baseline benchmarks, we embed the selected machinery dataset using 10 default embedding models and running HDBSCAN with default parameters. The default parameters for HDBSCAN clustering are `min_cluster_size` = 5 and `min_samples` is set to equal `min_cluster_size`. However, early experiments showed that our dataset requires relaxing these parameters, and we opted for the values `min_cluster_size` = 3 and `min_samples` = 1.

We used the FPLs for path labels and weighted the category and path embeddings equally $w_C = w_P = 0.5$. We conduct manual validation to estimate the initial coverage level C_0 and accuracy A_0, thereby setting the baseline benchmark for the number of correctly mapped categories.

6.2 Detailed analysis

This analysis aims to expand on the baseline runs by reviewing the methodology in more detail and exploring the resulting clusters in different settings. This analysis also provides insights for the grid searches. We have limited most experiments in this section to a single LM. We selected MiniLM as it offers high-quality embeddings and is computationally efficient.

We begin this exploration by inspecting and sampling clusters to understand their structure and common characteristics. We visualize clusters using scatter plots and review the descriptive statistics and unsupervised metrics to comprehensively view the data distribution and high-level patterns. Inspecting outliers is important to identify anomalies and to understand the data distribution further. We include the outlier points

in cluster visualizations to assess the robustness of clusters. Finally, we investigate imbalanced clusters where one taxonomy dominates and review their overall impact on the results.

We then compare the methods for path label generation, namely the HPLs and FPLs. We run two clusterings in the same setting, only varying the path label. We review the results and sample representative clusters for comparison in terms of quality and metrics. We also review the descriptive statistics and visualize the data to identify high-level differences in cluster robustness and noise. We determine and select the superior label generation strategy based on this analysis.

To expand on the baseline runs in default settings, we experiment with various hyperparameter values related to the embeddings and clustering to understand their impact on the results. We mainly review high-level automated statistics but also include some `1/0` manual validation (as described in Section 5.4), which requires limiting the number of runs at this stage. The parameter values and weights are selected to cover default and promising settings and edge cases. This exploration helps us prepare for a more systematic grid search, narrowing to suitable parameters and promising value ranges.

Finally, we review the unsupervised clustering metrics, which measure clustering quality without assessing ground truth labels. We compare these metrics to descriptive statistics and manually validated results. We want to assess the reliability of unsupervised metrics and the potential negative or positive correlation with coverage and validated accuracy. We consider further exploring this relationship by implementing a regression model predicting validated accuracy from unsupervised metrics. Unsupervised clustering metrics are sensitive to varying dimensions and are meant to be used with different clusterings of identical data. This is one reason for limiting this analysis to a single dataset and embedding model. Simply varying the weights of path and category labels will modify the data. However, as it is an essential aspect of this work and the underlying data is the same, we review the metrics when varying these embedding weights.

6.3 Grid search

After the detailed analysis, we conduct a grid search for the same dataset and selected LM. This allows for exploring the selected clustering parameters and embedding weights more freely. The hyperparameters related to the embedding model are the category and path embedding weights w_C, w_P. The hyperparameters related to HDBSCAN clustering are `min_cluster_size` and `min_samples`. The grid search ranges for the hyperparameters are outlined below. The default values are mentioned for clarity and do not affect the grid search.

w_C:	0.4 to 1.0 in increments of 0.1, default = 0.5
`min_cluster_size`:	{2,3,4,5}, default = 5
`min_samples`:	{1,2,3}, default = `min_cluster_size`

There are 84 unique combinations of the above parameters. However, due to the constraint that `min_samples` must not exceed `min_cluster_size`, invalid combinations were excluded (`min_cluster_size = 2` and `min_samples = 3`), resulting in 77 runs total. The primary indicator we look for in grid search results is the coverage of categories, as it sets the upper limit for the number of accurate mappings. Additionally, unsupervised metrics can be used as indicators for cluster quality, and the correlation between these metrics and coverage is reviewed across the entire grid search. With the machinery dataset and MiniLM embedding model, we can include several manually validated accuracies from runs in previous experiments. We manually validate additional promising runs found in the grid search.

We repeat the grid search for other models and datasets, comparing the highest coverages and overall statistics to baseline runs and the grid search with MiniLM and the machinery dataset. We select the best-performing settings for manual validation and final comparison of different models. We conclude the experiments with a large-scale run over the global dataset, corresponding to a comprehensive category harmonization of Sievo customer taxonomies. This run serves as a benchmark for the potential impact of this methodology.

7 Results

7.1 Baseline evaluation of models

In this section, we review the baseline clusterings for different embedding models. We embedded the 1 314 categories and path labels in the machinery dataset with each embedding model. We used default parameters for all model parameters, and clustering parameters were set as `min_cluster_size` = 3 and `min_samples` = 1. These runs' category and path weights were balanced equally by setting the weights $w_C = w_P = 0.5$. The path label used here is the FPL.

The coverage and `1/0` accuracies for the baseline runs are presented in Table 9. The overall coverage of data points in clusters is relatively low, ranging from 20% to 50%. On the other hand, the validated accuracies are high, mostly between 75% and 90%. The most important metric is the resulting number of correctly mapped categories M, also listed in the table.

These runs indicate poor performance of fastText models. The model trained from scratch with 100 dimensions results in a poor coverage of 30.2% and an accuracy of 56.9%. While the accuracy increases greatly to 84.3% with the larger 300-dimensional embeddings, the coverage was simultaneously lowered to 23.2%, leading to a similar amount of correctly mapped categories. The fastText model pre-trained on Wikipedia and news datasets shows more promise: the coverage is 39.0%, and accuracy is over 81.8%. However, these numbers fall short of the best models' coverages at around 50% and accuracies close to 90%.

Table 9: Results of baseline runs. Coverage (C), validated accuracy ($A_{1/0}$) and total number of correctly clustered categories (M) in runs with default settings. d is the dimensionality of embeddings, and machinery is the dataset used.

Language Model	d	C	$A_{1/0}$	M
fastText	100	30.2 %	56.9 %	226
fastText	300	23.2 %	84.3 %	257
fasttext-wiki-news-subwords-300	300	39.0 %	81.8 %	419
all-MiniLM-L6-v2	384	47.9 %	**87.3 %**	**549**
paraphrase-MiniLM-L6-v2	384	48.9 %	85.4 %	**549**
all-mpnet-base-v2	768	**49.4 %**	86.4 %	**561**
paraphrase-mpnet-base-v2	768	**51.2 %**	**86.6 %**	**583**
all-distilroberta-v1	768	45.7 %	**87.2 %**	524
distiluse-base-multilingual-cased-v2	512	46.2 %	76.6 %	465
sentence-t5-base	768	**49.8 %**	83.8 %	548

The general purpose MiniLM model reaches a coverage of 47.9% and accuracy of 87.3%, resulting in 549 correctly clustered categories, setting a strong baseline. The version trained with paraphrasing tasks has a slightly higher coverage but lower accuracy, resulting in very similar clusters and the exact same number of correctly mapped categories. This indicates that the paraphrase model is more sensitive in

detecting similarities, but many of these additional mappings were incorrect in our task. The higher accuracy of the general purpose model is also in line with the reported higher performance in the STS performance metric (see Table 7). This model may be more balanced and suitable for our task.

The MPNet model slightly outperforms MiniLM with 561 correct mappings, originating from a higher coverage of 49.4%. Moreover, the version trained on paraphrasing has the highest coverage, at 51.2%, without compromising accuracy, leading to 583 correctly mapped categories. This is the best result in baseline runs and shows that for MPNet, the paraphrasing model is favourable as it detects substantially more similarities, where many remain correct.

The RoBERTa and USE models suffered from low coverages of 45.7% and 46.2%, respectively. The USE model additionally has the lowest validated accuracy (76.6%) amongst pre-trained models, which aligns with the reported lower performance metric in Table 7. The T5 model performed well, with high coverage and relatively high accuracy at 83.8%, resulting in 548 correctly mapped categories, and may be interesting for further analysis. Based on these results, we selected the general-purpose MiniLM, the paraphrase version of MPNet, and T5 for further analysis.

The baseline clusterings reveal two key findings. Firstly, many points are set as outliers or entangled in single taxonomy clusters, as the maximum achieved coverage is only 51.2%. This number might vary in datasets and be explained by substantial differences between taxonomies. However, it is quite low if all categories were eventually to be mapped. Secondly, in recent transformer-based models, the validated accuracy is very high, a promising sign that the universal embeddings accurately represent the similarities of procurement categories and perform well in our task. This indicates that increasing the coverage may be important in finding more mappings.

7.2 Detailed analysis

The descriptive statistics and unsupervised clustering metrics for transformer-based models in baseline runs can be seen in Table 10. The general-purpose MiniLM model grouped the dataset into 117 clusters.

Most clusters have 3–6 categories, with a median of 4 categories per cluster. while a few larger clusters increase the mean cluster size to 5.4. The largest cluster has 16 categories. The clusters include categories from 3.2 taxonomies on average, from a total of 8 taxonomies in the dataset. At most, 6 different taxonomies were mapped together. Notably, every category was validated as correctly clustered in two 10-category clusters where this occurred. These numbers and findings for MiniLM align closely with the averages across all models. The paraphrase MPNet model found the most clusters, totalling 130, while USE only identified 108 clusters.

Most unsupervised metrics are uniform across the models, with relative standard deviations below 10% for SI, CH and DBI. The T5-based model is an outlier, having an AvgSep of 0.51 and an AvgComp of 0.34. This means that the clusters are closer to each other and more compact, indicating that the embeddings generated by T5 are highly compact and located centrally. Higher dimensions of the model embeddings may cause sparsity and less meaningful distances. MPNet does not exhibit these outlier

scores despite having the same high dimensions. Other metrics for T5 remain close to the averages and seem more robust.

Table 10: Descriptive statistics and unsupervised clustering metrics for transformer-based models in baseline runs. Outlier statistics are highlighted in red.

Metric	MiniLM	T5	Average
k	117	118	120
$\bar{n}_C$	5.4	5.5	5.3
$\bar{t}_C$	3.2	3.1	3.1
SI	0.23	0.22	0.23
CH	10.71	10.17	10.81
AvgSep	1.07	0.51	0.97
DBI	1.33	1.34	1.32
AvgComp	0.70	0.34	0.63

A subset of clusters is visualized in Figures 12 and 13. The clusters were projected onto 2 dimensions using UMAP with parameters `n_neighbours` = 9 and `min_dist` = 0.3. We display the categories but hide the path labels in these plots. The example in Figure 12 has three clusters related to company cars, fleet management and accommodation. The example in Figure 13 has a cluster for HR and some loosely related outlier points, such as employee benefits and employment services. These examples demonstrate that related themes are located closely on a global scale, while clusters are sufficiently fragmented to separate specific categories into distinct groups. Two additional cluster visualizations are displayed in the Appendix in Figures A1 and A2.

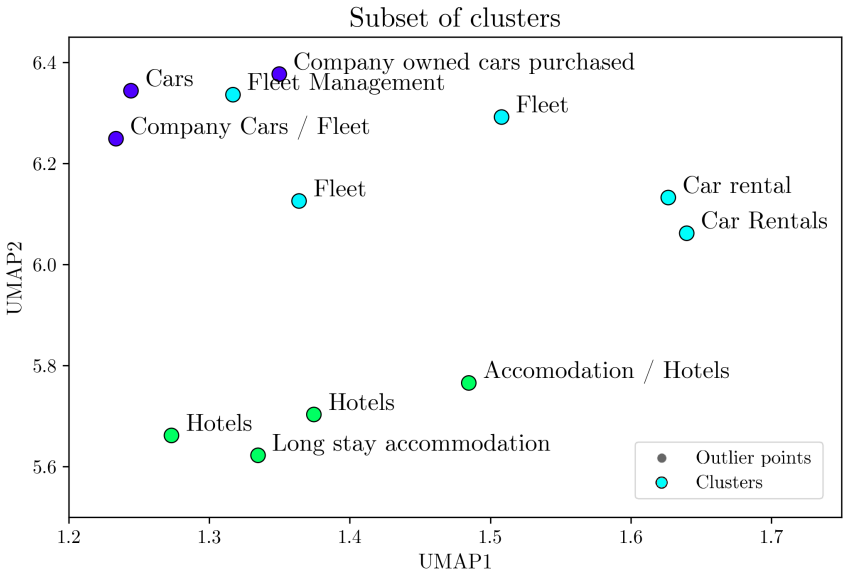


Figure 12: Example 1 of clusters in MiniLM baseline run.

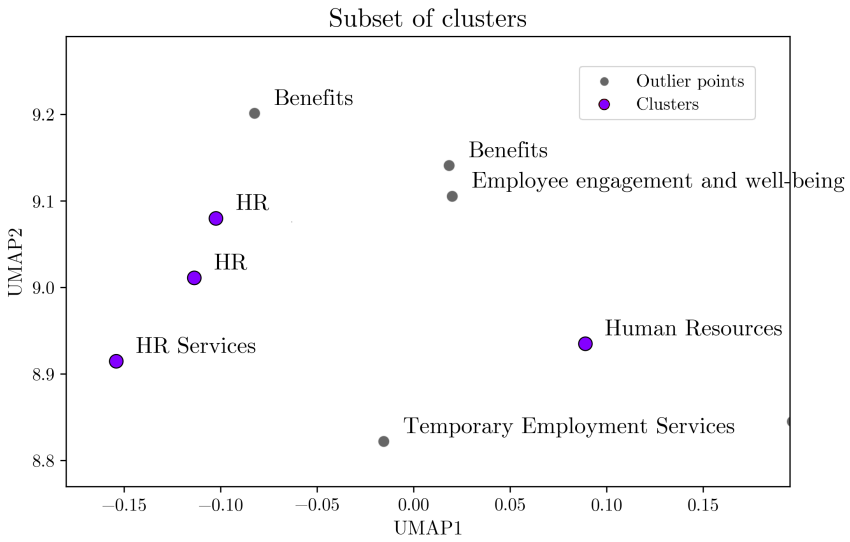


Figure 13: Example 2 of clusters in MiniLM baseline run.

The full dataset is visualized in 2D as scatter plots in the Appendix Section A.2. A colour scheme distinguishes the clusters, however due to the large number of clusters (117) some colors are very similar. Black circles and triangles differentiate outliers and single taxonomy clusters. Figure A3 presents a very local view of the dataset, where UMAP considers 2 neighbouring points to calculate local structures. The clusters are well separated, even excessively, leading to visual splitting of some clusters. The points within some clusters are very tightly packed and difficult to distinguish from single points. This plot fails to reveal any global patterns.

Figure A4 offers a moderately local view of the dataset, where UMAP considers 9 neighbouring points when calculating local structures. The clusters are expanded and easier to see but not as well separated. Outliers can be seen as points in low-density areas. However, on the right side of the scatterplot, there are also many outlier points in seemingly high-density regions. This can be explained by the global view, where very local densities are difficult to judge, or by the lost distance information when points are projected into 2D.

Finally, Figure A5 highlights global structures by considering 20 neighbouring points when calculating the local structure. This scatterplot resembles the previous one, indicating that Figure A4 already emphasized global patterns. This setting typically results in over-smoothing, causing increased overlap between clusters and obscuring local details. Projecting high-dimensional data onto 2 dimensions is challenging. However, closer examination of these scatterplots, such as the examples in Figures 12 and 13 reveal that clusters are generally well-separated, and similar themes appear in closeby regions.

The number of data points in multi-taxonomy clusters determines the coverage. Typically, most clusters have categories from multiple taxonomies, as similar categories repeat across customers rather than within the same taxonomy. The number of outlier points is the main driver for low coverage, and having close to half of the data points

as outliers is not common in density-based clustering. The outliers explain 85% of missing points in baseline runs, while the remaining are in single taxonomy clusters.

Causes for the high number of outliers could be the high dimensions and sparsity of data, limited actual overlap between taxonomies, or uniformity of data with overall high noise in categories and paths. The clustering parameters also impact the number of outliers. However, the parameters are quite relaxed, as described earlier and further in the grid search section (7.3). The points in single taxonomy clusters result from a small number of similarities within taxonomies, which is expected. Less than 10% of all categories are in single taxonomy clusters.

Table 11: Example single taxonomy clusters.

Categories in single taxonomy clusters
Indirect/Capex/Opex > Other services and OPEX > Bank charges & fees
Indirect/Capex/Opex > Other services and OPEX > Membership fees
Indirect/Capex/Opex > Travel > Visa Fees
Steel & Parts Subcontracting > Steel and structures > Square bars
Steel & Parts Subcontracting > Steel and structures > Round bars
Steel & Parts Subcontracting > Steel and structures > Flat bars

We investigated single taxonomy clusters and outlier points from the MiniLM baseline clustering. Of the 748 categories in clusters, 119 were in single taxonomy clusters in this run. Table 11 presents two examples of single-taxonomy clusters. The first cluster maps "Bank Charges & fees" together with "Membership fees", where in addition to the common word "fees", the taxonomy paths are the same. Additionally, the "Visa Fees" category is included in this cluster, although it belongs to a different parent category, "Travel". The second cluster is an example of various types of "bars" repeating in a taxonomy with a common parent category. These examples show that using path labels also results in bias for categories in the same taxonomy or subtree, as the paths are identical. Path weighting should be significantly lower than category weighting to avoid single taxonomy clusters.

Regarding outliers, the MiniLM baseline clustering set 566 categories as outliers. For example, "Programming" was an outlier point when there was a cluster for "Software" and "IT Consulting". Similarly, "Relocation" was an outlier when there was a cluster for "Moving Services". The outliers also include less common abbreviations, such as "DIT" or "APT". For common categories such as different versions of "Tools", many are in clusters, but some also remain in outliers. Some outliers include company specific conventions such as use of specifying words like "Other", "Resale" or "Service". We did not identify any distinct patterns or substantial noise in outliers. Many outlier categories are unique, indicating that the coverage level is better than expected. Some are loosely related to existing clusters but not clear matches. Outlier points and single taxonomy clusters can be seen in the previously discussed scatter plots in the Appendix Section A.2. Outlier points are distributed evenly across the data. Some outliers appear closely packed or in high-density regions, likely due to the aggressive dimensionality reduction to 2 dimensions.

7.2.1 Comparison of path labels

In this section, we review the hierarchical path labels. These labels are less processed than the FPLs and preserve the hierarchical information using > as a separator and include more repetition. For fastText models, which are average word embeddings, the > separator is considered its own word, whereas the pre-trained transformer-based models embed the whole sentence in one go. The path weight (w_P) is 0.5 in these runs, meaning that half of the weight is given to the paths when considering the similarities, while half is reserved for the category.

We rerun the baselines using the HPLs. The coverage increased for all models with an average 2.5% increase, except for T5, where the coverage lowered by 1.8%. The changes were largest, around 5% increases, for fastText models, while the average increase for transformer-based models is 1.5%, corresponding to 20 added categories. The change in number of clusters was ±6, excluding the 100-dimensional fastText, where 12 more clusters were found with HPLs. The number of clusters did not always increase when the coverage increased. In fact, the number of clusters for five models stayed the same or decreased. Changes in other statistics and unsupervised clustering metrics were insignificant. To evaluate the potential benefit of this increased coverage, we `1/0` validated runs for the selected best models from Table 9. The results are shown in Table 12.

Table 12: Baseline runs using hierarchical path labels for selected models.

Language Model	C	$A_{\texttt{1/0}}$	M
all-MiniLM-L6-v2	48.3 % (+0.4)	86.3 % (-1.0)	548 (-1)
paraphrase-mpnet-base-v2	51.8 % (+0.6)	85.2 % (-1.4)	580 (-3)
sentence-t5-base	47.9 % (-1.8)	84.9 % (+1.1)	535 (-13)

Despite the substantial increase in coverages for most models, the number of correctly mapped categories did not increase when using the hierarchical path labels. The decreased accuracy in MiniLM and MPNet clusterings resulted in fewer correctly mapped categories. While this decrease is very small, the change in total number of incorrect mappings is substantial, as there were more categories in total. As described above, T5 is an outlier in these runs, being the only one amongst all runs where the coverage decreased using HPLs. This induced a natural increase in accuracy, but the total number of correctly mapped categories again decreased.

This analysis indicates that the hierarchical path labels negatively impact the performance in this clustering task. The additional complexity and length of HPLs likely bring more noise than useful information. This can be seen with a substantial increase in the number of categories while the number of clusters stayed the same. In other words, the clusters are slightly larger (average +0.2 increase in cluster size) and contain more incorrect categories, leading to lower `1/0` accuracies. If the path weight were to be increased, the observed impact would likely be more pronounced. Based on this analysis, we favour the simpler and more processed flat path labels in the following experiments.

7.2.2 Hyperparameter exploration

To further expand on baseline runs, we experimented with different hyperparameters. We ran 16 clusterings using the general purpose MiniLM model and FPLs. We performed `1/0` validation for a subset of these runs.

We explored different weightings for categories and paths. We observed that when the category weight is low and the path weight is high ($w_C < 0.4, w_P > 0.6$), the quality of clusters decreases, and the coverage drops significantly. With $w_C = 0.2$ the coverage with MiniLM drops to 24.4% from the baseline coverage of 47.9%. This confirms that clustering complex path labels is far more difficult than clustering the shorter category labels and reinforces the choice of the simpler FPLs for path labels. However, this also raises the question of whether path labels should be used. The coverage remains high when $w_C = 1.0$ and $w_P = 0.0$, and descriptive statistics seem promising. This question is further addressed with the grid searches. The variations in overall statistics were low, with category weights 0.6–0.9. Finding the right balance in how much weight is given to the path labels is an interesting challenge. Figure 14 illustrates the importance of path labels in, e.g., avoiding falsely mapping homonym categories.

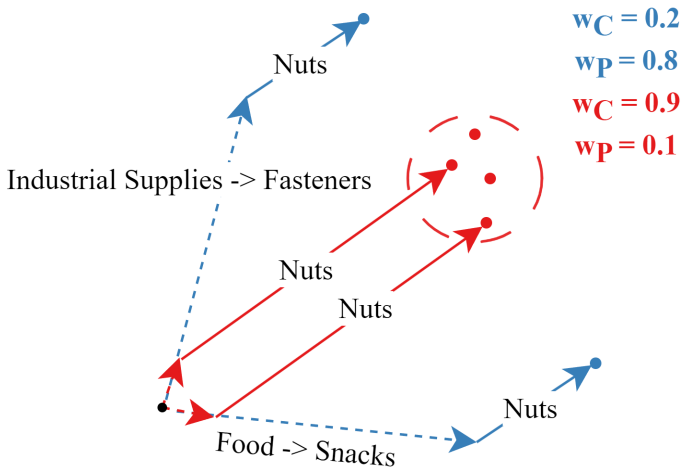


Figure 14: Impact of label weight variation. In this example, high category weights lead to incorrect mapping of homonym categories. This plot is an illustration.

We also varied clustering parameters and found that when `min_samples` is above 3 and `min_cluster_size` is above 5, clusters expand quickly, leading to minimal outliers, but most categories being placed in a few large clusters with hundreds of categories. The clustering parameters impact the number of clusters, cluster sizes and overall coverage much more than the label weights. We found that increasing `min_samples` from 1 to 2 impacts the coverage substantially, often dropping more than 5%. This change essentially filters out clusters with a single central point.

As for `min_cluster_size`, we observed the highest coverage when setting it as the minimum of 2, at around 52–53%. These runs also seemingly scored high in validated accuracies. However, reviewing these clusters reveals two issues. Firstly, the increase

in small 2–3 category clusters also decreases customer diversity, leading to one-to-one category mappings instead of the desired many-to-many mappings. Secondly, many of these small clusters could be merged, which the one-sided validation of the `1/0` Accuracy measure fails to punish. Instead, increasing the `min_cluster_size` to 4 leads to around a 3–4% decrease in coverage. These findings indicate that the clustering parameters selected in baselines were balanced and effective.

We manually validated most clusters for 10 runs to get estimates for the accuracy. Accuracies are very close, at around 85%, and contain a certain margin of error. Runs outside of the 0.6–0.9 range for w_C scored lower but still above 78%. Another outlier is a run with `min_cluster_size` set to 2, with a high yet misleading accuracy of 92%.

7.2.3 Unsupervised metrics

In baseline runs, the unsupervised clustering metrics were not comparable, as the dimensionality and distribution of the data differed across models. We now have 10 runs from the same model along with the estimated accuracies. In this section, we review the reliability of unsupervised metrics and consider implementing a regression model to estimate accuracy from metrics. This type of model would be useful considering our unlabeled datasets.

Figure 15 shows Min-Max normalized and scaled unsupervised metric scores and the estimated accuracy. The runs are ordered by coverage, which is also displayed in the plot. Despite the use of estimates, heavy scaling, and limited data points, several observations can be made. Firstly, there are no clear trends in accuracy and other metrics when runs are ordered by increasing coverage, indicating that increasing coverage does not necessarily decrease accuracy. The coverage ranged between 45% and 53% in these runs. Secondly, the unsupervised metrics show similar trends to the estimated accuracy, with the exception of the CH metric.

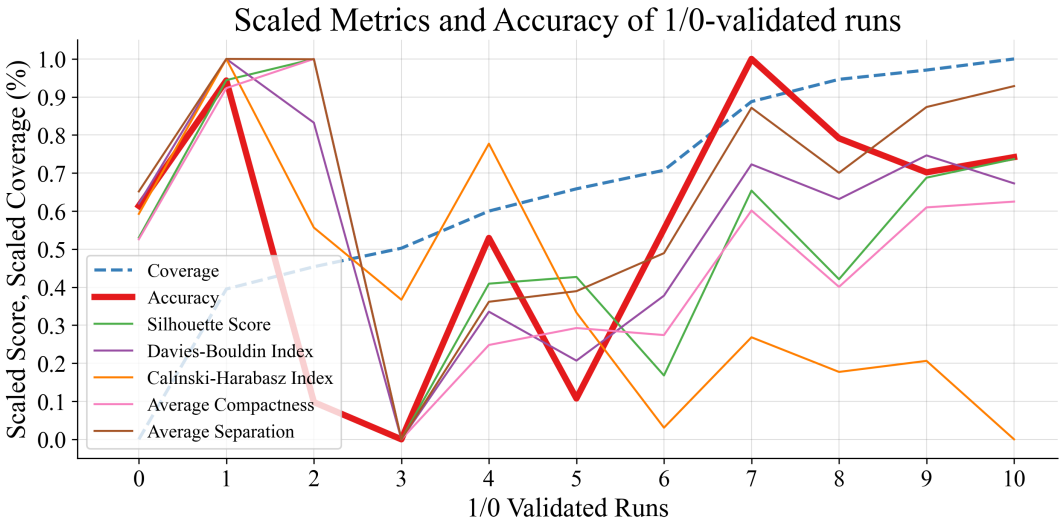


Figure 15: Scaled unsupervised metrics and validated accuracy.

We computed the Spearman correlation between the validated accuracies and unsupervised metrics, displayed as a heatmap in Figure 16. This plot confirms that CH is an outlier, exhibiting very little correlation with accuracy and other metrics. On the other hand, the remaining four clustering metrics are highly correlated. DBI and AvgComp are negatively correlated with other metrics, as low scores are optimal, whereas high scores are optimal for accuracy, AvgSep and SI. This heatmap also shows that the unsupervised metrics' correlation with accuracy is modest and may not be reliable enough to use in grid searches.

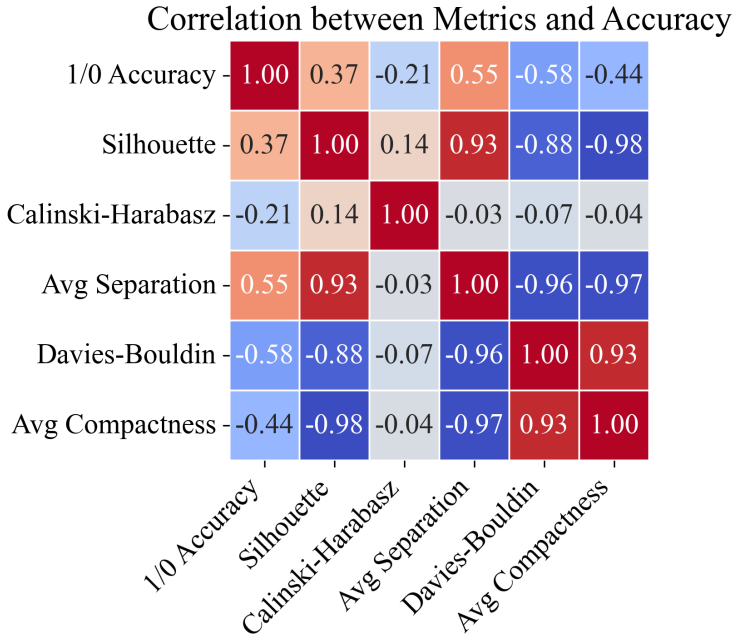


Figure 16: Spearman correlation between accuracy and unsupervised metrics in validated runs with MiniLM.

Implementing a regression model is not feasible because of the very high correlation within unsupervised metrics and the low to modest correlation with accuracy. We observed high multicollinearity, requiring the removal of most metrics as explanatory variables. Additionally, many more validations would be required to avoid overfitting. It is also not certain that the model would generalise well to other datasets.

7.2.4 Manual validations

Manually reviewing and labelling the clusters revealed several key findings. After preprocessing, there are some trivial matches, i.e., clusters with identical categories. Overall, the clusters are of good quality with related paths and categories. Some weaker clusters are connected by a common word such as "Services" or "Transport" while the categories do not match well otherwise. The clusters are output as a list, sorted by descending average SI. The clusters of MiniLM and MPNet were very similar, whereas fastText produced the most distinct clusters. The `1/0` labelling process

was an effective way of estimating accuracy. However, many clusters have some loosely related categories, making it difficult to decide on the label. Some clusters also get a low score in this setting when they would score high if split into two clusters. At the same time, this labelling process does not penalize clusters that should be merged. This can result in overly granular clusters getting a high accuracy estimate while missing several joins and larger clusters scoring lower estimates while only a few splits are needed. In MiniLM and MPNet, we also observed that good, trivial clusters are quick to label, as well as tail-end bad clusters. Median clusters require careful review and also provide the most room for optimization.

7.3 Grid search with MiniLM on machinery dataset

This section presents the grid search results with MiniLM on the machinery dataset. We review the descriptive statistics and coverage level across runs and identify the best-performing settings. We also review outlier runs and unsupervised metrics.

The analysis with MiniLM on the machinery dataset in Section 7.2 showed that any category weighting $w_C < 0.5$ decreases both the quality of clusters and especially the coverage of clusters. However, we covered values ranging from 0.4 to 1.0 with increments of 0.1. When $w_C = 1.0$, the path embeddings are discarded completely, and only the categories are clustered. For `min_cluster_size` and `min_samples`, the minimum possible values are 2 and 1, respectively, and both are covered in the grid search. Initial findings also showed that low, relaxed clustering parameter values perform well, while higher values lowered coverage substantially and later caused expansion into a few massive clusters.

The best runs in terms of coverage find over 130 clusters with close to 700 data points, covering up to 52.7% of categories. The median cluster sizes in runs with coverage above 45% range from 4 to 7 and there are on average 3 to 4 taxonomies per cluster. This seems reasonable as the dataset has 8 taxonomies, but it also indicates some repetition of the same taxonomy in clusters. The coverage drops below 30% in the worst runs. These runs only find around 30–50 clusters, which are larger on average. Low coverage runs typically had lower category weights and higher clustering parameter values. Results for the top, middle and bottom 7 runs of the grid search are shown in Table 13.

The grid search reveals that coverage is highly variable, and selecting suitable parameters is important to increase coverage. High coverage is achieved with relaxed clustering parameters. `min_samples` seems to be the main driver, as all of the best runs have the lowest value of 1 while all of the worst runs have the strictest value of 3. The best run with `min_samples` = 3 only includes 69 clusters with 34.9% coverage.

Similarly, lower values for `min_cluster_size` provide higher coverage. This pattern is not as apparent, as `min_samples` dominates the results, but it is clear that clustering parameters greatly influence the coverage, and lower values perform better. There is more variation for category weight, however there are no runs with $c_W = 0.4$ and very few runs with $c_W = 0.5$ in the top quartile of runs in terms of coverage. Grouping the results by clustering parameter pairs and reviewing the category weight, there is some indication that higher values result in higher coverage.

Table 13: Grid search results. Top, middle and bottom 7 runs in grid search with MiniLM and machinery dataset. The baseline run is highlighted in grey.

w_C	min_clu.	min_sam.	k	$\bar{n}_C$	$\bar{t}_C$	SI	DBI	C (%)	$A_{\texttt{1/0}}$ (%)	M
1.0	3	1	135	5.1	3.3	0.30	1.23	52.7	86.4	598
0.9	3	1	132	5.2	3.4	0.30	1.19	51.8	88.1	**600**
0.7	3	1	128	5.2	3.4	0.27	1.24	50.9	89.5	**599**
0.8	3	1	131	5.1	3.3	0.29	1.22	50.9	88.2	590
0.6	3	1	123	5.4	3.3	0.25	1.31	50.2	88.8	585
0.5	3	1	117	5.4	3.2	0.23	1.33	47.9	87.3	549
1.0	4	1	88	7.1	4.0	0.26	1.41	47.3	82.3	512
1.0	4	2	76	7.1	3.9	0.29	1.30	41.0		
0.6	2	2	108	5.0	3.2	0.29	1.17	40.9	89.4	481
0.6	3	2	98	5.4	3.3	0.30	1.18	40.2		
0.8	4	2	77	6.9	3.8	0.29	1.26	40.2	82.6	436
0.7	4	2	77	6.5	3.8	0.29	1.23	38.3		
0.5	4	2	61	8.2	3.7	0.22	1.43	38.0		
0.5	2	2	97	5.1	3.1	0.27	1.23	37.5		
0.9	5	3	39	10.1	4.7	0.30	1.29	30.0		
1.0	5	3	41	9.4	4.7	0.31	1.33	29.2		
0.7	5	3	42	8.8	4.2	0.28	1.27	28.2		
0.5	5	3	37	10.0	4.1	0.23	1.44	28.1		
0.6	5	3	42	8.8	4.0	0.26	1.37	28.1		
0.4	3	3	42	7.9	3.5	0.23	1.33	25.3		
0.4	4	3	36	8.8	3.7	0.23	1.39	24.2		

Manual validation is necessary to ensure that accuracy is not compromised. We first observed clear outlier runs, which are discarded from this analysis. Two runs with strict clustering parameters and low category weights resulted in 2 large clusters, including nearly all data points. While the `1/0` accuracy has some margin of error, we observed a significant decrease in the accuracy of runs with $w_C = 1.0$, as seen in Table 13. This demonstrates that only using the category labels decreases the accuracy of mappings. On the other hand, the higher coverage of runs where only categories are used compensates for the loss in accuracy.

We also discarded all runs with the most relaxed clustering parameters `min_samples` = 1 and `min_cluster_size` = 2. Reviewing these clusters revealed that while they scored high in coverage and `1/0` Accuracy, they were very small and fragmented, often pairs or triplets. This is an important finding regarding the interpretability of `1/0` Accuracy, as it fails to penalize runs that are too fragmented. Also somewhat overlooked in this work, but potential future implementation will have further constraints for small clusters to ensure anonymity and compliance with relevant regulations.

7.3.1 Unsupervised metrics and coverage

Initial review of grid search results revealed no clear patterns in unsupervised metrics. Sorting the runs by these metrics results in very varied coverages and validated accuracies in top runs. The correlation matrix in Figure 17 shows close to zero correlation between unsupervised metrics and coverage. The metrics are again highly correlated with each other. CH is an outlier, indicating a moderate negative correlation with coverage. This could mean the metric works well, as more conservative runs result in fewer but better clusters. However, we have observed that cluster quality remains high when coverage increases.

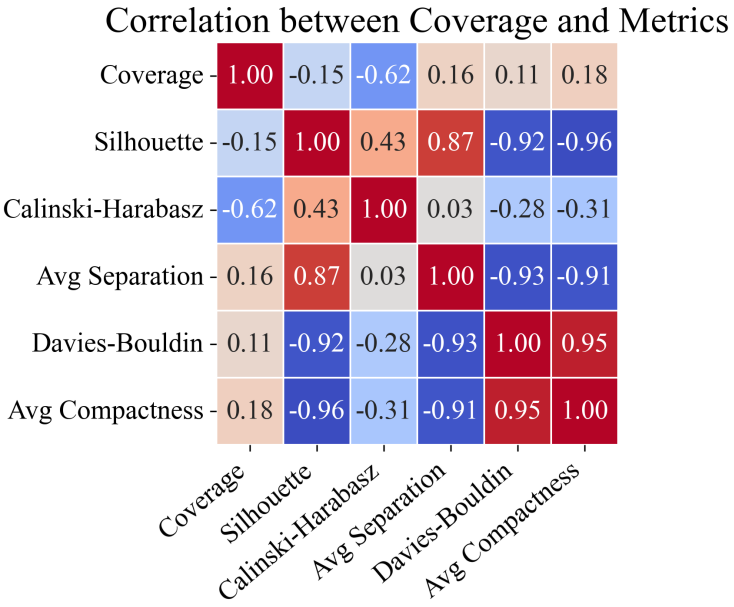


Figure 17: Spearman correlation between coverage and unsupervised metrics in grid search using MiniLM.

On the contrary, AvgSep surprisingly has a positive, albeit low, correlation with coverage. This may be caused by the relatively simple metric, the average Euclidian separation between clusters, combined with the complex, high-dimensional data points. AvgSep had a moderate positive correlation, 0.55, with validated accuracy in the previous section. Figure 18 displays two examples of scaled metric scores and the coverage level across the grid search. The runs have been sorted by ascending metric score, and the first plot shows no linear trend in coverage with DBI, while in the second plot, there is a downward trend for coverage with CH. These trends are aligned with the correlation values.

Heatmaps for coverage and DBI depending on `min_cluster_size` and w_C are shown as an example in Figures 19 and 20. The heatmap for coverage shows that higher values of w_C perform better, but there are also irregularities, such as $w_C = 0.5$, outperforming $w_C = 0.6$. For `min_cluster_size`, low values provide higher coverage, but there is variance here as well, e.g., the lower scores for low values when $w_C = 0.4$

or $w_C = 0.5$. The heatmap for DBI similarly indicates preference towards low values of `min_cluster_size` and for w_C around 0.7–0.8.

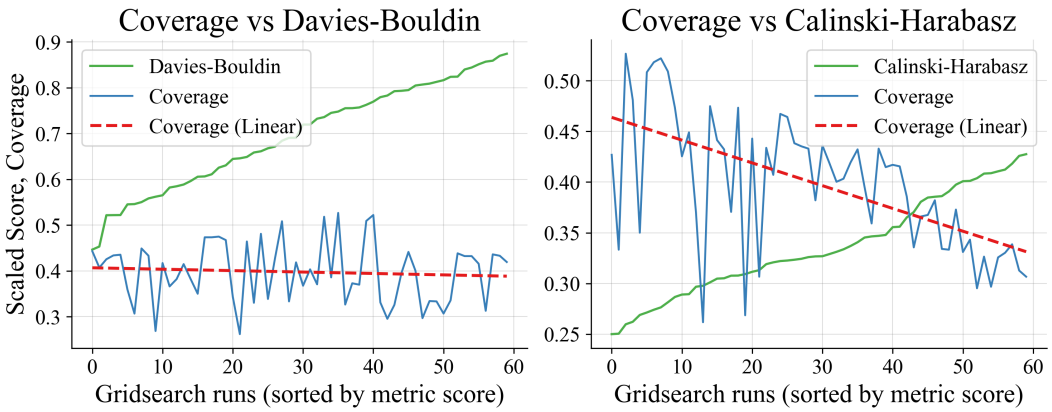


Figure 18: DBI and CH scores and coverage level across grid search with MiniLM.

These findings indicate that using unsupervised metrics in this exploratory phase is challenging due to the high variations in coverage, which significantly impact the results. The correlation between unsupervised metrics and coverage is low. The metrics could be revisited when there are numerous runs with similar coverage level to evaluate. Additionally, as shown in Section 7.2, the correlation between metrics and validated accuracy is not sufficiently high to rely on without manual validation. Similarly, we observed a low correlation between metrics and validated accuracy in the grid search.

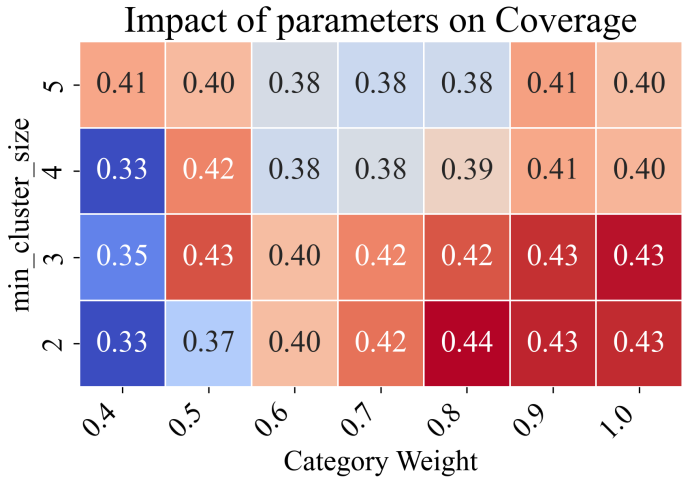


Figure 19: Heatmap of coverage in grid search. Impact of parameters `min_cluster_size` and w_C on coverage in grid search using MiniLM.

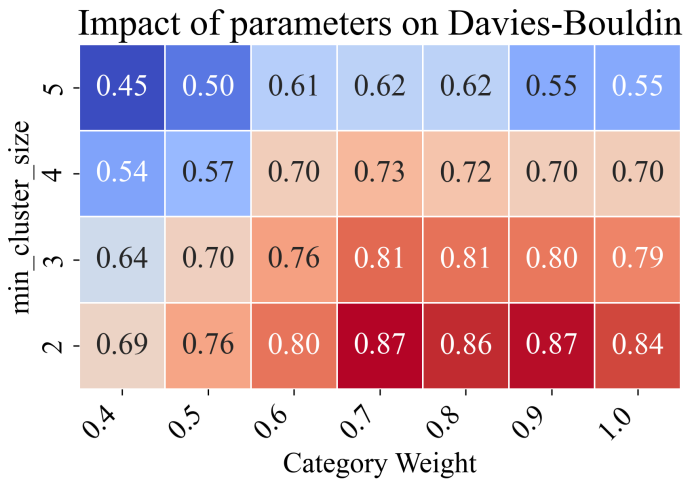


Figure 20: Heatmap of DBI in grid search. Impact of parameters `min_cluster_size` and w_C on DBI in grid search using MiniLM.

In this section, we showed that running a grid search and validating some runs manually efficiently finds well-performing settings. The clustering parameters impact coverage the most, and some runs were filtered out due to being overly granular. The best value for `min_samples` is 1, and the best value for `min_cluster_size` is 3 or 4. Regarding embedding weighting, the runs remain varied, but we found some evidence that higher values of $w_C \geq 0.8$ increase coverage, while values 0.7–0.9 align best with metrics and validated accuracy. Additionally, in earlier sections, we demonstrated that the path weight should be kept low to prevent clustering of similar paths from a single taxonomy.

7.4 Grid search with other models

We repeated the grid search using the best models from baselines: MPNet and T5. We additionally include fastText despite its weaker performance in baselines as it is remarkably different as an average word embedding model and acts as a base model. The findings in earlier sections may not apply directly to fastText, and it is interesting to review in a grid search setting.

With MPNet, we observed slightly higher coverage across the grid search than with MiniLM. Otherwise, the observations and findings are similar, confirming that the grid search is effective and reliable in optimizing parameters and identifying high-coverage runs. The best runs with MPNet are very similar to those of MiniLM, with the same parameters, above 130 clusters of similar size, and covering up to 54.4% of categories. The values 1 and 2 for `min_samples` overlap slighlty more, however 1 remains optimal and `min_samples` is the dominating parameter in terms of coverage. The worst runs have similar coverage, with 7 runs under 30% and a lowest coverage of 25.3% (see Table 13 for MiniLM runs). There was one outlier run with 2 expanded clusters. The unsupervised metrics are surprisingly similar in scale despite the large difference in embedding dimensions (see Table 9). The metrics correlations are also very similar.

The performance of T5 falls short of that of MPNet and MiniLM in terms of coverage and robustness. The highest achieved coverage is 51.6%, and 21 only runs have a coverage above 40%, whereas 34 and 33 runs reached over 40% coverage for MiniLM and MPNet, respectively. The lowest run has a coverage of 22.4%, and over 20 runs fail to reach 30%. We identified 6 outlier runs with a few huge clusters. This indicates that MPNet and MiniLM are more robust towards stricter clustering parameters, identifying mid-sized clusters before expanding into huge clusters.

The coverage across the search with fastText is lower compared to other models. The highest coverage is 46.3%. The clusters are similar in size, but fewer are found, 121 being the highest number. 27 runs failed to reach 30% coverage, and only 4 runs reached 40%. A major difference is also seen in outliers, as 12 runs with stricter clustering parameters result in 2–3 huge clusters with most of the data points. One key differentiating finding is that category weight seems to have no impact in fastText runs, whereas for the other models, the lowest $w_C = 0.4$ clearly performed worse. This indicates that the averaging fastText embeddings capture less of a difference between the short categories and the longer path labels.

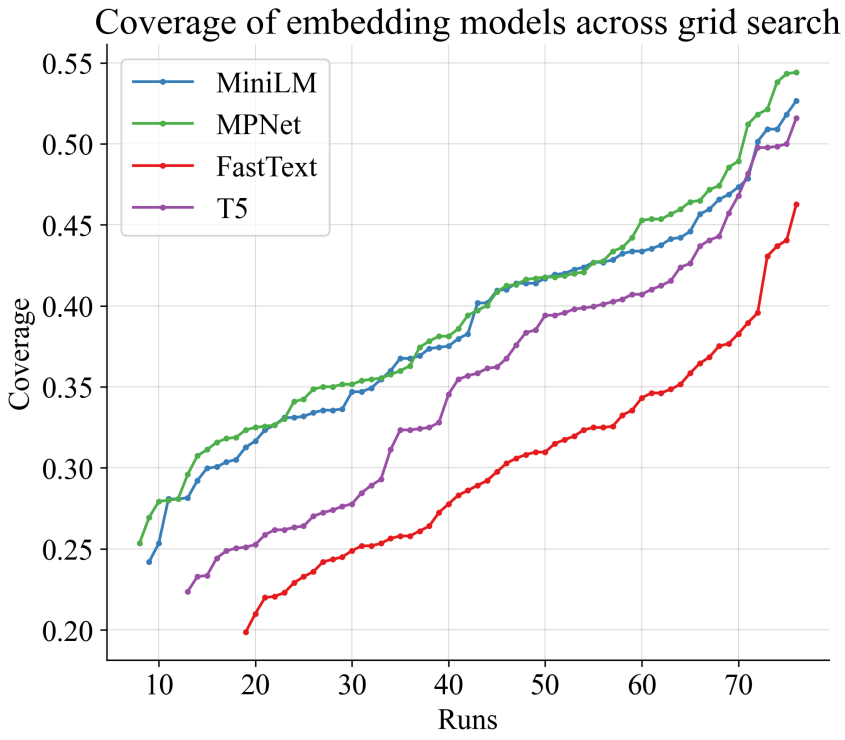


Figure 21: Sorted coverages across grid searches with different embedding models. Outlier and overly fragmented runs have been removed.

Figure 21 displays each embedding model's sorted coverages in grid search runs. Note that the earlier-mentioned outlier and overly fragmented runs have been removed, causing the lines to shift to the right. MPNet and MiniLM are close across the grid search, but the best runs for MPNet have higher coverages. Conversely, for T5, the overall coverage is much lower, while the best runs reach close to the coverage level of

MiniLM and MPNet. Coverage in fastText runs clearly falls short of the other models.

We conducted several manual validations for MiniLM runs in previous experiments. We observed that high accuracy is maintained in runs with high coverage. The grid searches also revealed significant variation in coverage levels, with top runs exhibiting much higher coverage across all models. This indicates that even if a very high accuracy is found in a moderate coverage run, the total number of correct mappings would be insufficient due to the lower coverage. Therefore, given the limited time for manual review, we selected a few runs for all models with the highest coverage and promising statistics for manual validation. Outliers and overly fragmented runs were not considered. The results are shown in Table 14.

Table 14: Selected best performing runs. High coverage runs from grid searches have been selected for manual validations. Overly fragmented runs have been removed. Baseline runs are highlighted in grey.

Model	d	w_C	min_clu.	min_sam.	k	C (%)	$A_{1/0}$(%)	M
MPNet	768	0.9	3	1	137	**54.3**	88.1	629
MPNet	768	0.6	3	1	135	**54.4***	86.9	621
MPNet	768	1.0	3	1	137	**53.8**	86.7	613
MPNet	768	0.8	3	1	135	51.8	**89.6***	610
MPNet	768	0.7	3	1	134	**52.1**	88.5	606
MiniLM	384	0.9	3	1	132	51.8	**88.1**	600
MiniLM	384	0.7	3	1	128	50.9	**89.5**	599
MiniLM	384	1.0	3	1	135	**52.7**	86.4	598
T5	768	0.6	3	1	124	51.6	87.3	592
MiniLM	384	0.8	3	1	131	50.9	88.2	590
T5	768	0.8	3	1	131	50.0	**89.5**	588
MiniLM	384	0.6	3	1	123	50.2	**88.8**	585
MPNet	768	0.5	3	1	130	51.2	86.6	583
T5	768	0.7	3	1	129	49.8	87.9	576
T5	768	0.9	3	1	127	49.8	87.9	575
MiniLM	384	0.5	3	1	117	47.9	87.3	549
T5	768	0.5	3	1	118	49.8	83.8	548
T5	768	1.0	3	1	124	48.2	86.3	546
MPNet	768	0.8	4	1	90	48.6	84.0	536
MPNet	768	0.6	4	1	97	48.9	83.2	535
fastText	300	0.8	3	1	115	43.7	83.6	480
fastText	300	0.9	3	1	117	44.1	82.7	479
fastText	300	0.5	3	1	105	39.0	81.8	419

The table is sorted by the total number of correct mappings, M. The highest coverages and validated accuracies are highlighted in bold and marked with an asterisk. Note that, e.g., MPNet and MiniLM have further runs with higher coverage than the included runs of T5 and fastText. All runs have `min_samples` = 1, and most runs have `min_cluster_size` = 3, while two runs for MPNet with value 4 are included. MPNet performs the best, producing above 600 correct mappings in best runs, with a

maximum of 629. MiniLM is quite close, finding 590–600 correct mappings. The best run for T5 finds 592 mappings, and the best run for fastText only produces 480 correct mappings. The validated accuracy in transformer-based models is around 86–89%, the highest accuracy for each model being around 89.5%. Validated accuracies in fastText are lower, around 81–84%.

In terms of w_C, 0.5 and 1.0 provide lowest validated accuracy, while values between 0.7–0.9 yield the best results. Runs with $w_C = 1.0$ demonstrate that clustering categories results in more but lower quality mappings. Although the significant increase in coverage mostly compensates for the loss in accuracy, incorporating some path weight is likely beneficial. The higher accuracies observed with w_C between 0.7 and 0.9 suggest that a balanced approach leads to more reliable, higher quality results.

In conclusion, MPNet is the best-performing model in our task. T5 performs worse than both MiniLM and MPNet, while all transformer-based models clearly outperform fastText. Validated accuracies remain high in runs with high coverage, emphasizing the importance of expanding coverage to increase the total number of mappings. Coverage is influenced by clustering parameters, which likely have different optimal values for different datasets. Additionally, incorporating path labels with a small weight improves the accuracy of mappings. The optimal value range for w_C is around 0.7 to 0.9.

7.5 Large scale run and business impact

This section presents the mapping of the global dataset. We discuss the additional challenges the dataset provides and the impact of this exercise. We present the results of a limited grid search and evaluate the overall impact and effectiveness of the methodology in a business context.

We evaluate the methodology using the global dataset (see Table 5). This dataset has over 140 thousand categories and presents new challenges compared to the smaller industry-specific datasets. Mapping categories of the global dataset provides a potential starting point to *mapping all customer categories* and closing the gap to the long-term goal: extensive community benchmarks. This exercise assesses the feasibility and effectiveness of the methodology presented in this work to reach this goal. The earlier sections have shown that this methodology produces mappings of high accuracy in smaller datasets. The mappings include around half of the categories, a modest coverage that was substantially increased using grid searches.

The automated hierarchy selection process decreases the global dataset to 23.5 thousand categories of similar levels. In contrast to earlier experiments, the clusters' expected size is unclear. Typical indirect spend related categories may be found in all 106 taxonomies, while industry-specific production related categories may only exist in a few customers. HDBSCAN supports clusters of varying sizes as clusters are grouped based on the density of points, and the size-related hyperparameters define the minimums. Still, setting, e.g., `min_samples` to a suitable value is difficult.

Moreover, the dataset includes taxonomies at all extremities, ranging from the shallowest to the deepest hierarchies and taxonomies with tens of thousands of categories. Some taxonomies are outliers and have extensive company-specific

conventions, for example. The line between a high-quality cluster and loosely related mappings is broader, and the optimal cutoff likely varies case by case.

We performed a limited grid search to cluster the global dataset. We used the general-purpose MiniLM model due to its efficiency and lower embedding dimensions. We varied the clustering hyperparameters and explored stricter settings such as setting `min_cluster_size` to 10 and `min_samples` to 3. We ranged the category weight from 0.6 to 0.9 and used FPLs. However, there are over a thousand clusters with more than 10 000 categories, so manual validation is not feasible at scale. We focused on descriptive statistics and the coverage level in this grid search.

We use a slightly modified definition for coverage in this experiment, as further business limitations are applied when filtering the produced mappings. The highest coverage runs provide mappings for around 10 400 categories, corresponding to 44.4% of included categories. The category weight is high in these runs, 0.8 or 0.9, and clustering parameters are relaxed. The most relaxed parameters result in substantially lower coverage. In high coverage runs, the clusters are very small, with 5–8 categories on average, much lower than the number of taxonomies (106). However, the largest clusters have above 80 categories, while the maximum is 162, considering all runs in the grid search. There are 4–6 taxonomies per cluster, 34 at most in best runs. In other runs, up to 52 taxonomies are found in a single cluster. The number of clusters varies greatly, from 1 500 to above 2 000 in best runs. The smallest number of clusters across all runs was 370, with a much larger average cluster size of 20.2, resulting in a modest coverage of 31.7%

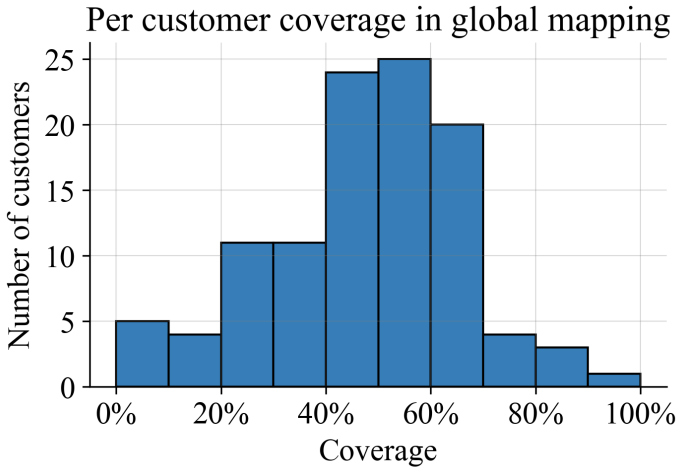


Figure 22: Coverage level per customer in mapping of global dataset.

We reviewed per-customer statistics for one of the best runs, with 42.9% overall coverage. We selected this run as it has high coverage and does not have overly fragmented clusters. The average coverage per customer is higher, 47.5%, as many smaller taxonomies have been mapped effectively, while some large taxonomies have lower coverage. The coverage is between 40–70% for most customer taxonomies, as shown in the histogram plot in Figure 22. There are 8 customers with a +70% coverage

and 4 with a +80% coverage, the highest being 92.7%. 9 customers have less than 20% of categories mapped.

We sampled clusters from this run and observed they are very high quality. With tens of thousands of categories, there are numerous trivial clusters where categories are exact or very close duplicates after preprocessing. We also reviewed the largest clusters; most are of very high quality. Some generic clusters, like "Services" have incorrect mappings and we found a few worse clusters with categories in other languages. Overall, sampling clusters indicates very high quality. The challenges described above were mitigated by the larger dataset, which resulted in numerous trivial clusters and increased similarities across categories. Larger datasets typically benefit statistical and machine learning methods, but we expected more noise-related issues. However, this analysis suggests that the methodology is perhaps better suited for larger datasets as similarity comparisons increase significantly. The methodology could also be adjusted to increase coverage further.

With 42.9% global coverage and an expected 90% accuracy in mappings, there is a clear opportunity to reduce manual work linearly. Mapping all available taxonomies is estimated to require hundreds of full-time employee days. This approach demonstrates the feasibility of delivering time and cost savings to Sievo during the mapping process. Implementing advanced mapping algorithms can streamline the process, minimizing errors and improving accuracy. By reducing manual effort, Sievo can reallocate resources to higher-value tasks, enhancing overall productivity. Additionally, the scalable methodology supports growing data assets and changes in taxonomies, ensuring ongoing efficiency as the company expands.

7.6 Summary of findings

We set the initial coverage and accuracy benchmarks in baseline clusterings. MiniLM, MPNet and T5 scored the highest in baselines, mapping around 550–580 categories correctly. fastText models lagged behind, especially in the coverage of categories in clusters. The baselines revealed that coverage overall is low, as only around half of the considered data points are mapped. In contrast, the quality of clusters is high as the validated accuracies were above 80%.

We conducted an in-depth analysis, first reviewing the patterns and distribution of clusters. We found that the clusters are granular, and most of the missing points are set as outliers, while a small subset belongs to single taxonomy clusters. We also found that most of the outliers are very unique categories or loosely related to existing clusters, indicating that the achieved coverage is, in fact, high considering the source data quality. This reinforces the choice of HDBSCAN, which effectively identifies high-quality clusters and appropriately labels many data points as outliers.

We evaluated hierarchical path labels and found that they did not provide significant improvements compared to flat path labels. We explored clustering hyperparameters and identified promising ranges. We manually validated 10 runs and showed that the accuracy was stable and remained high with category label weights between 0.6–1.0. We also note that the `1/0` validation fails to punish overly granular clusterings. Finally, we reviewed the reliability of unsupervised metrics and showed that the correlation

with validated accuracy is low.

We further explored hyperparameter values in a grid search with MiniLM. We found that the clustering parameters impact the coverage greatly, and the grid search effectively found the best settings. Lower, relaxed values of clustering hyperparameters provide the best coverage, but we also identified clusterings that were too granular. The optimal category and path weighting remains uncertain, however category weights between 0.7–0.9 are most common in best runs. One reason for this uncertainty is the many tradeoffs in using path labels that we have identified during this work. Path labels provide useful information and context, increasing the accuracy of mappings but also induce bias towards categories within the same taxonomy, lowering coverage. We also observed a significant drop in coverage with high path weights as shorter category labels contain less noise and are easier to cluster. In conclusion, we found that including path labels with a low weight in the representation is beneficial. We also identified outliers, where clusters explode, and clusterings that score very highly while being overly fragmented. We discussed unsupervised metrics and found manual validation of handpicked runs more effective and reliable in identifying high-accuracy runs.

In subsequent grid searches, we found very similar settings to perform well for MPNet and T5. MPNet performed the best out of all models, with noticeably higher coverage while maintaining high accuracy. The best runs identified close to 630 correct mappings, a notable increase from baseline runs. Coverage emerged as the main factor influencing the number of mappings, as accuracies remained high even with varying coverage levels. Lower coverage in runs with T5 resulted in worse performance than MiniLM and MPNet. The performance of fastText substantially improved from its baselines, but it still fell short compared to modern transformer-based models both in coverage and accuracy.

Finally, we attempted to map the global dataset with 106 taxonomies and 23 thousand categories to benchmark the potential business impact of this work. Despite the many challenging aspects of the large and imbalanced dataset, the clusters were of very high quality, and the coverage remained moderately high. The larger dataset allowed for discovering numerous trivial clusters due to the increasing number of similarity comparisons. We showed that most taxonomies had 40–70% of categories mapped, some reaching a coverage above 80%, a great result considering the high accuracy and fully automated methodology. However, for some taxonomies, the coverage was below 20%. This experiment proved a clear opportunity to reduce manual work at a large scale.

8 Conclusion

In this thesis, we introduced a hierarchical text clustering problem in the procurement analytics field. We implemented state-of-the-art sentence embedding models to accurately represent textual data and found the best performance with MPNet. We used a density-based clustering algorithm, HDBSCAN, to cluster semantically similar purchasing categories. The final mappings with MPNet cover 54.3% of categories in the machinery dataset with nearly 90% cluster validation accuracy, demonstrating the feasibility of mapping similar categories across customers. The coverage is high, given the quality of the source data and the observed uniqueness of outlier categories. With the MiniLM model, the dataset was mapped with a coverage of 51.8% and an accuracy of 88.1%. Additionally, the global dataset, containing tens of thousands of categories was mapped with up to 44.4% coverage and a similarly high accuracy estimate, highlighting the scalability of the approach.

This automated solution indicates direct cost and time savings by reducing manual efforts in the mapping process. This work contributes to the NLP field by evaluating state-of-the-art sentence embedding models on a clustering task with specialized categories related to the procurement field. Additionally, we propose an automated solution to mapping purchasing categories for community benchmarks, contributing to the spend analysis field.

The research problem had several challenges. The dataset is highly specialized, with purchasing categories from diverse industries. The data points are organized into complex hierarchies of varying sizes. We aim to find business-relevant many-to-many category mappings despite the absence of ground truth or labelled datasets.

In Section 3, we reviewed the literature on the development of language models and identified the relevant state-of-the-art sentence embedding models, such as SBERT, RoBERTa, MiniLM and MPNet. These models provide high-quality embeddings for sentences, capturing meaning and similarity. We also explored text clustering, where this context is quite unique, considering sentence length, hierarchical structure and availability of cutting-edge language models. We identified HDBSCAN as a suitable algorithm, as density-based clustering can handle high dimensions and detect outlier points.

We presented a data scoping strategy in Section 4. Data scoping was a crucial step to process the complex data from imbalanced taxonomies, enabling business relevant mappings. Categories with suitable levels of detail are automatically selected from hierarchies of different sizes and depths. This scoping decreased imbalance in the datasets, removed redundant repetition and enabled business relevant clusters to be found. We applied typical preprocessing steps in NLP to normalize the data. Additionally, we constructed path labels, to study if hierarchical context from the taxonomy can be used to enrich the data points.

In Section 5, we presented various SBERT sentence embedding models and fastText – a standard average word embedding model. We used a fastText model pre-trained on Wikipedia and news datasets. We presented HDBSCAN and discussed the evaluation metrics. In Section 6, we discussed the design of experiments: setting baselines, followed by various experiments and concluding in a grid search.

The results are reviewed in Section 7. In baselines, we found that the validated accuracy of clusters is relatively high while coverage is low. We visualized the clusters, investigated outlier points and observed that most outlier points are unique and do not relate to the clusters. We observed that clustering category labels is intrinsically easier than clustering longer path labels with parent categories across the hierarchy. However, we also showed that including the path labels with a small weight increased validated accuracy. We discussed unsupervised clustering metrics and manual validation of clusters. Unsupervised metrics did not correlate well with accuracy validated based on the business relevance of mappings. Finally, we narrowed down to the most promising models, namely a general-purpose MiniLM model, an MPNet model trained on paraphrasing, a T5-based model and a pre-trained fastText model.

We concluded with a grid search and final validations of top runs to determine the best model. The grid search was most useful in identifying hyperparameters that increase coverage. Manual validations showed that while some clusterings were too fragmented, the accuracy mostly remained high. We demonstrated that transformer-based pre-trained sentence embedding models (SBERT) outperformed the classical approach of averaging word embedding vectors, such as fastText. The SBERT models have been pre-trained on various STS datasets and tasks, and their architectures are based on revolutionary models such as BERT. We found the best performance with MPNet, closely followed by MiniLM.

This work focused exclusively on pre-trained language models. Curating a labelled dataset for fine-tuning or adapting sentence embeddings to the procurement domain could enhance performance (Wang et al., 2022, 2021). We used HDBSCAN for clustering but did not evaluate other algorithms; further exploration of clustering methods may provide valuable insights. While our data scoping resulted in business-relevant mappings, expanding to most of the hierarchy is a natural next step in increasing overall coverage. Additionally, reducing the high dimensions of modern embeddings could be interesting (Aggarwal et al., 2001; Kusupati et al., 2022). Lastly, while our approach leveraged parent categories from the hierarchy, it did not utilize graph-based methods, which could be an interesting direction for future research.

References

Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459, 2010. ISSN 1939-0068. doi: 10.1002/wics.101. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101.

Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer. ISBN 978-3-540-44503-6. doi: 10.1007/3-540-44503-X_27.

Lahbib Ajallouda, Kawtar Najmani, Ahmed Zellou, and El habib Benlahmar. Doc2Vec, SBERT, InferSent, and USE: Which embedding technique for noun phrases? In *2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, pages 1–5, March 2022. doi: 10.1109/IRASET52964.2022.9738300. URL https://ieeexplore.ieee.org/abstract/document/9738300.

Argyris Argyrou. Clustering hierarchical data using self-organizing map: A graph-theoretical approach. In José C. Príncipe and Risto Miikkulainen, editors, *Advances in Self-Organizing Maps*, pages 19–27, Berlin, Heidelberg, 2009. Springer. ISBN 978-3-642-02397-2. doi: 10.1007/978-3-642-02397-2_3.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*, November 2016. URL https://openreview.net/forum?id=SyK00v5xx.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, June 2017. ISSN 2307-387X. doi: 10.1162/tacl_a_00051. URL https://doi.org/10.1162/tacl_a_00051.

T. Caliński and J Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, January 1974. ISSN 0090-3272. doi: 10.1080/03610927408827101. URL https://www.tandfonline.com/doi/abs/10.1080/03610927408827101.

Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-37456-2. doi: 10.1007/978-3-642-37456-2_14.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, September 2019. URL https://openreview.net/forum?id=r1xMH1BtvB.

Alexis Conneau and Douwe Kiela. SentEval: An evaluation toolkit for universal sentence representations. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1269.

David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979. ISSN 1939-3539. doi: 10.1109/TPAMI.1979.4766909. URL https://ieeexplore.ieee.org/abstract/document/4766909.

Jeroen de Knijff, Flavius Frasincar, and Frederik Hogenboom. Domain taxonomy learning from text: The subsumption method versus hierarchical clustering. *Data & Knowledge Engineering*, 83:54–69, January 2013. ISSN 0169-023X. doi: 10.1016/j.datak.2012.10.002. URL https://www.sciencedirect.com/science/article/pii/S0169023X12000973.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Matthew Freestone and Shubhra Kanti Karmaker Santu. Word embeddings revisited: Do LLMs offer something new?, March 2024. URL http://arxiv.org/abs/2402.11094.

Mofiz Mojib Haider, Md. Arman Hossin, Hasibur Rashid Mahi, and Hossain Arif. Automatic text summarization using Gensim Word2Vec and K-Means clustering algorithm. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 283–286, June 2020. doi: 10.1109/TENSYMP50017.2020.9230670. URL https://ieeexplore.ieee.org/abstract/document/9230670.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, September 2022. URL https://openreview.net/forum?id=sE7-XhLxHA.

Teuvo Kohonen. *Self-organizing maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 2001. ISBN 978-3-540-67921-9 978-3-642-56927-2. doi: 10.1007/978-3-642-56927-2. URL http://link.springer.com/10.1007/978-3-642-56927-2.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, December 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/c32319f4868da7613d78af9993100e42-Abstract-Conference.html.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, September 2019. URL https://openreview.net/forum?id=H1eA7AEtvS.

Xiaoxiao Li, Jiyang Chen, and Osmar Zaiane. Text document topical recursive clustering and automatic labeling of a hierarchy of document clusters. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 197–208, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-37456-2. doi: 10.1007/978-3-642-37456-2_17.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach, July 2019. URL http://arxiv.org/abs/1907.11692.

J MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN 978-0-521-86571-5. URL https://nlp.stanford.edu/IR-book/.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction, September 2020. URL http://arxiv.org/abs/1802.03426.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, September 2013. URL http://arxiv.org/abs/1301.3781.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.148. URL https://aclanthology.org/2023.eacl-main.148.

OpenAI. GPT-4 technical report. Technical report, OpenAI, March 2024. URL https://arxiv.org/abs/2303.08774.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. ISSN 1533-7928. URL http://jmlr.org/papers/v12/pedregosa11a.html.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL http://aclweb.org/anthology/D14-1162.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. Evaluation of sentence embeddings in downstream and linguistic probing tasks, June 2018. URL http://arxiv.org/abs/1806.06259.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

David M. W. Powers. Applications and explanations of Zipf's law. In *Proceedings of the joint conferences on new methods in language processing and computational natural language learning*, pages 151–160. Association for Computational Linguistics, 1998.

Ethan M. Rasiel. *The McKinsey way*. McGraw-Hill, 1999. ISBN 978-0-07-179035-2.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks, August 2019. URL http://arxiv.org/abs/1908.10084.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. Classification and clustering of arguments with contextualized word embeddings. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 567–578, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1054. URL https://aclanthology.org/P19-1054.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, November 1987. ISSN 0377-0427. doi: 10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Rohan Saha. Influence of various text embeddings on clustering performance in NLP, May 2023. URL http://arxiv.org/abs/2305.03144.

J. Sheela, S. Divya Meena, Thoom Purna Chander Rao, B. Meghna Chetty, Swetha Krishna Gajula, Reethwik Ramdev Reddy, Mannam Sravan Kumar, and Adabala Lakshmi Nagesh. Text clustering using fuzzy rule and lexical term. *AIP Conference Proceedings*, 2869(1):050013, October 2023. ISSN 0094-243X. doi: 10.1063/5.0168206. URL https://doi.org/10.1063/5.0168206.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c3a690be93aa602ee2dc0ccab5b7b67e-Abstract.html.

Anca Tache, Gaman Mihaela, and Radu Tudor Ionescu. Clustering word embeddings with self-organizing maps. Application on LaRoSeDa - a large Romanian sentiment data set. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 949–956, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.81. URL https://aclanthology.org/2021.eacl-main.81.

Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to data mining*. Pearson, 2 edition, 2018. ISBN 978-0-13-312890-1.

P. D. Uma and P. Santhi Thilagam. Leveraging structural and semantic measures for JSON document clustering. *Journal of Universal Computer Science*, 29(3): 222–241, 2023. doi: 10.3897/jucs.86563.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural

language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Hongwei Wang, Hongming Zhang, and Dong Yu. On the dimensionality of sentence embeddings. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10344–10354, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.694. URL https://aclanthology.org/2023.findings-emnlp.694.

Kexin Wang, Nils Reimers, and Iryna Gurevych. TSDAE: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 671–688, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.59. URL https://aclanthology.org/2021.findings-emnlp.59.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.168. URL https://aclanthology.org/2022.naacl-main.168.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Tim Weißer, Till Saßmannshausen, Dennis Ohrndorf, Peter Burggräf, and Johannes Wagner. A clustering approach for topic filtering within systematic literature reviews. *MethodsX*, 7:100831, January 2020. ISSN 2215-0161. doi: 10.1016/j.mex.2020.100831. URL https://www.sciencedirect.com/science/article/pii/S2215016120300510.

Spyros Zoupanos, Stratis Kolovos, Athanasios Kanavos, Orestis Papadimitriou, and Manolis Maragoudakis. Efficient comparison of sentence embeddings. In *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*, SETN '22, pages 1–6, New York, NY, USA, syyskuu 9, 2022. Association for Computing

Machinery. ISBN 978-1-4503-9597-7. doi: 10.1145/3549737.3549752. URL https://doi.org/10.1145/3549737.3549752.

A Appendix

A.1 Metric formulations

Silhouette score

Rousseeuw (1987) defines the Silhouette score (SI) as:

$$s(i) = \frac{b_i - a_i}{\max\{a_i, b_i\}} , \tag{A1}$$

where a_i is the average distance from point i to other points in the same cluster and b_i is the average distance from point i to points of the closest neighbouring cluster.

Davies-Bouldin index

Davies and Bouldin (1979) define the Davies-Bouldin index (DBI) as:

$$\bar{R} = \frac{1}{N} \sum_{i=1}^{N} \max_{i \neq j} \left(\frac{S_i + S_j}{M_{ij}} \right) , \tag{A2}$$

where N is the number of clusters, S_i and S_j are dispersions of clusters i and j and M_{ij} is the distance between representation vectors of clusters i and j.

Calinski-Harabasz index

Caliński and Harabasz (1974) define the Calinski-Harabasz index (CH) as:

$$\text{VRC} = \frac{\text{BGSS}/(k-1)}{\text{WGSS}/(n-k)} , \tag{A3}$$

where n is the number of data points, k is the number of clusters, BGSS is the between-group (or cluster) sum of squares, and WCSS is the within-group sum of squares. The sum of squares corresponds to dispersion, i.e., for BGSS, it is the sum of squared Euclidean distances between cluster centroids and the mean of the data.

Average separation

We define the average separation (AvgSep) as the mean distance between the centroids of all pairs of clusters:

$$\text{AvgSep} = \frac{2}{k(k-1)} \sum_{1 \leq i < j \leq k} \|c_i - c_j\| , \tag{A4}$$

where k is the number of clusters, $k(k-1)/2$ is the number of unique cluster pairs, c_i and c_j are cluser centroids and $\|\cdot\|$ is the euclidean distance provided by `pairwise_distances` function from scikit-learn's `metrics` module (Pedregosa et al., 2011). This metric provides an overall estimate of how well-separated the clusters are.

Average compactness

We define the average compactness (AvgComp) as the mean of the average pairwise distances within each cluster:

$$\text{AvgComp} = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{2}{n_i(n_i-1)}\sum_{1\le p<q\le n_i}\|x_p - x_q\|\right), \tag{A5}$$

where k is the number of clusters, n_i is the number of points in cluster i and $n_i(n_i-1)/2$ is the number of unique data point pairs in cluster i, x_p and x_q are points of cluster i and $\|\cdot\|$ is the euclidean distance provided by `pairwise_distances` function from scikit-learn's `metrics` module (Pedregosa et al., 2011). This metric provides an overall estimate of how tightly the points are located within clusters.

A.2 Scatter plots of clusters in baseline run of MiniLM

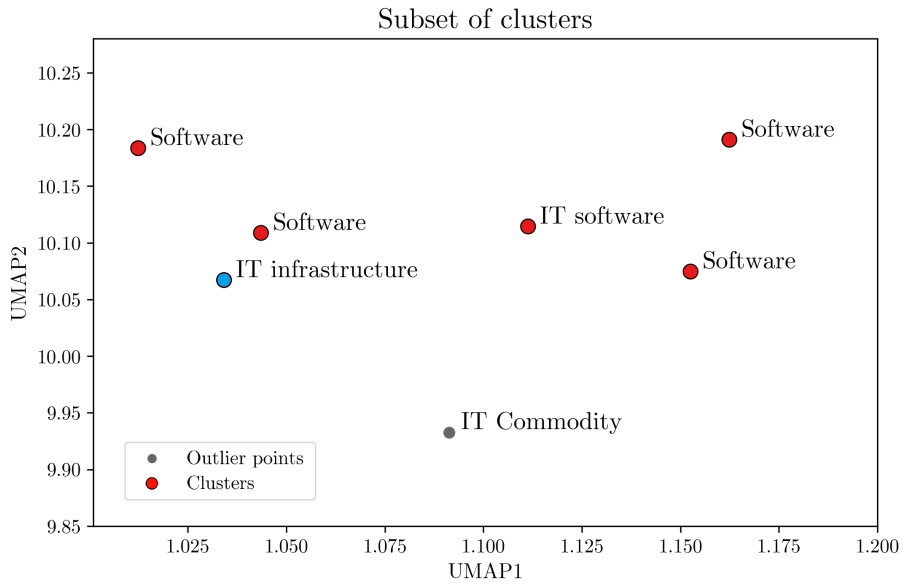


Figure A1: Example 3 of clusters in MiniLM baseline run.

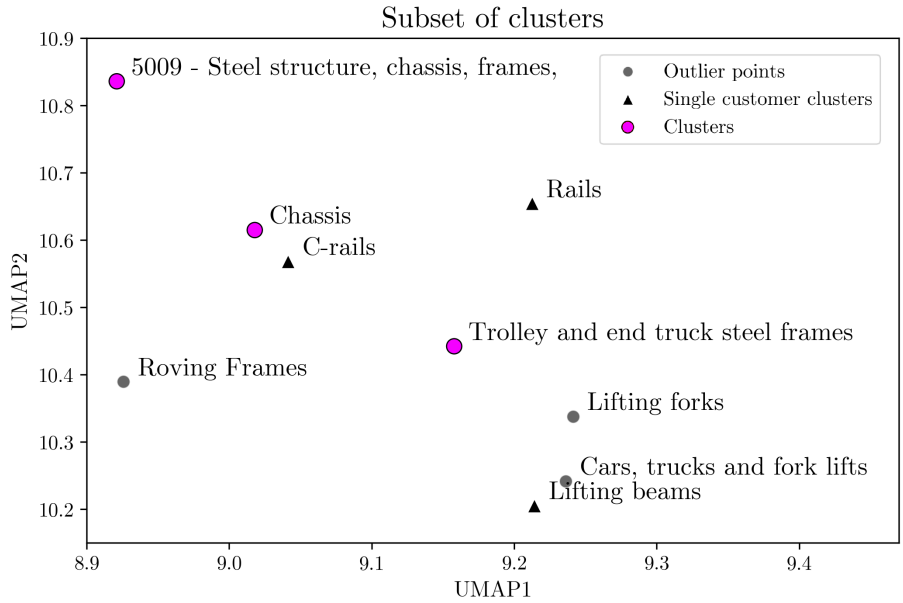


Figure A2: Example 4 of clusters in MiniLM baseline run.

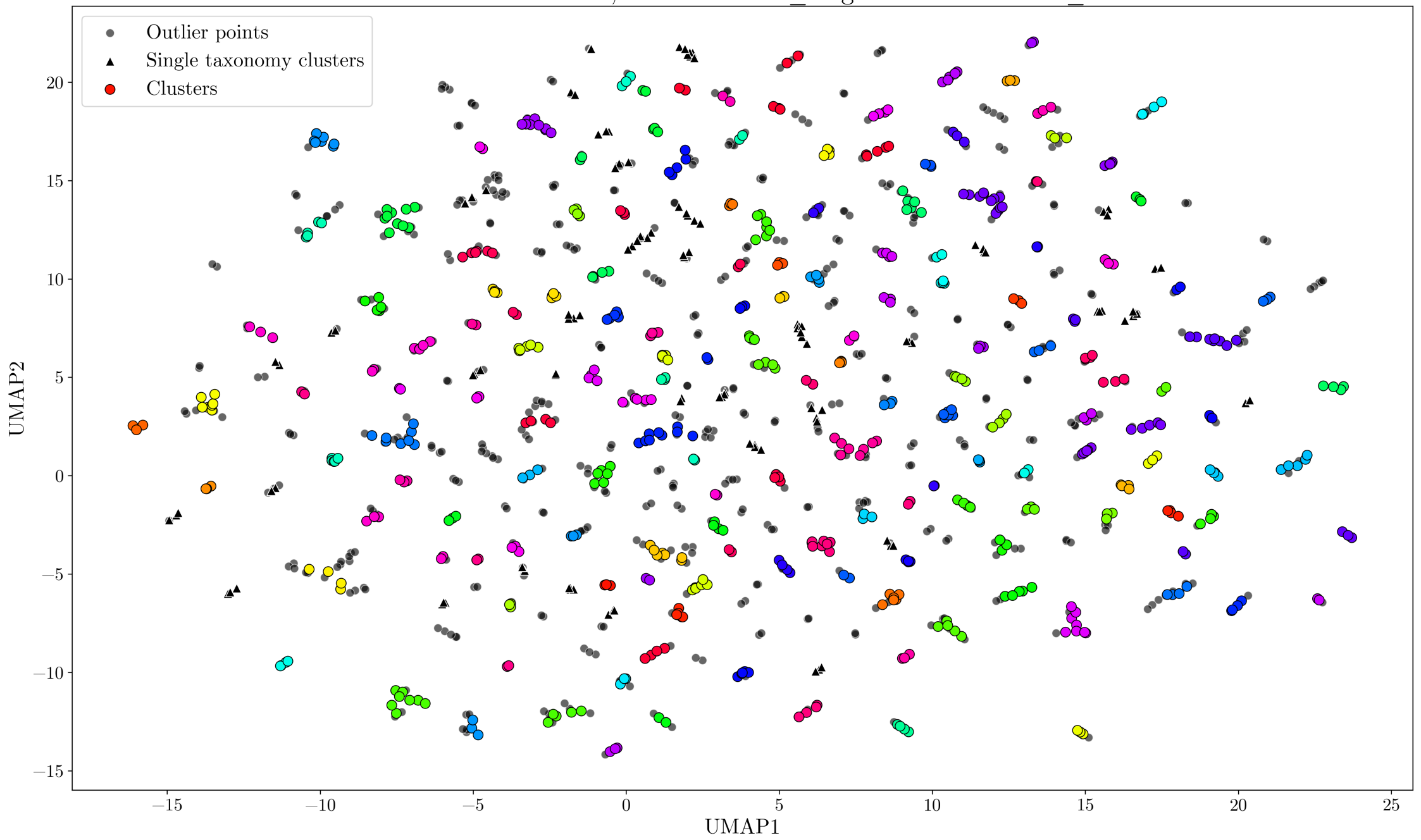


Figure A3: Clusters and outlier visualized with UMAP. Local perspective.

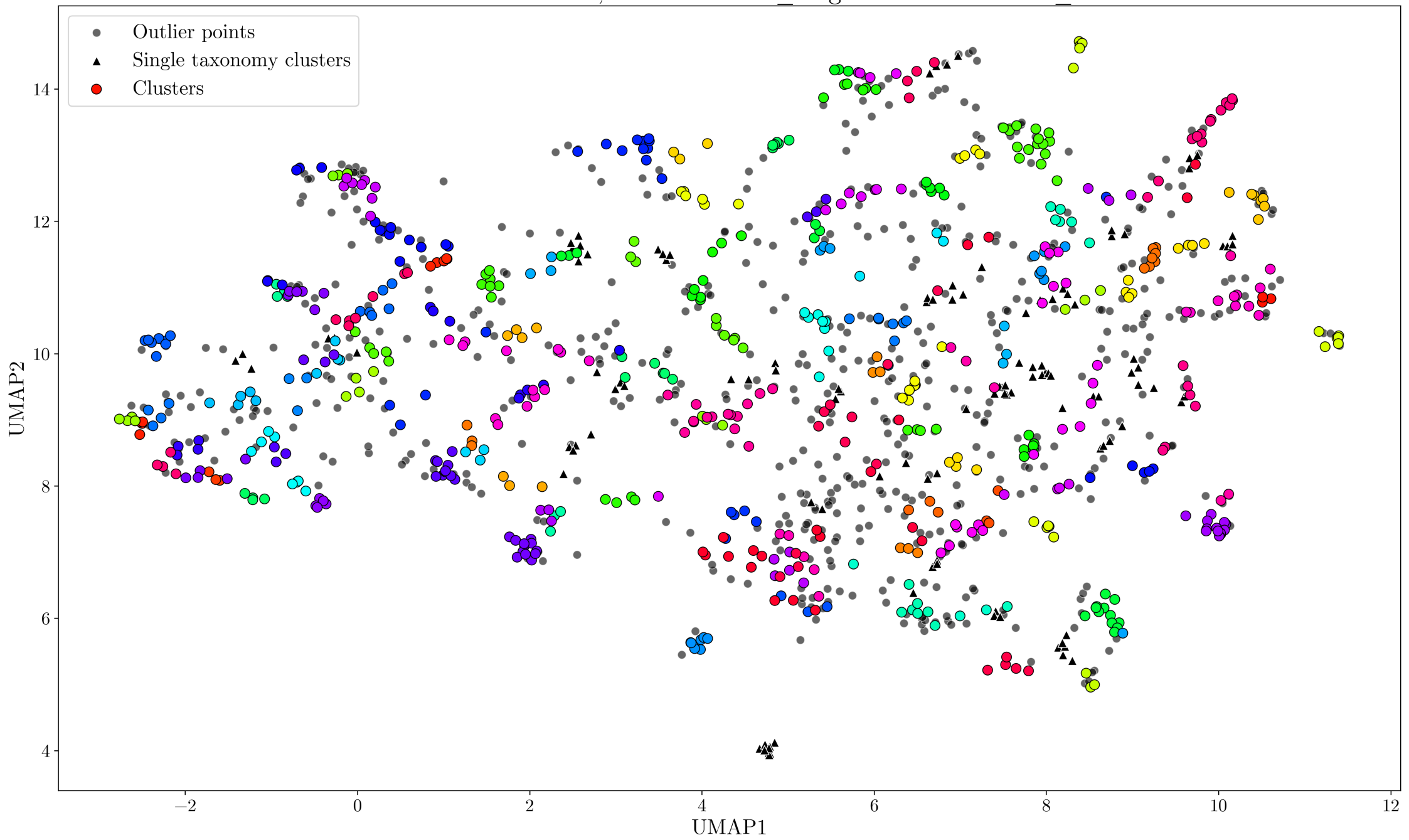


Figure A4: Clusters and outlier visualized with UMAP. Local and global perspective.

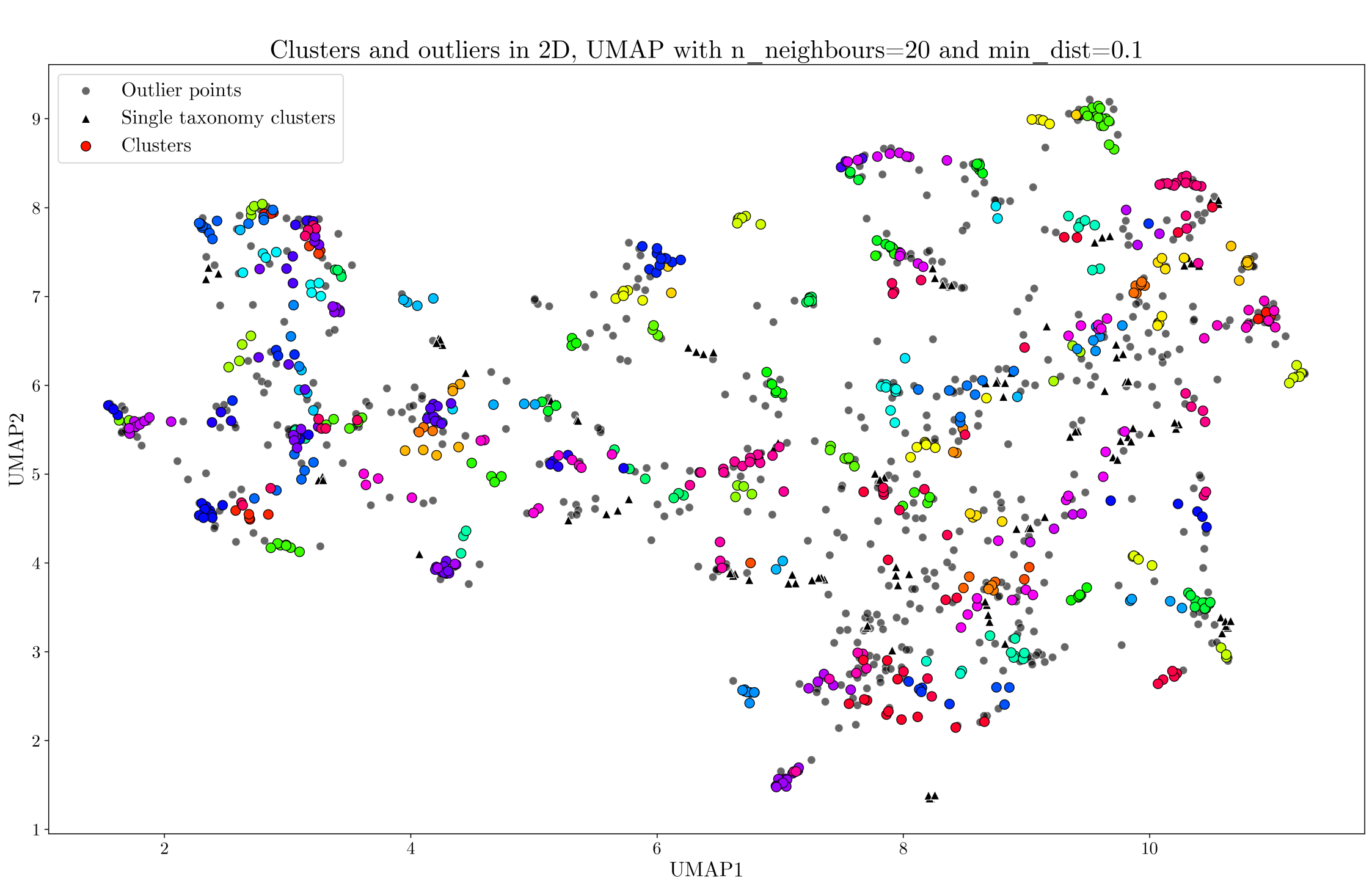


Figure A5: Clusters and outlier visualized with UMAP. Global perspective.