# WebPatterns:
# An Interactive Dashboard for Website Text Clustering

Vikas Gajanan Kale
*Data and Knowledge Engineering*
*Otto von Guericke University*
*Magdeburg, Germany*
*Email: vikas.kale@st.ovgu.de*

***Abstract*—Modern company websites are a storehouse of data like business models, competitive position, products and services, geographic scope, and even unspoken future intentions, yet this information is rarely mined at scale. In this study, we explore how text clustering can surface those latent patterns, industry trends, and competitive insights and make them accessible to analysts without a data science background. We start with a critical review of the two major algorithms of unsupervised clustering: centroid-based clustering, valued for speed and interpretability, and density-based clustering, regarded for the ability to find loosely shaped clusters and noisy points. With a corpus of 5702 German websites belonging to multiple sectors like finance and insurance, hotel and restaurant, transport and storage, and information and technology, we compare both categories of algorithms under three feature generation strategies - TF-IDF, Word2Vec, and Sentence-BERT with a combination of dimensionality reduction and evaluate them based on the silhouette score, and Davies-Bouldin Index. With an interactive dashboard, we use clean and preprocessed data from company caches and create features using the techniques mentioned above. Users can choose K-means or DBSCAN clustering, adjust parameters, and get silhouette and Davies–Bouldin scores for model evaluation. This end-to-end pipeline allows analysts to experiment and compare cluster strategies, turning raw web content into meaningful groupings and guiding data-driven decision-making for analysis and market research. Our results show that TF-IDF with DBSCAN clustering on UMAP-reduced features achieves the best balance of cluster separation and interpretability, outperforming the other feature generation and clustering techniques implemented on the business data.***

## 1. Introduction

The ever-growing amount of unstructured textual information on company websites, such as self-descriptions, services, and strategic hints offer a rich yet underutilized source of insights into business ecosystems. However, these patterns are difficult to mine at a large scale, given language variance, noisy content, and the absence of a standard structure. By clustering textual data, hidden structures, and industry synergies can be revealed, but for different algorithms and feature representation methods, the clustering results may differ according to the data and targets.

This work aims to bridge that gap by applying both centroid-based (K-Means) and density-based (DBSCAN) clustering algorithms to German company website data, and evaluating their performance across three feature representations:

- TF-IDF: a traditional sparse lexical representation
- Sentence BERT (SBERT): Providing contextual sentence-level embeddings.
- Word2Vec: Capturing semantic context

We conducted this study on a dataset of 5,702 German websites from multiple sectors such as finance, hospitality, logistics, and IT. For each method, we evaluated clustering quality using the Silhouette Score and Davies–Bouldin Index, across K - Means and DBSCAN algorithms.

Our contributions include:

- A comparative study of clustering strategies (Algorithm × Representation) on German corporate website text.
- Demonstration of the trade-offs between interpretability (TF-IDF + K-Means) and semantic grouping (SBERT + DBSCAN).
- Development of a backend–frontend pipeline for real-time parameter exploration and visualization.

## 2. Related Work

The clustering of unstructured textual information has been researched extensively, particularly for document analysis, topic modeling, and semantic segmentation. One of the seminal methods of text

representation is the application of TF-IDF(Term Frequency–Inverse Document Frequency), which maps raw text into sparse, high-dimensional term-weighted matrices defined across documents. Though effective for the capture of word relevance, TF-IDF lacks in semantic interpretation and performs poorly with sparse or high-dimensionality of the data cluster [10].

To overcome these shortcomings, word embeddings like FastText have emerged with their rich vector representations that maintain semantic relationships between words based on subword modeling and context [3]. Recent advances in text embeddings have made clustering and classification tasks much more effective. When these embeddings are averaged across documents, they create compact feature vectors that work well for various applications [12].

The landscape changed significantly with contextual language models like BERT (Bidirectional Encoder Representations from Transformers), which researchers adapted for semantic clustering using sentence embeddings. Sentence-BERT, or SBERT [11], was a game-changer because it could generate fixed-size sentence vectors through a siamese network architecture. This made similarity computation much faster and more efficient. SBERT has proven particularly effective for semantic search and unsupervised clustering, especially when working with multilingual content or specialized domains.

When it comes to clustering algorithms, K-Means continues to be popular because it is computationally efficient and easy to implement. But it has limitations, it assumes clusters are spherical and requires to specify the number of clusters upfront, which doesn't work well with complex text distributions [4][1]. Density-based methods like DBSCAN (Density-Based Spatial Clustering of Applications with Noise) offer more flexibility by identifying clusters of any shape and automatically spotting outliers. This makes them particularly useful for reduced-dimensional embeddings. HDBSCAN(Hierarchical Density-Based Spatial Clustering of Applications with Noise) takes this even further by improving cluster stability and handling varying densities while being less sensitive to parameter choices [6].

A recent survey draws attention to co-clustering methods, which cluster documents and features at the same time [2]. According to the authors, these techniques, such as spectral and matrix factorization approaches, can make it easier to interpret results and reduce the complexity of high-dimensional, sparse text data. The survey finds that co-clustering tends to reveal patterns in the data that are not only more compact but also easier to interpret. Even so, these methods have yet to see widespread adoption. In practice, most existing work has been restricted to English-language data or narrowly defined topics, so there is still considerable room to apply co-clustering to larger, more diverse, and industry-relevant datasets.

Researchers have tested various combinations of embeddings and clustering approaches, [8] tried FastText with OPTICS (Ordering Points To Identify the Clustering Structure) for topic modeling, while [12] found that SBERT combined with K-Means gives competitive results across different text domains, but the final cluster quality depends heavily on the dataset. However, most of this research has focused on English datasets or specialized areas like social media and academic abstracts. What is missing is comprehensive research on real-world multilingual corporate content, particularly in the German business environment.

Our study addresses this gap by testing multiple feature generation approaches, namely TF-IDF, FastText, and SBERT alongside K-Means and DBSCAN clustering using over 5,700 German company websites. What sets our work apart is the systematic comparison under consistent conditions, plus we provide both quantitative results and practical deployment tools through an analytics pipeline.

## 3. Implementation

This section presents the full implementation pipeline for clustering company websites. It includes the system architecture, data cleaning process, text-to-vector transformation (feature engineering), clustering configurations, cluster visualization, evaluation metrics, and results.

### 3.1. System Overview

We built the system as a modular pipeline that can cluster text data from German company websites. It handles the entire process from start to finish which includes taking raw text files (includes the actual website content), extracting features, performing clustering, and evaluating results. The system also provides programmatic access through a REST API, and we designed the architecture to easily connect to an interactive frontend for visualization and data exploration. We used FastAPI, a lightweight framework to implement the API connectivity and ReactJS, an interactive JavaScript framework for cluster visualizations and interactivity.

The system starts by cleaning and lemmatizing raw website data and metadata, then combines the pre-processed data into a single CSV file. From there, we

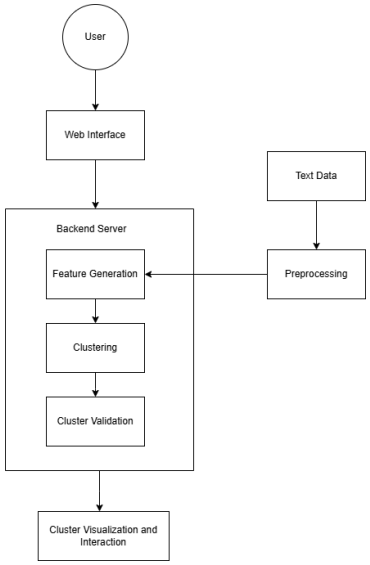



Figure 1: End-to-end system architecture

convert the text into numerical feature vectors using one of three methods: TF-IDF, SBERT, or Word2Vec. These vectors then get clustered using either K-Means or DBSCAN. We store all the results which include the features, cluster assignments, and performance metrics, so users can visualize and compare different approaches. A FastAPI backend lets users trigger the feature generation and clustering processes on demand.

## 3.2. Data Preparation and Preprocessing

The dataset includes 5,702 German company directories, and each company one has multiple .txt files, .html files, and a company_data.json file. Each .txt and .html file represents a webpage from the website. A rule-based filter uses regex patterns to spot and remove noisy, CAPTCHA-like pages. Any webpage under 1,500 characters that mentions captcha texts like "verify", "cloudflare", or "not a robot" gets removed. If a CAPTCHA was present on just part of a page, we deleted only that specific section. We also filtered out common German legal terms and boilerplate text like "Datenschutz," "Impressum," "AGB," cookie consent messages, and standard disclaimers. This way, the content we kept was focused only on business-relevant information.

To clean the text, the German spaCy model (de_core_news_sm) [5] is used, but NER(Named Entity Recognition) and dependency parsing are turned off to make the pre-processing step fast. The text is lemmatized, stopwords are removed (including business suffixes like "GmbH"), and dates and URLs are stripped out. Each document is then cut off at 10,000 characters.

After preprocessing, if any text has fewer than 30 tokens, it was excluded. We save the clean text in a CSV file and keep track of any skipped files and filtering details in a separate Excel file for further analysis. A summary of data retention is provided in Table 1.

| Step / Filter | Count | % of Comp. |
|---|---|---|
| Total comp. (raw) | 5,702 | 100% |
| junk/CAPTCHA | 9 | 0.16% |
| short text | 309 | 5.4% |
| **comp. in cleaned CSV** | **5,384** | **94.4%** |

Table 1: Summary of preprocessing and filtering steps for German company website data. The majority of companies were retained after exclusion of junk/CAPTCHA, and short text entries.

## 3.3. Feature Engineering

We used three different text representation methods to understand how they affect the clustering results: TF-IDF, SBERT, and Word2Vec(FastText). Each has its own balance of interpretability, semantic depth, and scalability.

**3.3.1. TF-IDF.** TF-IDF represents the classic approach to text vectorization which focuses on word frequency to transform documents into numerical representations. This method calculates term importance by considering term frequency within individual documents and rarity across the entire corpus, making it a commonly used method for text analysis tasks.

The TF-IDF value for term $t$ in document $d$ within corpus $D$ is:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{1}$$

where $\text{TF}(t, d) = \text{freq}(t, d)/|d|$ and $\text{IDF}(t, D) = \log(|D|/|\{d : t \in d\}|)$.

This formula emphasizes terms frequent within documents but rare across the corpus, identifying distinct vocabulary for specific documents or topics.

We tuned the TF-IDF representation with three main hyper-parameters. First, the n-gram range was tested at (1, 1), (1, 2), and (1, 3). This parameter defines the length of continuous word sequences treated as features: (1, 1) restricts the model to unigrams, (1, 2) adds bigrams, and (1, 3) further includes trigrams. While larger ranges capture richer context, they also increase computational cost. Second, the maximum vocabulary size (max_features) was fixed at 1,000, 3,000, and 5,000, limiting dimensionality by retaining only the most frequent terms. Third, the min_df threshold excluded very rare tokens, removing probable

typographical errors and overly specific vocabulary.

We chose TF-IDF as our baseline because it is both conceptually transparent and computationally light. By tying each term's weight directly to its frequency within the corpus and the individual document, the method clarifies why a feature matters, streamlining interpretation and troubleshooting. This clarity makes TF-IDF a sensible starting point for generating text features and for gauging the added value of more intricate models.

**3.3.2. SBERT (Sentence-BERT).** Sentence-BERT (SBERT) adapts transformer models to produce embeddings at the sentence level, emphasising context over mere word counts. By modelling each token against its surrounding words, SBERT preserves semantic relations and subtle nuances, yielding a more faithful representation of textual meaning and a stronger basis for measuring document similarity than frequency-based vectors.

For this study we used the multilingual MiniLM-L12-v2 checkpoint from the SentenceTransformers library. The model outputs 384-dimensional dense vectors that preserve cross-lingual semantics while remaining computationally compact. MiniLM's pared-down design retains most of the representational power of larger Transformer variants at a fraction of the cost, providing an efficient yet expressive backbone for downstream analyses.

To counteract the curse of dimensionality, we reduced the 384-dimensional SBERT vectors to lower-dimensional representations, systematically varying the output dimensions to study their effect on cluster quality with Uniform Manifold Approximation and Projection (UMAP) [7]. This non-linear reducer preserves both neighbourhood structure and broader geometry while markedly reducing computational load. UMAP was chosen because it retains meaningful relationships in low-dimensional space yet remains fast to compute. The compressed embeddings were then clustered with DBSCAN. Combining UMAP with a density-based algorithm alleviates common high-dimensional pitfalls, such as distance concentration and weakened similarity measures, so clusters capture genuine semantic affinity.

The transformer embeddings themselves recognise semantic overlap even when different vocabularies are used, and because the model is pre-trained, no large domain corpus is required. The reduction-and-clustering workflow therefore delivers tractable computation without sacrificing the information needed for accurate grouping.

**3.3.3. Word2Vec.** Word2Vec is a neural embedding method that maps words into a continuous vector space, capturing distributed semantic regularities. Implementation used FastText [3], an extension that enriches embeddings with subword information, allowing it to robustly handle out-of-vocabulary items and morphologically rich languages by representing character-level n-grams.

For our experiments, we rely on the publicly available German FastText model (cc.de.300.bin) trained on Common Crawl and Wikipedia. Its broad vocabulary ensures dependable coverage of the corpus under study. Each token is mapped to a 300-dimensional dense vector whose coordinates encode both semantic proximity and syntactic function, placing related terms in close neighborhood. Because FastText builds each word vector by summing the embeddings of character n-grams(typically three to six letters long), it can generate plausible representations for unseen forms. This property is especially valuable in German, where compounding and rich inflection create many rare tokens.

To obtain document-level embeddings, we averaged the FastText vectors of every token in a document, producing a single 300-dimensional vector. Although this simple aggregation discards word order, it yields a stable representation that resists variation in document length and preserves the text's overall meaning

FastText suits German data for three main reasons. First, its subword modelling captures the language's rich morphology. Second, the pre-trained weights embed knowledge drawn from large German corpora. Third, mean pooling supplies uniform document vectors regardless of size. The resulting 300-dimensional space strikes a practical balance between descriptive power and computational cost for subsequent clustering.

## 3.4. Clustering Methods

We implemented two clustering methods to evaluate the effectiveness of different text representation approaches across various clustering paradigms.

**3.4.1. K-Means.** K-Means is a centroid-based partitioning algorithm that clusters observations by minimising the within-cluster sum of squares (WCSS). The procedure alternates between assigning each point to the nearest centroid and recomputing the centroid as the mean of its assigned members until the assignments stabilise, yielding k discrete clusters at a local optimum.

**Algorithm Configuration and Parameter Selection**

**Cluster Range:** We evaluated K-Means with cluster counts from k = 4 to k = 20 to balance coarse

and fine partitions while avoiding fragmentation. The lower bound (k = 4) secures adequate intracluster cohesion and intercluster separation; the upper bound (k = 20) prevents over-segmentation that could create singletons or very small, hard-to-interpret clusters.

**Feature Space Application:** We ran K-Means directly on the features produced by TF-IDF, SBERT, and FastText, skipping any preliminary reduction. Retaining the full dimensionality keeps every feature intact and lets the algorithm use each model's detail, giving a fair basis for comparing their output while avoiding the information loss a reducer might introduce.

We did not reduce the number of features before running KMeans. This follows what is recommended in standard references like [4] and [1], which advise applying KMeans directly to the original feature vectors, whether these are TF-IDF, dense embeddings, or word vectors. The reason is that KMeans centroids and cluster assignments are meaningful only in the original feature space. Commonly used toolkits such as scikit-learn[9] also use KMeans directly on high-dimensional TF-IDF data, reserving dimensionality reduction for density-based clustering or visualization tasks. Lowering the number of dimensions beforehand can change the structure of the data and make the resulting clusters less reliable or harder to interpret[4][1]. With over 5,000 companies and fewer than 10,000 features, our dataset is easily manageable for these algorithms. For both consistency with prior work and reliable results, we always used the unreduced feature matrices when running KMeans.

**Algorithmic Characteristics and Assumptions:**
**Computational Efficiency:** K-Means grows roughly in proportion to the size of the dataset, so it remains practical even for large corpus of documents. Its loop of reassignment and centroid update usually settles after a handful of iterations, which gives us a reliable yardstick when we later consider heavier clustering methods.

**Geometric Assumptions:** K-Means assumes that each cluster is spherical and broadly uniform in size and density. It assigns points to the cluster whose centroid lies closest in Euclidean space. This simplification can fail when clusters are elongated, vary markedly in density, or contain nested subgroups, situations that frequently arise in high-dimensional text embeddings.

**Limitations and Considerations:** The geometric assumptions weaken the suitability of K-Means for text embeddings, where semantic regions often trace irregular paths in high dimensional space. The method is also sensitive to the initial placement of centroids and obliges the analyst to set the number of clusters (k) in advance. Reliable results therefore require testing multiple k values and checking the stability of each solution before choosing a final partition.

**3.4.2. DBSCAN.** DBSCAN forms clusters by looking for pockets of high point density separated by sparse areas; observations that lie outside any dense pocket are treated as noise and set aside. Because membership depends on local density rather than a pre-defined shape, DBSCAN can uncover clusters that curve, vary in thickness, or differ in size and patterns that centroid methods often overlook.

**Algorithm Configuration and Parameter Space.**

**Feature Space Application:** We applied DBSCAN only to all the three embeddings after reducing their dimensionality with UMAP. Using these compact, semantically rich vectors preserves the neighbourhood relations that density estimation needs and keeps computation manageable. The core parameters eps and min_samples were tuned empirically to match the density profile of the embedded corpus. UMAP produced a lower dimensional space, which lessens distance concentration and improves the reliability of density calculations in high dimensions.

**Parameter Optimization:** We examined DBSCAN's behaviour across its two key hyperparameters. For the neighbourhood radius ε we tested five values—0.2, 0.4, 0.6, 0.8, and 1.0. A small radius yields compact, restrictive clusters, whereas a larger radius permits broader group boundaries. For min_samples we tried 5, 10, and 15 points, the minimum needed to mark a core region. Raising this threshold increases cluster stability and filters noise but risks overlooking small yet meaningful patterns; lowering it has the opposite effect.

**Parameter Interaction:** Crossing the five ε settings with the three min_samples levels produced a 15-cell grid that gave a thorough view of the parameter space. In practice, low ε combined with high min_samples generated compact, high-confidence clusters, while high ε paired with low min_samples led to more inclusive, loosely defined groupings.

**Algorithmic Advantages for Semantic Embeddings**

**Shape Flexibility:** DBSCAN organises points by local density rather than by distance to a fixed centre, so it can trace clusters that bend, branch, or vary in thickness. This feature suits sentence- or document-level embeddings, where related texts often occupy curved or elongated regions of the space.

**Noise Detection:** Points that lack enough neighbors within the chosen radius are marked as noise. In a text corpus these outliers typically correspond to documents

on idiosyncratic topics or with too little semantic content to join a broader theme, giving analysts a clear set of items for separate review.

**Density Adaptivity:** Because DBSCAN decides cluster membership by counting nearby points, it can accommodate groups that differ in both size and density. This trait suits text corpora, where some topics attract many near-duplicate documents and others are represented by only a handful of more varied texts.

**Integration with Dimensionality Reduction:** Our pipeline first converts documents to neural embeddings, then projects them with UMAP, and finally clusters them using DBSCAN. UMAP keeps each document's close neighbours together while trimming dimensions, which allows DBSCAN to detect coherent semantic groups without our having to set the number of clusters in advance.

### 3.5. Evaluation Metrics

To evaluate the quality of the resulting clusters, we employed internal validation metrics that do not depend on reference labels.

**Silhouette Score:** This metric served as our primary internal metric for judging cluster quality. For each data point, it compares the mean distance to other members of the same cluster with the mean distance to the closest neighboring cluster. Scores range from negative 1 to positive 1 and the values near 1 denote compact, well-separated clusters, whereas negative scores point to likely misclassification. Because it unifies intra-cluster cohesion and inter-cluster separation into a single statistic, the silhouette score offers an intuitive summary of overall clustering performance.

**Davies-Bouldin Index (DBI):** This score weighs how tightly each cluster is packed against how far it sits from its most similar neighbour. A lower value signals better structure because clusters are both compact and well separated. By penalising groups that are either too diffuse or too close together, the index offers a clear basis for comparing alternative clustering runs.

**Noise Ratio:** For DBSCAN we also track the share of points assigned the noise label (minus 1). A large ratio hints that $\varepsilon$ or min_samples is set too strictly and is breaking apart legitimate groups, whereas an extremely small ratio suggests parameters that over-merge distinct topics. Monitoring this figure helps fine-tune the density thresholds for reliable clustering.

### 3.6. Cluster Visualization

We developed an interactive visual module in the web-based dashboard to give analysts an intuitive view of the clustering results. The component displays the clustered company websites as two-dimensional scatter plots produced with Uniform Manifold Approximation and Projection (UMAP), offering a concise picture of how firms group according to textual similarity.

Each point in the plot represents one website and is coloured by its assigned cluster. When the clusters come from K-Means, cohesive input features yield compact, well-separated clouds. DBSCAN clusters, by contrast, may curve or branch, so the scatter plot is essential for diagnosing parameter choices and overall model behaviour.

To address the high dimensionality of the feature vectors (300 dimensions or more), we first project them to two dimensions with UMAP before plotting. This non-linear reducer maintains local neighbourhoods and the broader geometry of the data more faithfully than linear methods such as principal component analysis, an advantage in complex semantic spaces.

The module supports several interactive actions:

- Hovering over a point reveals key metadata, including domain, WZ code, and stated business purpose ("Zweck").
- A separate summary pane lists, for each cluster, its size and the most frequent WZ code.
- A configuration switch lets users compare different feature-extraction and clustering settings without rerunning the full processing pipeline.

This visual layer links numerical cluster metrics to domain insights, helping users detect industry structure, niche groupings, and potential outliers. Automatic cluster labels based on dominant WZ codes further support interpretation. Because UMAP relies on random initialization, successive runs can differ in layout. Nonetheless, the relative positions of clusters remain stable, and future versions could include a deterministic seed or multiple projections to confirm robustness.

## 4. Results and Analysis

### 4.1. K-Means Clustering Results

We ran K-Means on each feature space namely TF-IDF, SBERT, and Word2Vec, using k values from 4 to 20 and judged the outcome with the silhouette statistic and the Davies-Bouldin index.

**TF-IDF Performance:** For TF-IDF, the highest silhouette value, 0.114, appeared at k = 20 (ngram =

(1,1), max_features = 1000, min_df = 2), hinting at only a faint clustering signal. The lowest Davies–Bouldin score, 2.79, surfaced at k = 9 (ngram = (1,2), max_features = 3000, min_df = 3), again pointing to noticeable overlap between groups. In short, sparse term-frequency vectors gave K-Means little structure to exploit (see Fig. 2).

**SBERT Performance:** With SBERT embeddings, the best silhouette was lower than TF-IDF, 0.067 at k = 4, and the minimum Davies–Bouldin value reached 2.82 at k = 17. Across all tested k, scores remained weak, confirming that the dense, high-dimensional SBERT space resists partitioning by this centroid method (see Fig. 3).

**Word2Vec Performance:** By contrast, Word2Vec fared well. A silhouette of 0.372 at k = 4 and a Davies–Bouldin score of 1.38 at k = 4 suggest compact, well-separated clusters. Among the three representations, Word2Vec therefore offered the most suitable terrain for K-Means (see Fig. 4).

### 4.2. DBSCAN Clustering Results

We applied DBSCAN to the UMAP-reduced TF-IDF, SBERT and Word2Vec embeddings and, after a pilot check confirmed feasibility. An exhaustive grid search varied ε from 0.2 to 1.0, min_samples from 5 to 15 and dimensions ranging from 2 to 100 (2, 5, 10, 50, 100).

**TF-IDF with DBSCAN:** TF-IDF features consistently yielded the strongest DBSCAN clustering results. Silhouette scores ranged from 0.469 to 0.548 across UMAP dimensions, with the best result (0.548) achieved at UMAP dim = 100, $\varepsilon = 1.0$, min_samples = 10. Noise ratios were generally low ( 0.09), indicating most points were successfully clustered. The lowest Davies–Bouldin Index (DBI) was 0.25 at dim = 10. The number of clusters varied substantially, reflecting DBSCAN's sensitivity to parameter settings and dimensionality.

**SBERT with DBSCAN:** Performance improved over the K-Means baseline, but SBERT embeddings produced lower silhouette scores than TF-IDF for all tested settings. The highest silhouette score (0.282) was achieved at UMAP dim = 10, $\varepsilon = 1.0$, min_samples = 5. Noise ratios were near zero at these best settings, and DBI reached a minimum of 0.31 at the same parameters, suggesting that DBSCAN is able to form reasonably tight semantic clusters from SBERT embeddings after dimension reduction.

**Word2Vec with DBSCAN:** Results were generally consistent, but somewhat weaker than TF-IDF and SBERT. The best silhouette score (0.240) was observed at UMAP dim = 20, $\varepsilon = 1.0$, min_samples = 10, with a DBI of 0.90 and a very low noise ratio. Across all UMAP dimensions, silhouette scores remained below those for TF-IDF, and noise ratios were slightly higher for lower dimensions.

Tables 2 and 3 present the top results for each feature extraction method and UMAP dimension. Overall, the combination of TF-IDF features with DBSCAN and UMAP reduction provided the best balance of cluster separation and density.

### 4.3. Comparative Performance Analysis

Comparing the three types of feature representations across both clustering algorithms shows some clear differences in clustering quality (see Table 4). Using TF-IDF features with DBSCAN and UMAP consistently led to the best results, reaching silhouette scores as high as 0.548 and Davies-Bouldin indices as low as 0.25. This suggests that the clusters were both compact and well separated. Word2Vec paired with K-Means performed next best, with a top silhouette score of 0.372 and a DBI of 1.38, still showing good cluster quality, though not quite as strong as the best DBSCAN result.

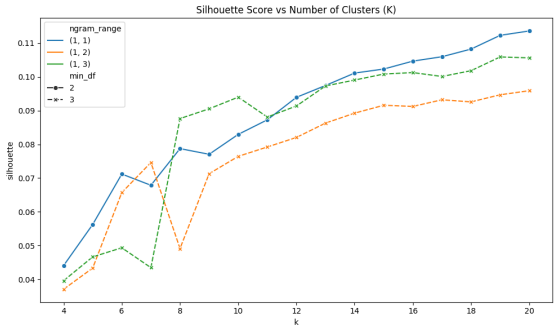
With SBERT embeddings, DBSCAN worked better than K-Means, but both produced weaker overall structure. The best silhouette score for SBERT with DBSCAN was 0.282, and just 0.067 with K-Means. For DBSCAN, the proportion of points marked as noise was generally low, between 0.0 and 0.09 for the best parameter settings. This shows that DBSCAN is good at finding core groups while recognizing a small number of outliers.

K-Means always puts every data point into a cluster, no matter how well it fits. DBSCAN sometimes leaves points out when they do not match any cluster closely. In our results, DBSCAN with UMAP and TF-IDF gave the cleanest separation between groups and made it easier to spot outliers in the German business website data.

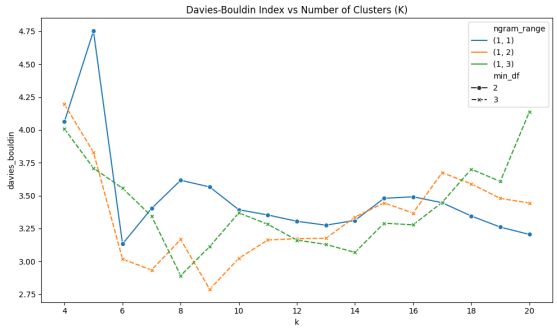
## 5. Challenges & Limitations

Several significant challenges were identified during the implementation and evaluation of our text clustering approach, which provide important insights for future research.

While the FastText model (cc.de.300.bin) gave strong semantic embeddings, generating document-level vectors on CPU-only machines was painfully slow. Even with modest batch sizes, processing took several minutes. Although caching the model helped a bit, this

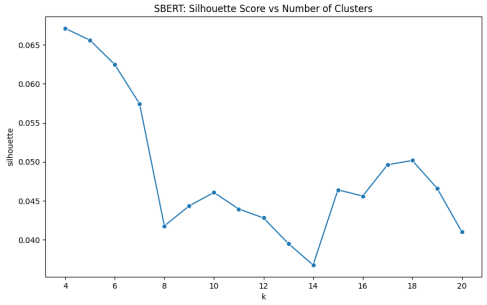



(a) TF-IDF Silhouette Scores

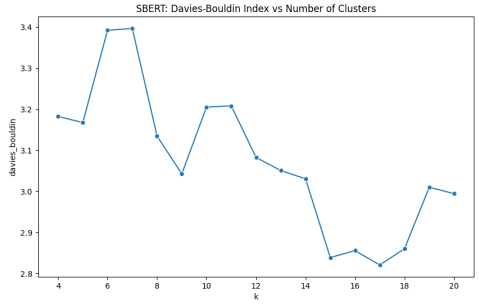



(b) TF-IDF Davies-Bouldin Index

Figure 2: K-Means clustering performance across TF-IDF. Left panels show silhouette scores and right panels show Davies-Bouldin Index values for varying numbers of clusters k. Higher silhouette scores and lower DBI values indicate better clustering quality.

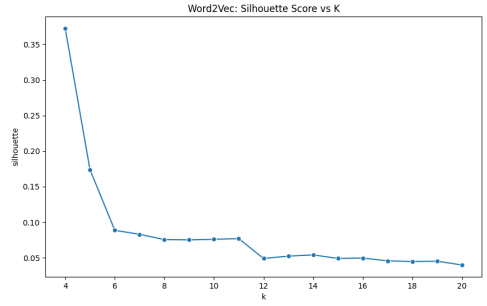



(a) SBERT Silhouette Scores

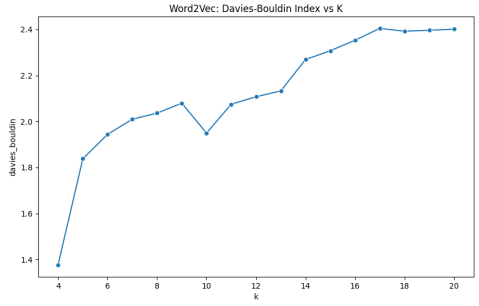



(b) SBERT Davies-Bouldin Index

Figure 3: SBERT K-Means clustering performance showing consistently low silhouette scores and high DBI values across all k values, indicating poor clustering effectiveness with centroid-based methods on dense semantic embeddings.




(a) Word2Vec Silhouette Scores




(b) Word2Vec Davies-Bouldin Index

Figure 4: Word2Vec K-Means clustering performance demonstrating results with silhouette score and DB Index

Table 2: Best DBSCAN clustering results (by silhouette score) for each UMAP dimension using TF-IDF features. For each setting, the best parameter combination is shown.

| n-gram | Max Feat | Min DF | UMAP Dim | eps | min_samples | #Clusters | Silhouette | DBI | Noise Ratio |
|---|---|---|---|---|---|---|---|---|---|
| (1,2) | 5000 | 2 | 2 | 0.2 | 10 | 103 | 0.469 | 1.18 | 0.047 |
| (1,3) | 5000 | 2 | 5 | 0.2 | 5 | 154 | 0.530 | 0.79 | 0.064 |
| (1,2) | 5000 | 2 | 10 | 1.0 | 5 | 61 | 0.536 | 0.25 | 0.000 |
| (1,3) | 5000 | 2 | 20 | 0.2 | 5 | 162 | 0.516 | 0.76 | 0.083 |
| (1,2) | 3000 | 2 | 50 | 0.2 | 5 | 157 | 0.537 | 0.82 | 0.090 |
| (1,2) | 5000 | 2 | 100 | 1.0 | 10 | 58 | 0.548 | 1.19 | 0.010 |

Table 3: Best DBSCAN clustering results (by silhouette score) for each UMAP dimension using S-BERT and Word2Vec features. For each setting, the best parameter combination is shown.

| Method | UMAP Dim | eps | min_samples | #Clusters | Silhouette | DBI | Noise Ratio |
|---|---|---|---|---|---|---|---|
| SBERT | 2 | 0.8 | 15 | 14 | 0.092 | 1.61 | 0.013 |
| | 5 | 1.0 | 15 | 11 | 0.260 | 1.05 | 0.012 |
| | 10 | 1.0 | 5 | 18 | 0.282 | 0.31 | 0.000 |
| | 20 | 1.0 | 15 | 11 | 0.263 | 1.01 | 0.012 |
| | 50 | 1.0 | 15 | 11 | 0.268 | 1.18 | 0.012 |
| | 100 | 1.0 | 15 | 11 | 0.278 | 1.15 | 0.012 |
| Word2Vec | 2 | 0.2 | 15 | 58 | 0.144 | 3.30 | 0.150 |
| | 5 | 1.0 | 15 | 16 | 0.181 | 1.18 | 0.016 |
| | 10 | 0.4 | 15 | 41 | 0.191 | 1.22 | 0.129 |
| | 20 | 1.0 | 10 | 19 | 0.240 | 0.90 | 0.005 |
| | 50 | 1.0 | 10 | 24 | 0.219 | 0.97 | 0.006 |
| | 100 | 0.4 | 15 | 44 | 0.173 | 1.23 | 0.131 |

Table 4: Best clustering performance for each feature representation and algorithm, showing highest silhouette and lowest Davies–Bouldin index (DBI) from search experiments.

| Feature | Clustering | Silhouette | DBI | #Clusters | Noise Ratio | Key Parameters |
|---|---|---|---|---|---|---|
| TF-IDF | KMeans | 0.114 | 2.79 | 20 | 0.000 | k=20, ngram=(1,1), max_feat=1000, min_df=2 |
| TF-IDF | DBSCAN | 0.548 | 1.19 | 58 | 0.010 | eps=1.0, min_samples=10, UMAP_dim=100 |
| SBERT | KMeans | 0.067 | 2.82 | 4 | 0.000 | k=4 |
| SBERT | DBSCAN | 0.282 | 0.31 | 18 | 0.000 | eps=1.0, min_samples=5, UMAP_dim=10 |
| Word2Vec | KMeans | 0.372 | 1.38 | 4 | 0.000 | k=4 |
| Word2Vec | DBSCAN | 0.240 | 0.90 | 19 | 0.005 | eps=1.0, min_samples=10, UMAP_dim=20 |

kind of delay would be a serious bottleneck in real-time or large-scale applications.

DBSCAN didn't handle high-dimensional data well. Since it's a density-based algorithm, DBSCAN struggles in high-dimensional spaces, this is the classic "curse of dimensionality" scenario. The SBERT and Word2Vec embeddings, each over 300 dimensions, posed a problem. To fix this, we used UMAP to shrink the data down into lower dimensions before clustering. While that helped, it also introduced some randomness and may have lost important information.

The scraped data varied a lot, some pages barely had any useful info, others were packed with noise or just boilerplate like contact forms and CAPTCHA text. This inconsistency made it tough to form meaningful clusters. We did a lot of filtering and preprocessing to clean things up, but the lack of rich content in many entries still weakened the final results.

There is an issue with unlabeled data, we have to rely on internal metrics like silhouette scores and DBI. Though helpful, these numbers don't tell whether the clusters actually make sense in a real-world context. Without expert input or predefined categories, it's hard to say if the groupings are truly valuable.

## 6. Conclusion

This study presents a fully working, modular setup for clustering German company websites based on

their text. The system performs preprocessing, feature extraction, clustering, and evaluation into a single, repeatable pipeline. It supports three text representation methods - TF-IDF, SBERT, and Word2Vec and uses KMeans and DBSCAN for clustering. The pipeline handled 5,700 real company websites, which had a wide range in content quality.

Preprocessing was designed to filter out distractions like CAPTCHA pages, boilerplate texts and common German legal terms, giving the system a solid base to work from. Each text representation behaved differently: TF-IDF worked best with KMeans, SBERT gave moderate to weak results with both algorithms, and Word2Vec, while slower, helped form understandable clusters. To fine-tune the settings, we ran a series of visual and numerical evaluations using metrics like silhouette score, Davies-Bouldin index, and noise ratio. DBSCAN, when used with SBERT embeddings reduced by UMAP, stood out for producing well-separated, low-noise clusters.

The backend, built with FastAPI, makes each stage of the process accessible through customizable endpoints, setting the stage for an interactive dashboard on the frontend.

## 7. Future Work

Although the core pipeline is up and running, there are a few directions worth exploring next. An interesting idea is to bring in extra data like WZ codes, business categories and expert evaluation of clusters for better interpretability. These metadata features might uncover patterns that text alone doesn't catch. We could also work on adding weak supervision or semi-supervised learning to make the clusters easier to understand and help users to visualize better. An improvement to the current implementation would be applying consistent dimensionality reduction across all clustering methods. Another step would be turning this pipeline into a flexible visualization platform that supports multiple languages and different modern clustering methods.

## References


[1] C. C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer, 2012.

[2] D. Battaglia, D. Quercia, A. Giacometti, I. Bartolini, and P. Ciaccia. Co-clustering: A survey of methods, metrics, and applications. *Information Fusion*, 96:273–298, 2024.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2 edition, 2009.

[5] M. Honnibal and I. Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. https://spacy.io, 2017.

[6] L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[7] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[8] D. Nabergoj, A. D'Alconzo, D. Valerio, and E. Strumbelj. Topic extraction by clustering word embeddings on short online texts. *Elektrotehniški Vestnik*, 89(1-2):64–72, 2022.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] J. Ramos. Using tf-idf to determine word relevance in document queries. Technical Report DCS-2003-27, Department of Computer Science, Rutgers University, Piscataway, NJ, 2003.

[11] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 3982–3992, 2019.

[12] T. Walkowiak and M. Gniewkowski. Evaluation of vector embedding models in clustering of text documents. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 1145–1153, 2019.