

Cross-lingual Clustering of News Stories using Embeddings-based Community Detection

Nikolaos Nikolaidis^{1*} Kristina Kovacikova^{2*} Ion Androutsopoulos^{1,3}

¹Athens University of Economics and Business, Greece

²European Commission, Joint Research Centre, Italy

³Archimedes, Athena Research Center, Greece

Abstract

The widespread adoption of Large Language Models has accelerated news content generation, yet methods to help readers contextualize and responsibly consume news content lag behind. Existing approaches to clustering news articles face significant limitations. Methods based on dense representations often struggle with robustness and efficiency, while those based on sparse (n-gram) representations are unsuitable for cross-lingual applications. Additionally, LLM-based methods are expensive to apply at the needed scale, restricting their use to only well-funded organizations.

We present a novel method for efficient clustering of news articles using cross-lingual, sentence-level embeddings and community detection on similarity graphs. Our method can construct a document-level similarity graph from sentence-level embeddings, avoiding the use of sub-optimal embeddings of entire documents. The proposed method can be executed on consumer hardware, enabling practical news analysis for personal use and research.

We also release a manually curated dataset of clustered news articles, providing a valuable resource for future research. We use the dataset to benchmark our method using seven text embedding models with cross-lingual capability and propose various practical configurations.

1 Introduction

With the ever-increasing production of news and newsworthy content on the web, today's internet user is exposed to an unprecedented volume of information. The recent growth in the capacity of Large Language Models (LLMs) has added a new dimension to the already exploding content production, but at the same time, progress on developing tools for organizing and consuming content lags behind. This asymmetry between the capabilities

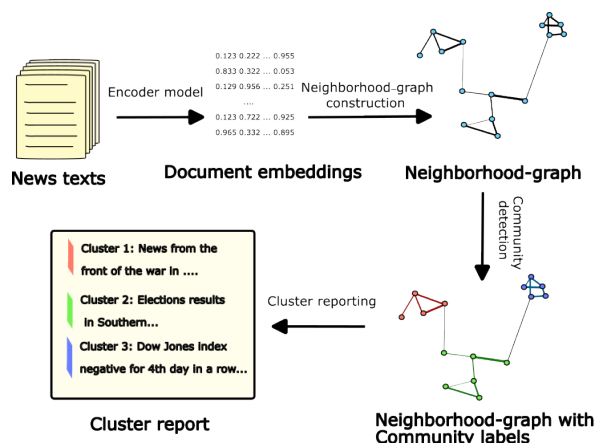


Figure 1: Illustration of the single-vector variant of our news stories clustering method (GCom).

of content creators and that of consumers presents a fertile ground for the dissemination of problematic content, including mis/disinformation, hyper-partisan news, divisive and highly biased content, fake news, and low quality reporting. This situation can overwhelm even adept and highly literate media consumers, rendering them unable to critically reflect on the ongoing discourse, contextualize the content, and gather a diversity of viewpoints.

Consequently, developing tools for rapid, comprehensive and insightful news analysis based on state-of-the-art yet resource-efficient (and thus accessible) methods is more necessary than ever. These methods should be able to handle international discourse by providing cross-lingual capabilities to incorporate news reported in multiple languages. Moreover, this capability should be available within the reach of a consumer with modest computational resources in order to avoid information gate keeping by computationally rich platforms and organizations.

We present a clustering methodology capable of organizing news content into coherent and easily understandable clusters of articles. This enables a news consumer to efficiently digest the large volume of news content and analyze the discourse by

* These authors contributed equally to this research.

detecting divergent viewpoints and perspectives. In addition, it makes various information manipulation methods, such as targeted cross-posting, content reuse, and spinning, more easily detectable. Recent research toward this objective either focuses on studying the cross-lingual representation performance of embeddings (Hanley and Durumeric, 2025) or makes heavy use of LLMs and evaluates using non-standard measures (Ashkenazi et al., 2025). As our method is intended to be practical and within reach of users without large computational resources, we do not require any use of LLM decoding models in our approach, and everything is computed locally on common consumer hardware.

Granular news clustering datasets are scarce, due to the difficulty of retrieving diverse yet similar enough content from the global online news coverage. To facilitate experiments and future research, we also present and release a comprehensive multilingual dataset, **UA-RU News**¹, of news clusters on Russia’s invasion of Ukraine, a manually labeled dataset with articles sourced from a publicly available media analysis corpus² described by Vianini et al. (2026). The dataset comprises **4,251 news articles** curated into **110 clusters**, constituting a valuable resource for further research and analysis. We evaluate 11 different configurations of our method against the widely-used HDBSCAN (McInnes et al., 2017) clustering algorithm, using standard clustering evaluation metrics.

In sum, we present the following contributions:

(1) We present **GCom**, a lightweight and efficient method to cluster multilingual news stories, with two variants: one optimal for long-context embedding models and one for sentence-similarity models, (2) We release a manually curated **multilingual dataset of news clusters**, related to Russia’s aggression against Ukraine, (3) We perform a comprehensive set of experiments using seven different embedding models.

2 Related Work

The first widely acknowledged effort to produce media analysis capability was the Topic Detection and Tracking (TDT) project (Allan et al., 2018). Since then, several frameworks have been proposed to analyze and organize news. Originally, most approaches relied on lexical matching such as BM25

(Nicholls and Bright, 2018), or matrix factorization to perform news story deduplication (Pouliquen et al., 2008). These techniques inherently catered only to the monolingual scenario. Attempts to enable multilingual capacity relied on complicated matrix calculations (Steinberger et al., 2002) and had very poor cross-lingual performance.

More recent approaches rely on dense representations specifically trained to capture semantic similarity across sentences in different languages (Feng et al., 2022). However, past research indicates that producing document-level embeddings by naively extending methods that produce sentence-level embeddings often fails to adequately capture document semantics (Sannigrahi et al., 2023).

The latest developments employ LLM-based components to incrementally produce news clusters. Pattnaik et al. (2024) use an iterative approach combining agglomerative clustering with LLM summarization. Zhang et al. (2023) adopt a similar strategy, with an emphasis on prompting LLMs to refine cluster boundaries. Wang et al. (2023) propose an interesting combination by prompting an LLM to propose clusters, then assign candidate clusters to each document, subsequently framing the problem as Integer Linear Programming to identify optimal clusters. Fu et al. (2025) integrate a Graph Neural Network encoder with a set of LLM agents suggesting cluster assignments. All these methods rely on decoder-LLMs, invoked from one prompt per document to one prompt per cluster (multiple times), generating large numbers of tokens. This results in high costs, high latency, and requires significant resources that the average user lacks, especially at scale.

Closer to our needs, Hanley and Durumeric (2025) utilize data augmentation to fine-tune the embeddings of the models, aiming to enhance cross-lingual performance, yet without considering a real-world clustering scenario. Ashkenazi et al. (2025) employ a document graph representation coupled with an LLM summarization process. However, they evaluate their approach using ad-hoc, non-standard clustering measures.

3 Methods

Our method (*GCom*) relies on embedding-based text representations, efficient nearest neighbor graph construction, and robust graph-based community detection. We propose two variants:

The **single-vector variant** employs a single em-

¹<https://data.europa.eu/doi/10.2905/JRC.RQRFAPG>

²<https://data.europa.eu/doi/10.2905/JRC.4WVSF8>

bedding vector to represent each document. These vectors are used to create a top- k neighborhood similarity graph. Community detection then segments the graph into communities, which are directly utilized as the final document clusters.

The **sentence-level variant** encodes each sentence of a document into its own embedding vector. A k -nearest neighbor graph is constructed from all the sentence vectors, regardless of the documents they come from, and community detection identifies communities of topically similar sentences. The sentence communities are then utilized to form document clusters based on shared topics.

The first variant is better suited to encoders capable of handling directly long text, whereas the second variant is tailored for encoders that handle well only shorter messages or individual sentences.

Hereafter, we distinguish between two key terms: **Communities** are the results of community detection algorithms when applied to a graph constructed using pairwise similarity between embeddings.

Clusters represent the ultimate grouping of documents based on shared content, obtained as the final result of a clustering methodology.

We explore alternative ways to infer clusters from communities produced by different forms of the similarity graph. The proposed methods are designed to be simple and efficient, operating solely on CPU. We avoid model fine-tuning, pooling operations (e.g., based on self-attention), or LLM decoding, ensuring a streamlined and resource-conscious approach to clustering.

3.1 Document embeddings

We employ multilingual encoders to turn textual content into dense vectors. As demonstrated in previous research (Sannigrahi et al., 2023), some language model-based embeddings exhibit their highest effectiveness at the sentence level, as document-level embeddings can be inconsistent and potentially inaccurate when encoding texts of varying lengths. To address this issue, our *sentence-level variant* uses a language-aware sentence splitter to divide each document into sentences, producing a separate embedding for each sentence. By contrast, our *single-vector variant*, which we discuss first, directly encodes entire documents.

3.2 Single-vector variant

We explored three different methods for representing a document as a single vector: encoding the title alone, the title plus a snippet from the docu-

ment’s beginning, and the title with the full text. Once document embeddings have been produced, we construct a k -Nearest Neighbor graph and apply community detection algorithms to partition the graph into coherent communities. The following subsections detail the process of obtaining topical clusters from document embeddings.

3.2.1 k -Nearest Neighbor Graph construction

An approximate k -nearest neighbors adjacency matrix $\mathbf{A}$ is constructed from the document embeddings. This is a sparse $n \times n$ matrix, where n is the total number of documents. Each element a_{ij} of $\mathbf{A}$ is the cosine distance (ranging in $[0, 1]$) between the embeddings of documents i and j , if j is among the approximate k closest neighbors of i , and one otherwise. We use PyNNDescent (Dong et al., 2011) to obtain the approximate top- k neighbors of each document, with $k = 100$. Since we use approximate top- k neighborhoods, $\mathbf{A}$ is not necessarily symmetric. A more detailed analysis of the influence of parameter k on final clustering is provided in Appendix C.3 and Table 4.

Elements a_{ij} exceeding a preset threshold distance t_{cos} are disregarded to eliminate potential outliers; t_{cos} depends on the choice of embedding model (Appendix C.4). The resulting trimmed $\mathbf{A}$ is then used to construct a directed k -nearest neighbor graph, where nodes represent documents. A directed edge from node i to node j is included in the graph if $a_{ij} \leq t_{cos}$. Intuitively, each document is linked to its k nearest neighbors, as long as their cosine distances are below the threshold t_{cos} .

3.2.2 Community detection

We perform modularity-based community detection on the k -nearest neighbor directed graph using the Leiden algorithm, as implemented in the LeidenAlg library (Traag et al., 2019). The main advantage of the Leiden algorithm is its ability to effectively handle directed graphs. During our experimentation we observed that it tends to produce stable partitions while avoiding the frequent problem of forming one very large community due to the presence of a high density core. In the single-vector variant, the communities discovered by Leiden are directly used as document clusters. A key advantage of choosing Leiden as a core building block for our clustering approach, compared to other commonly used clustering algorithms, such as k -means (Hartigan and Wong, 1979), is that it does not require prior knowledge of the number

of clusters. Instead, it offers modular partitioning based solely on the graph’s structural properties.

3.3 Sentence-level variant

The *sentence-level variant* is designed for embedding models that may excel at sentence-level similarity, without producing equally good embeddings for longer text (§3.1). In this variant, we split each document into sentences using a language-aware sentence splitter, and compute one embedding per sentence. We then follow the same k -nearest neighbor graph construction (§3.2.1) and community detection (§3.2.2) steps as in the single-vector variant, but applied to all the sentence embeddings, regardless of the documents they come from, obtaining communities of *sentences*. Finally, we obtain clusters of documents from the sentence communities. The details of this last step are described below.

Let D be the set of all documents in the examined dataset, let s_i^j be the j -th sentence of document d_i , and S_i be the set of all sentences of document d_i ; we treat the title of each document as one of its sentences. Then $S = \bigcup S_i$ is the set of all sentences (now coming from any document d_i) that participated (as nodes) in the community detection step (§3.2.2). Let C denote the resulting set of sentence communities. In effect, we have two binary relations, $\mathcal{R} \subset S \times C$ and $\mathcal{T} \subset S \times D$, which map each sentence s_i^j to exactly one community c_k and exactly one document d_i , as summarized below.

$$\begin{array}{ccc} \mathcal{R}: & S & \longrightarrow C \\ & \Downarrow & \Downarrow \\ & s_i^j & \longmapsto c_k \\ \\ \mathcal{T}: & S & \longrightarrow D \\ & \Downarrow & \Downarrow \\ & s_i^j & \longmapsto d_i \end{array}$$

The two binary relations form a **tripartite graph**, depicted on the left hand side of Figure 2. The first partition comprises documents (left partition), the second one sentences (middle partition), and the third one sentence communities (right partition).

The last step of the sentence-level variant first combines $\mathcal{T}$ and $\mathcal{R}$, eliminating the middle partition of sentences, resulting in a **bipartite graph** with an n -to- n mapping from documents to sentence communities, depicted on the right hand side of Fig. 2. Note that the bipartite graph may contain multi-edges, i.e., a document may be now linked to the same community of sentences via multiple edges (corresponding to different intermediate sentences of the sentences partition that has now be

removed). Alternative methods, discussed below, can then be applied to the bipartite graph to assign a single topical cluster to each document.

Right projection clusters the sentence communities of the right partition. A weighted edge is added between two communities if they share documents from the document partition, resulting in a *community-to-community* (c-to-c) similarity graph. Applying again a community detection algorithm, this time to the c-to-c graph clusters sentence communities. Intuitively, the original sentence communities of the right partition can be thought of as topics; then the right projection clusters topics frequently mentioned together in the same documents. The weights of the edges of the c-to-c graph can be computed using the *Jaccard index* or the *Szymkiewicz–Simpson coefficient* (Ma et al., 2023). See Appendix B for details. In our experiments, we use the latter, which we empirically found to perform better when documents vary in length.

Left projection creates a *document-to-document* (d-to-d) similarity graph on the document partition. A weighted edge is now added between two documents if they share common sentence communities. Applying community detection to the d-to-d graph clusters documents that mention similar topics. Edge weights can again be computed with the *Jaccard index* or *Szymkiewicz–Simpson coefficient*.

Mixed projection: In this case, each multi-edge of the bipartite graph is replaced by a single weighted edge, with the weight being the number of merged edges. Applying community detection to the bipartite graph results in mixed clusters that include both documents and sentence communities (topics). This allows viewing (in each cluster) similar documents along with their shared topics. Furthermore, topics not included in any of mixed clusters can be considered irrelevant or rarely mentioned.

We use the mixed projection in our main experiments. If two documents belong to the same mixed community, they are clustered together. For comparison, we also provide results from the right projection. To detect communities in the c-to-c graph of the right projection, we use again the Leiden algorithm, but with weighted edges. To detect communities (mixed clusters) on the bipartite graph of the mixed projection, we use Leiden optimized for bipartite graphs, as implemented in the LeidenAlg library (Barber, 2007).

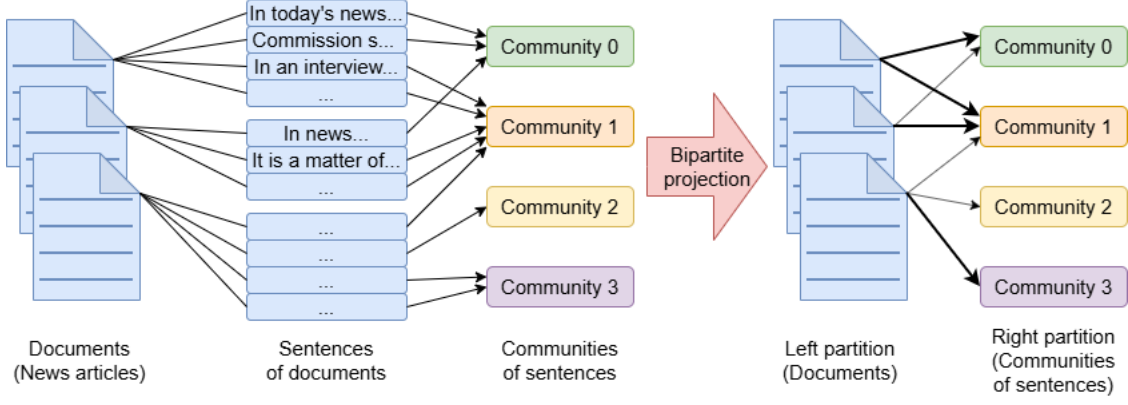


Figure 2: Illustration of the tripartite and bipartite graphs in the sentence-based variant of our clustering method.

	# vectors	# clusters		# singletons		completeness		homogeneity		NMI		AMI		time (minutes)
		HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	
Configuration A: Single-vector variant of GCom, and corresponding HDBSCAN baseline														
Title	4251	2144	821	1823	544	.560	.716	.996	.979	.717	.827	.420	.742	.32 .05
Title + snippet	4251	1800	455	1514	275	.595	.802	.998	.970	.745	.878	.518	.836	.34 .05
Title + text	4251	1855	156	1528	57	.579	.880	.997	.926	.733	.903	.481	.882	.34 .06
Configuration B: Sentence-level variant of GCom with right projection , and corresponding HDBSCAN baseline														
Title + 6 sent.	29270	814	348	627	180	.685	.815	.946	.966	.794	.884	.692	.847	6.62 .58
Title + 6 sent., filt.	25166	927	426	728	241	.686	.803	.975	.974	.805	.880	.700	.839	5.31 .49
Title + all sent.	97650	431	157	284	35	.729	.858	.884	.919	.799	.887	.730	.861	77.50 4.41
Title + all sent., filt.	86154	481	199	314	67	.741	.859	.936	.932	.827	.894	.761	.868	61.98 3.71
Configuration C: Sentence-level variant of GCom with mixed projection , and corresponding HDBSCAN baseline														
Title + 6 sent.	29270	505	210	367	91	.732	.918	.910	.926	.811	.922	.743	.904	6.41 .28
Title + 6 sent., filt.	25166	663	283	508	150	.734	.912	.953	.955	.829	.933	.757	.917	5.15 .25
Title + all sent.	97650	248	104	126	18	.763	.954	.894	.842	.823	.895	.772	.878	71.01 1.07
Title + all sent., filt.	86154	308	126	176	39	.778	.953	.932	.869	.848	.909	.801	.893	59.00 .95

Table 1: Experimental results on **UA-RU News** using the **LaBSE** text encoder.

	# vectors	# clusters		# singletons		completeness		homogeneity		NMI		AMI		time (minutes)
		HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	
Configuration A: Single-vector variant of GCom, and corresponding HDBSCAN baseline														
Title	13874	7893	5337	7163	4331	.576	.651	.993	.992	.729	.786	.371	.583	3.47 .12
Title + snippet	13874	7006	3232	6343	2555	.602	.746	.994	.987	.750	.850	.456	.752	3.61 .15
Title + text	13874	6907	1497	6125	1041	.596	.829	.996	.954	.746	.887	.441	.837	3.62 .18
Configuration B: Sentence-level variant of GCom with right projection , and corresponding HDBSCAN baseline														
Title + 6 sent.	93474	4635	2454	4132	1931	.647	.778	.954	.956	.771	.858	.577	.781	40.62 3.03
Title + 6 sent., filt.	85618	5153	2788	4637	2226	.643	.771	.974	.977	.775	.862	.568	.781	34.81 1.24
Title + all sent.	345041	2330	1614	1973	1173	.721	.792	.867	.906	.787	.845	.678	.775	507.54 23.11
Title + all sent., filt.	309385	2762	1830	2355	1313	.725	.811	.915	.950	.809	.875	.698	.814	368.74 13.84
Configuration C: Sentence-level variant of GCom with mixed projection , and corresponding HDBSCAN baseline														
Title + 6 sent.	93474	3141	1667	2882	1316	.710	.864	.833	.851	.767	.858	.643	.806	38.81 2.52
Title + 6 sent., filt.	85618	3731	2114	3369	1613	.724	.870	.918	.943	.810	.905	.689	.861	33.57 .79
Title + all sent.	345041	1262	878	1118	594	.795	.908	.779	.821	.787	.862	.720	.822	475.29 10.17
Title + all sent., filt.	309385	1736	1270	1518	925	.805	.906	.852	.892	.828	.899	.762	.864	344.84 3.13

Table 2: Experimental results on **Miranda** using the **LaBSE** text encoder.

4 Experiments and Results

To assess the performance of our approach, *GCom*, including its two variants, we experimented on two datasets, one that we created specifically to compare the performance of clustering methods on news articles and an existing one, using seven embedding models. We compare our method to

the widely-used HDBSCAN (McInnes et al., 2017) across a variety of configurations, using standard clustering evaluation metrics. Aside from its popularity, we chose HDBSCAN for its relative insensitivity to hyper-parameters or need to provide an estimate of the number of clusters. We used three HDBSCAN implementations (in Python,

model	configuration	NMI		AMI	
		HDBSCAN	GCom	HDBSCAN	GCom
LaBSE	title	.717	.827	.420	.742
	title+text	.733	.903	.481	.882
	title+6 sent., filtered	.829	.933	.757	.917
mE5-large	title	.722	.857	.444	.822
	title+text	.749	.871	.543	.849
	title+6 sent., filtered	.808	.913	.718	.892
GTE	title	.725	.884	.452	.859
	title+text	.765	.913	.583	.896
	title+6 sent., filtered	.831	.935	.757	.919
BGE-M3	title	.722	.847	.441	.785
	title+text	.759	.921	.564	.905
	title+6 sent., filtered	.832	.936	.763	.922
nomic-v2	title	.723	.862	.446	.811
	title+text	.749	.916	.540	.899
	title+6 sent., filtered	.825	.926	.747	.911
Gemma-0.3b	title	.706	.779	.380	.687
	title+text	.743	.872	.518	.847
	title+6 sent., filtered	.782	.863	.677	.820
Qwen3-0.6B	title	.717	.828	.422	.766
	title+text	.743	.908	.521	.890
	title+6 sent., filtered	.818	.910	.731	.891
Qwen3-4B	title	.714	.831	.409	.759
	title+text	.759	.924	.569	.905
	title+6 sent., filtered	.825	.929	.747	.912
Qwen3-8B	title	.723	.859	.443	.812
	title+text	.765	.915	.589	.897
	title+6 sent., filtered	.823	.933	.740	.918

Table 3: *NMI* and *AMI* results on **UA-RU News**, using **different embedding models**, for three indicative setups: Configuration A with titles only (title) or full text (title+text), Configuration C with titles, 6 leading sentences, and sentence filtering (title+6 sent., filtered). Similar results are obtained on Miranda (Appendix E).

C++, Rust)³ also measuring execution times, on the same compute infrastructure. We performed all tests using a single *Quadro RTX 8000 GPU (48GB)* and an *AMD EPYC 7282 16-Core Processor*. We report detailed results for the Python implementation (McInnes and Healy, 2017), which yielded the best performance.

4.1 Sentence filtering strategy

In our development runs, we noticed that the *sentence-level variant* was sensitive to generic outlier sentences. To make the *sentence-level variant* of GCom more robust to real-world conditions, we experimented with filtering sentences containing incomplete fragments, scraping errors, or irrelevant text (e.g., advertisements). For the latter, we employ a light irrelevant sentence classifier; more details can be found in Appendix A.3.

4.2 Datasets

Due to the relative shortage of full-text online news clustering datasets, we created a novel dataset, **UA-RU News** (Kovacikova and Nikolaidis, 2026) to

evaluate our method in a real-world online news setting. It was built as a subset of a larger publicly available multilingual corpus (Kovacikova, 2026) of real-world articles on the Russian invasion of Ukraine. To obtain a set of consistent topic clusters, we restricted the data to one complete month of publication (June 2024). Using the named-entity and geolocation annotations already provided with the source corpus, we grouped articles by shared entity mentions and shared geolocation into thematic groups. Because all articles fall within a narrow time window, these groups already approximate topical stories, and the entity and geolocation signals proved a strong scaffold for the manual work that followed.

Two annotators, both NLP specialists experienced with news data, then refined the thematic groups into the final clusters. They worked from the article titles and English machine translations produced by Meta’s NLLB-200 model (NLLB Team et al., 2022), consulting the original-language text whenever a detail needed checking. Working independently, each annotator could apply one of the following operations on every thematic group:

- (i) *Remove* articles not relevant to the group;
- (ii) *Merge* groups that describe the same story;
- (iii) *Split* a group that conflated several stories;
- (iv) (optionally) Write a *short description* for the resulting cluster.

The short description, operation (iv), was left unused for the experiments of this work, but was included in the dataset for future exploitation.

The guiding criterion was that *each cluster must correspond to a single newsworthy event*. Later updates, summaries, and refinements of that event belong in the same cluster, whereas a distinct follow-up event, or a reaction or opinion piece that merely refers to it, forms a separate cluster. Articles that share only named entities but describe different events are kept apart. This criterion is deliberately strict, because the corpus is topically homogeneous. Since every article concerns Ukraine and Russia, shared entities alone are a weak indicator that two articles report the same event. OK The two annotators produced identical clusters for the majority of the cases. The remaining disagreements, concentrated in fewer than 10% of clusters, were reconciled in a joint consolidation session held after independent annotation, in which the annotators discussed each disagreement and agreed on a single

³<https://github.com/scikit-learn-contrib/hdbscan>,
https://github.com/TutteInstitute/fast_hdbscan/,
<https://crates.io/crates/hdbscan>.

label. Disagreements typically concerned whether two groups with overlapping entities represented the same story or two distinct ones. Specific details regarding the annotation process are presented in Appendix A.4 and in the data repository¹. The final dataset comprises **110 clusters over 4,251 articles in 39 languages**, most frequently English. More details on the annotation process and dataset statistics are provided in Appendix A.1 and in the publicly available web page dedicated to this dataset¹ (Kovacikova and Nikolaidis, 2026).

In addition, we conducted the same set of experiments on **Miranda** (Miranda et al., 2018), a dataset containing annotated clusters of multilingual news articles. This dataset is derived from a collection of article pairs (Rupnik et al., 2016), originally designed for a binary classification task.

4.3 Configurations

We assessed the performance impact of each component of GCom by conducting a series of experiments using different configurations:

A. Single-vector variant (§3.2): It represents each document by a single embedding vector, constructs a k -nearest neighbors graph, and applies community detection to the graph obtaining document clusters. The baseline is the direct application of HDBSCAN, using the same document embeddings.

B. Sentence-level variant, right projection (§3.3): It encodes each sentence separately, constructs a k -nearest neighbors graph from the sentence embeddings, applies community detection to the graph, then uses the right projection (another application of community detection to an additional community-to-community graph). The baseline is applying HDBSCAN to the same sentence embeddings, to get sentence clusters (instead of sentence communities), then the same right projection.

C. Sentence-level variant, mixed projection (§3.3): Same as (B), but instead of the right projection, the second application of the community detection algorithm (mixed projection) is on the (weighted) bipartite graph of documents and sentence communities. The baseline is applying HDBSCAN to obtain sentence clusters (corresponding to sentence communities), then forming the (weighted) bipartite graph of documents and sentence clusters, and applying the same mixed projection.

For configuration **A**, we conducted tests where we embedded the *title* alone, the *title* combined

with the *leading snippet* of 350 characters, and the *full text* of each article. We applied each of these to HDBSCAN for comparison. For **B** and **C**, we used (again in both GCom and HDBSCAN) the title combined with either the first six leading sentences or all sentences; we also measure the effect of the irrelevant sentence classifier (§4.1). In all three configurations, we utilized the LeidenAlg library with its default parameters except for the resolution parameter (Appendix C.2).

4.4 Text encoders

In all configurations, we experiment with seven encoders (embedding models): **LaBSE** (Feng et al., 2022), **BGE-M3** (Chen et al., 2025), **GTE-multilingual-base** (Zhang et al., 2024), **nomic-v2-moe** (Nussbaum et al., 2024), **multilingual-E5-large** (Wang et al., 2024), **EmbeddingGemma** (Schechter Vera et al., 2025), **Qwen3-Embedding** (Zhang et al., 2025) in the 0.6B, 4B, 8B variants. Of those, **Qwen3-Embedding** and **GTE-multilingual-base** were explicitly trained to work efficiently with long-text understanding and retrieval.

4.5 Evaluation measures

To evaluate the generated clusters, we use internal metrics from Rosenberg and Hirschberg (2007), specifically *completeness*, *homogeneity*, and two mutual information (entropy-based) metrics from Vinh et al. (2010), namely *NMI* and *AMI*.

4.6 Evaluation of the generated clusters

We present detailed results for UA-RU News and Miranda using the LaBSE encoder in Tables 1 and 2, respectively. Table 3 shows *NMI* and *AMI* results for all encoders in all three configurations. In addition, we provide the summarized results of all the text encoder models for *title* only and for the top performing configurations (*title+text* and *title+6sent.,filtered*) in Table 3. The detailed results for all models and metrics are provided in Appendix E.

Cluster Homogeneity and Completeness: Tables 1 and 2 show that HDBSCAN consistently produces highly homogeneous clusters; however, it often struggles with completeness, because it generates smaller, coherent, but incomplete clusters and more singletons (unclustered items) than GCom (and, hence, many more clusters in total). Previous work (Allaoui et al., 2020) shows that the performance of HDBSCAN can be improved by applying

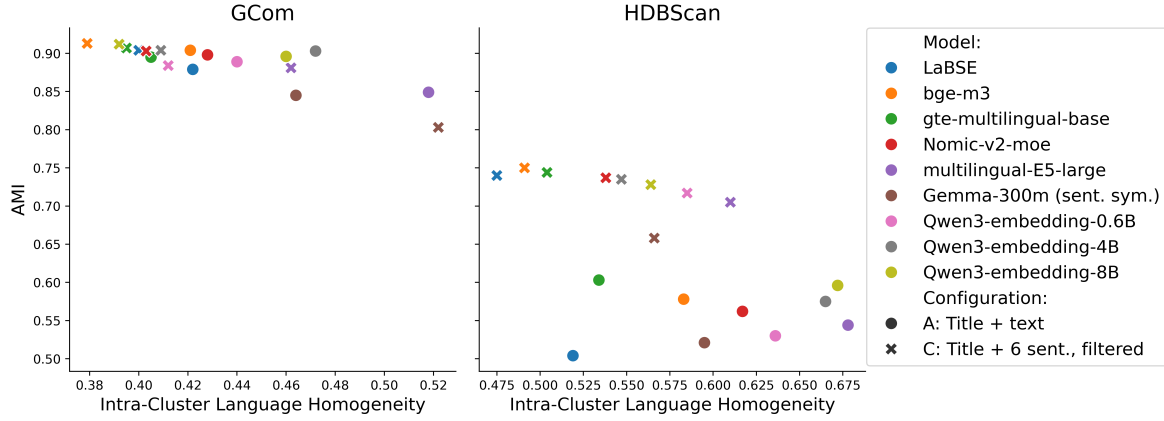


Figure 3: Topic Coherence (AMI) vs. Linguistic Bias (measured by intra-cluster language homogeneity) on **UA-RU News** for **GCom** and **HDBSCAN**, with the full text of configuration A and title and 6 sentences of configuration C.

dimensionality reduction, notably UMAP, before clustering. Our experiments on UMAP as preprocessing steps are included in Appendix D. While this improves clustering, it also adds complexity, necessitating the tuning of additional hyperparameters. GCom clearly outperforms HDBSCAN in completeness in all three configurations, while performing on par in homogeneity. In both datasets, the completeness of both HDBSCAN and GCom improves as we change configurations from A to B and C, at the expense of a milder deterioration of homogeneity. We observe similar overall trends when using the other text encoders (Appendix E). Finally, we show specific cases of clustering failure in Appendix F, with examples of incorrect merges and splits in Tables 6 and 7.

Mutual information metrics: Tables 1 and 2 show that GCom consistently outperforms HDBSCAN across all configurations in NMI and AMI . Again, results improve as we change configurations from A to B and C. Table 3 shows that GCom continues to consistently outperform HDBSCAN in NMI and AMI when using other embedding models.

Configurations B vs. C: Among the two configurations that create sentence-level embeddings, Configuration B results in lower scores (Tables 1 and 2) than Configuration C.

Configurations A vs. C: Our analysis (Table 3) highlights the good performance of Qwen models, including the smallest variant Qwen3-0.6B, when encoding the entire text as a single embedding (Configuration A). Configuration C introduces only marginal enhancements to Qwen’s effectiveness, compared to the simpler Configuration A. Despite these modest gains, Qwen’s ability to handle

long-context embeddings remains a strong asset, providing valuable insights into detailed narrative structures, without necessity to work with shorter parts of news articles to correctly capture the core of the narrative. However, for scenarios requiring news analysis on less powerful hardware, GCom’s sentence-level approach with smaller embedding models offers a more pragmatic choice, enhancing performance with simplicity and speed. More generally, the boost from Configuration A to C is more pronounced for smaller embedding models (Table 3), but marginal for larger models, suggesting that larger models in combination with the simpler Configuration A may be preferable when computational resources are abundant.

Effect of intra-language bias: We wanted to assess how sensitive GCom in combination with tested embedding models is to the inherent tendency of models to consider pairs of the same language as more similar than pairs of different languages (language bias). Fig. 3 illustrates the relationship between the Adjusted Mutual Information (AMI) score and intra-cluster language homogeneity across the tested embedding models. We focus on the best performing cases from configurations A and C, the full text and title combined with 6 leading sentences after filtering, respectively. The higher the AMI , the better the clustering (overall); the lower the intra-cluster language homogeneity, the more language-diverse the clusters are. Hence, the best results in Fig. 3 are those in the upper left parts of the two subfigures. Obtaining high AMI scores with high language homogeneity (top right), indicates that clustering coherence is achieved primarily through language-based separation rather than cross-lingual topic extraction. Fig. 3 shows

that clusters resulting from scenario C exhibit noticeably less linguistic bias compared to scenario A across all embedding models. Additionally, it highlights LaBSE’s advantage, which is optimized for both multilingual texts and shorter messages. We notice that using the sentence-level approach enables LaBSE (a smaller and older model) to compete effectively with larger models trained to embed longer text.

4.7 Efficiency and computational resources

A critical practical advantage of our approach (*GCom*) is its resource efficiency. As our stated objective is to enable widespread large-scale news analysis capability for public use, we also considered the computational performance of the two approaches in terms of CPU time. The runtime of *GCom* on the above machine varied from **3 sec** (single-vector) to **5 minutes** (sentence-level), while all HDBSCAN implementations we tested ranged from **25 sec** (single-vector) to **12 hours** (sentence-level), which presents a very noticeable difference. This trend was consistent with all three HDBSCAN implementations we tested. The detailed time analyses including number of embedding vectors used for clustering process using LaBSE model are presented in Tables 1 and 2. When it comes to embedding calculation for both methods, all models except *Qwen3-4B* and *Qwen3-8B* could fit on a significantly less powerful GPU than the one we used (Quadro RTX 8000 with 48GB) with even half the GPU memory, and the two smallest models (*LaBSE*, *Gemma-0.3B*) can fit even in GPU cards with 8GB VRAM. Thus, the GPU requirement is rather modest and within the reach of a consumer’s setup.

5 Conclusion and future work

Our method of clustering news articles (*GCom*) using community detection on a neighborhood graph of embedding vector similarities presents an efficient approach that yields better results than the application of the established HDBSCAN algorithm in the news domain. Using multilingual embedding models, along with the efficient implementations of PyNNDescent and the LeidenAlg library, we have shown the effectiveness of our method in a real-world use case.

Further benefits include avoiding the curse of dimensionality and allowing the community detection algorithm to function without requiring any

parameters to predefine the number of final clusters. For future work, we plan to:

- Propose and compare more complex methods for aggregating using bipartite weighted graphs to assign a single cluster to each document based on content.
- Experiment with ways to incorporate metadata (source details, engagement, temporal aspects) into the similarity graphs.
- Implement community detection through pooling with Graph Neural Networks (GNN) (Gkarpounis et al., 2024), which could improve the quality of the final clustering of documents and also reveal the intricacies of the clusters or included documents.

6 Limitations

Due to time and resource constraints, we did not perform a full hyper-parameter sweep for all the experiment runs. Also, due to the relatively small size of the datasets, we did not produce a dev/test split of each dataset. Instead, we performed a sweep in a single scenario and used the optimal hyper-parameters for all the others.

Since we rely on representations from the embeddings of pretrained encoders without any further fine-tuning, the notion of similarity used depends solely on the training of the respective model. Thus, the method inherits any bias present in these models.

We limited the scope only to textual content and did not address multi-modality aspects.

7 Ethics

Annotators were adequately compensated using the fixed rates of their contracting institutions.

Access to the full content of the articles might be subject to copyright, licensing restrictions or require justification of research-use exceptions. This could contain legal risk in certain jurisdictions and could make reproduction of this work harder.

Although the use of clustering with the objective of mapping the news discourse is intended and optimized to help a responsible reader digest the overwhelming news content produced, it could also potentially be abused by a malicious user to craft a more effective influence campaign.

8 Acknowledgments

This work has been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

References

- James Allan, Jaime G. Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 2018. [Topic detection and tracking pilot study final report](#).
- M. Allaoui, M.L. Kherfi, and A. Cheriet. 2020. [Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study](#). In *Image and Signal Processing. ICISP 2020. Lecture Notes in Computer Science*, volume 12119. Springer, Cham.
- Yoel Ashkenazi, Etzion Harari, Regev Yehezkel Imra, Naphtali Abudarham, Dekel Cohen, and Yoram Louzoun. 2025. [D2CS - documents graph clustering using LLM supervision](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23606–23623, Suzhou, China. Association for Computational Linguistics.
- Michael J. Barber. 2007. [Modularity and community detection in bipartite networks](#). *Physical Review E*, 76(6):066102.
- Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2025. [M3-embedding: Multilinguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation](#). *Preprint*, arXiv:2402.03216.
- Wei Dong, Moses Charikar, and Kai Li. 2011. [Efficient k-nearest neighbor graph construction for generic similarity measures](#). In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*, pages 577–586, Hyderabad, India. ACM.
- Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. [Language-agnostic BERT sentence embedding](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.
- Yiwei Fu, Yuxing Zhang, Chunchun Chen, Jianwen Ma, Quan Yuan, Rong-Cheng Tu, Xinli Huang, Wei Ye, Xiao Luo, and Minghua Deng. 2025. [MARK: Multi-agent collaboration with ranking guidance for text-attributed graph clustering](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6057–6072, Vienna, Austria. Association for Computational Linguistics.
- Georgios Gkarpounis, Christos Vranis, Nicholas Vretos, and Petros Daras. 2024. [Survey on graph neural networks](#). *IEEE Access*, 12:128816–128832.
- Hans William Alexander Hanley and Zakir Durumeric. 2025. Hierarchical level-wise news article clustering via multilingual matryoshka embeddings. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2476–2492, Vienna, Austria. Association for Computational Linguistics.

- J. A. Hartigan and M. A. Wong. 1979. [Algorithm as 136: A k-means clustering algorithm](#). *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.
- Kristina Kovacikova. 2026. Tracking the Narrative: A Data-Driven Analysis of Media Coverage of Russia and Ukraine 2013–2024 [Dataset]. European Commission, Joint Research Centre (JRC). DOI: [10.2905/JRC.4WVSYP8](#), [10.2905/184be9b0-4758-401a-afe1-8031ffe94721](#). PID: [http://data.europa.eu/89h/184be9b0-4758-401a-afe1-8031ffe94721](#).
- Kristina Kovacikova and Nikolaos Nikolaidis. 2026. UA-RU News: A Manually Annotated Multilingual Benchmark for News Story Clustering. European Commission, Joint Research Centre (JRC). DOI: [10.2905/JRC.RQRFAPG](#). PID: [http://data.europa.eu/89h/fc214973-ce7f-401f-ad6d-2a08fa97cd1d](#).
- T. Ma, L. Guo, H. Yan, and L. Wang. 2023. [Cobind: quantitative analysis of the genomic overlaps](#). *Bioinform Adv*, 3(1):vb0104.
- Leland McInnes and John Healy. 2017. Accelerated hierarchical density based clustering. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pages 33–42. IEEE.
- Leland McInnes, John Healy, and Steve Astels. 2017. [hdbscan: Hierarchical density based clustering](#). *Journal of Open Source Software*, 2(11):205.
- Sebastião Miranda, Artūrs Znotiņš, Shay B. Cohen, and Guntis Barzdins. 2018. [Multilingual clustering of streaming news](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4535–4544, Brussels, Belgium. Association for Computational Linguistics.
- Tom Nicholls and Jonathan Bright. 2018. [Understanding news story chains using information retrieval and network clustering techniques](#). *Communication Methods and Measures*, 13(1):43–59.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Hefernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2022. [No language left behind: Scaling human-centered machine translation](#). *Preprint*, arXiv:2207.04672.
- Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. [Nomic embed: Training a reproducible long context text embedder](#). *arXiv*, abs/2402.01613.
- Anup Pattanaik, Cijo George, Rishabh Kumar Tripathi, Sasanka Vutla, and Jithendra Vepa. 2024. [Improving hierarchical text clustering with LLM-guided multi-view cluster representation](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 719–727, Miami, Florida, US. Association for Computational Linguistics.
- Bruno Pouliquen, Ralf Steinberger, and Olivier Deguerne. 2008. [Story tracking: linking similar news over time and across languages](#). In *Coling 2008: Proceedings of the workshop Multi-source Multilingual Information Extraction and Summarization*, pages 49–56, Manchester, UK. Coling 2008 Organizing Committee.
- Andrew Rosenberg and Julia Hirschberg. 2007. [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, Prague, Czech Republic. Association for Computational Linguistics.
- Jan Rupnik, Andrej Muhič, Gregor Leban, Primož Škraba, Blaž Fortuna, and Marko Grobelnik. 2016. [News across languages - cross-lingual document similarity and event tracking](#). *Journal of Artificial Intelligence Research*, 55:283–316.
- Sonal Sannigrahi, Josef van Genabith, and Cristina España-Bonet. 2023. [Are the best multilingual document embeddings simply based on sentence embeddings?](#) In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2306–2316, Dubrovnik, Croatia. Association for Computational Linguistics.
- Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, and 69 others. 2025. [EmbeddingGemma: Powerful and lightweight text representations](#). *Preprint*, arXiv:2509.20354.
- Ralf Steinberger, Bruno Pouliquen, and Johan Hagman. 2002. [Cross-lingual document similarity calculation using the multilingual thesaurus EUROVOC](#). In *Computational Linguistics and Intelligent Text Processing, Third International Conference, CICLing 2002, Mexico City, Mexico, February 17-23, 2002, Proceedings*, volume 2276 of *Lecture Notes in Computer Science*, pages 415–424. Springer.
- V. A. Traag, L. Waltman, and N. J. van Eck. 2019. [From louvain to leiden: Guaranteeing well-connected communities](#). *Scientific Reports*, 9:5233.
- Irene Vianini, Sopho Kharazi, Bonka Kotseva, Kristina Kovacikova, Nicolò Faggiani, Nikolaos Nikolaidis, Leonida Della Rocca, Kristina Potapova, Olena Snigyr, and Jens P. Linge. 2026. [Tracking the narrative: A data-driven analysis of media coverage of russia and ukraine 2013–2024](#). *PLOS ONE*, 21(6):1–33.

- Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854.
- Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.
- Zihan Wang, Jingbo Shang, and Ruiqi Zhong. 2023. [Goal-driven explainable clustering via language descriptions](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10626–10649, Singapore. Association for Computational Linguistics.
- Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. 2024. [mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412, Miami, Florida, US. Association for Computational Linguistics.
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.
- Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023. [ClusterLLM: Large language models as a guide for text clustering](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13903–13920, Singapore. Association for Computational Linguistics.

A Datasets

A.1 UA-RU News dataset

UA-RU News is a subset of the publicly available multilingual corpus on the conflict between Ukraine and Russia (Kovacikova, 2026), which comprises over 22 million articles. As described in Section 4.2, we restricted the corpus to a single complete month of publication (June 2024), formed thematic groups from shared named-entity mentions and geolocation, and had two annotators independently refine these groups into clusters, followed by a joint adjudication session to resolve disagreements.

The named-entity mentions (mostly persons and organisations, extracted from the full article text) and the geolocation tags are those provided with the source corpus; we did not run our own entity recognition. The final dataset contains 4,251 articles in 39 languages. The most frequent are English (1,575), Russian (434), Spanish (350), German (286) and Bulgarian (281); the full language distribution is released with the dataset. The 110 clusters range in size from 2 to 423 articles, with a mean of 38.65.

A.2 Miranda

The dataset comprises 13,874 articles spanning nine languages: English (8,726), Spanish (2,177), German (2,101), Chinese (440), Russian (231), Italian (88), French (61), Slovenian (37), and Croatian (13). It is organized into 394 topical clusters, with cluster sizes ranging from 1 to 1,869, and an average cluster size of 35.21.

A.3 Sentence filtering, details

Our objective was to establish a proof of concept indicating that pre-cleaning the collection of sentence representatives before constructing the distance graph can enhance the performance of the final clustering algorithm. To this end, we implemented a straightforward binary classification model. This part of our work serves as a foundation, providing ample opportunity to investigate the efficacy of more complex approaches in future research. The idea is inspired by the work of Feng et al. (2022), where the authors describe a "sentence quality classifier" used prior to the training process of the LaBSE embedding model they developed.

We utilized the Gradient Boosting Classifier implemented in the sklearn library, configuring it with the following parameters: $n_estimators = 200$,

$learning_rate = 0.1$ and $max_depth = 3$. The model was trained on a dataset comprising 262,693 labeled sentences in 75 different languages. Each input to the model consists of an embedding vector representing the sentence, concatenated with the sentence's length. The classifier outputs a binary label, indicating whether the sentence meets the quality criteria.

A.4 Annotation process and guidelines

The two annotators followed the instructions shown in Figure 4:

```

BEGINNING of Annotation:
1. In the file received, choose a value from the
Entity groups, using the filter option.

2. Take a look briefly at the *used_entities* column
to get a glimpse of the detected Entities.

3. You will put the annotations in the following two
columns:
  * Similarity
  * Subcluster

PHASE 1: Pair matching

Starting from the first row:
1. Read the first title and description (use the EN
version if needed)
2. Read the second row.
  2.1 If the second row belongs in the *same story*
to the first row:
    WRITE "1" to the *Similarity* column
    If not:
    WRITE "0" to the *Similarity* column

3. Continue until all the non-similar articles are
removed.

PHASE 2: Subcluster consolidation

If you saw that there are one or more Groups of
articles in the cluster that share a *common story*,
sufficient to cluster them together:

For each new *Group*:
  * Make sure Similarity is "0" for all
  * Add a number starting from 1 to the *Subcluster*
column
  * Check if there are other articles in the previous
Group that could be included in the new Group
  * Check if the story coincides with a group
previously annotated.

```

Figure 4: Annotation instructions for the clustering annotation campaign.

After the first annotation pass where annotators created their independent groupings was completed, the annotators proceeded to a bilateral consolidation stage, where a single cluster assignment was finalized. In terms of per-cluster performance, the agreement was over 90%. Specific details can be found on the data repository.

The following criteria were agreed to decide that two articles belong to the same story:

1. They should refer to the same event, and not to a noticeable evolution of events or a response to some event.
2. They need not follow the same tone or framing but they should have a significant overlap in the information provided.
3. Analyses, opinions and reporting should be separated unless they explicitly refer to the same information (usually triggered by an event).
4. Elaborations or refinements (updates, summaries) of the same event should be grouped together.

B Definitions

Let us consider two sets, A and B .

B.1 Jaccard index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

B.2 Szymkiewicz–Simpson coefficient

$$SS(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}.$$

C Hyperparameter choices

C.1 HDBSCAN

We performed a hyperparameter sweep with α values from 1.0 to 1.30 with a step of 0.05, for the single-vector configuration of the LaBSE embedding model. We also set $min_cluster_size = 3$ to align with the same configurations used in the GCom case.

C.2 LeidenAlg

For all configurations, we utilized the default parameters except for the resolution parameters. In every instance of community detection on the distance graph, we employed the partition type `RBConfigurationVertexPartition` of `LeidenAlg` with resolution set to 10.

For community detection in the weighted bipartite graph, we utilized `CPMVertexPartition.Bipartite`, setting all three resolution parameters (one for within each of the two partitions, and one for the edges between partitions) to 10^{-3} for the **UA-RU News** dataset and 10^{-4} for the **Miranda** dataset.

C.3 Parameter k in approximate k -nearest neighbor graph

The experiments in Table 4 demonstrate that for the single-vector approach (Configuration A), higher values of parameter k , representing a more detailed neighborhood, improve performance up to a specific threshold, beyond which scores begin to decline. In contrast, Configuration C is significantly less sensitive to variations in this parameter. Nevertheless, we maintain $k = 100$ across all experiments for consistency.

dataset	configuration	LaBSE				
		k=10	50	100	150	200
UA_RU News	A: title+text	.816	.859	.884	.857	.822
	C: title+6 sent., filt.	.917	.916	.918	.919	.918
Miranda	A: title+text	.809	.837	.837	.834	.832
	C: title+6 sent., filt.	.861	.861	.861	.858	.861
dataset	configuration	Qwen3-Embedding-4B				
		k=10	50	100	150	200
UA_RU News	A: title+text	.799	.895	.907	.877	.846
	C: title+6 sent., filt.	.910	.907	.911	.906	.910
Miranda	A: title+text	.822	.858	.859	.862	.856
	C: title+6 sent., filt.	.874	.875	.875	.875	.875

Table 4: AMI scores across different values of parameter k . Calculations utilize the LaBSE and Qwen3-Embedding-4B text encoder model. Results represent the median value across five clustering runs per parameter setting.

C.4 Settings for the parameter t_{cos}

The values are the result of parameter tuning conducted on each dataset. The optimal parameter was determined separately for single sentence vector representation (configuration A) and multiple vector representatives per document (configurations B and C).

In addition, for Gemma-0.3B we tested multiple prompts to adapt the model to the clustering task. The results show that the most suitable is *prompt="task: sentence similarity | query:"*. Therefore we use only this setting in figures where comparison of models is highlighted.

Embedding model	UA-RU News		Miranda	
	A	B, C	A	B, C
LaBSE	0.75	0.70	0.70	0.75
BGE-M3	0.80	0.70	0.80	0.65
GTE-multilingual-base	0.75	0.70	0.75	0.60
nomic-embed-text-v2-moe	0.90	0.80	0.90	0.80
multilingual-e5-large	0.55	0.45	0.50	0.45
Gemma-0.3B (clustering)	0.60	0.35	0.60	0.35
Gemma-0.3B (sent. sym.)	0.70	0.60	0.70	0.60
Gemma-0.3B (no prompt)	0.75	0.60	0.75	0.55
Qwen3-Embedding-0.6B	0.85	0.75	0.90	0.65
Qwen3-Embedding-4B	0.75	0.70	0.90	0.65
Qwen3-Embedding-8B	0.80	0.70	0.80	0.65

Figure 5: Optimal values of parameter t_{cos} for **UA-RU News** and **Miranda** datasets.

D Preprocessing via UMAP dimensionality reduction model

For this experiment we use the standard *umap-learn* library with cosine metric and $n_neighbors$ parameter set to 100. The results for different settings of $n_components$ are presented in Table 5. Since **UMAP** transforms the data space from Spherical to Euclidean, in the case of **GCom** new values for distance threshold (alternative of t_{cos} for Euclidean metric) had to be found. The ideal distance threshold for configuration **A** is 0.5 and for configuration **C** it is 0.1.

We observe significant improvements for **HDBSCAN**, especially for the case of configuration **'A: title+texts'** while **GCom**'s scores change only slightly and in both cases of **'C: title+6 sent.'** and **'C: title+6 sent., filtered'** the scores decreased. Nevertheless, **HDBSCAN** doesn't outperform our method, **GCom**, and the computation time for **HDBSCAN** remains too high for experiments with larger datasets.

dim.	configuration	AMI		time (minutes)	
		HDBSCAN	GCom	HDBSCAN	GCom
NO	A: title+text	.481	.882	.34	.06
	C: title+6 sent.	.743	.904	6.41	.28
	C: title+6 sent., filtered	.757	.917	5.15	.25
	C: title+all sent.	.772	.878	71.01	1.07
	C: title+all sent., filtered	.801	.893	59.00	.95
8	A: title+text	.756	.894	.23	.29
	C: title+6 sent.	.855	.892	1.02	1.51
	C: title+6 sent., filtered	.836	.912	.78	1.49
	C: title+all sent.	.871	.873	3.92	5.82
	C: title+all sent., filtered	.863	.895	3.35	5.83
16	A: title+text	.750	.891	.27	.35
	C: title+6 sent.	.846	.898	1.04	1.65
	C: title+6 sent., filtered	.860	.907	.89	1.65
	C: title+all sent.	.871	.880	4.66	6.46
	C: title+all sent., filtered	.864	.890	3.86	6.45
32	A: title+text	.791	.892	.33	.36
	C: title+6 sent.	.860	.891	1.10	1.53
	C: title+6 sent., filtered	.856	.912	.95	1.52
	C: title+all sent.	.873	.877	5.70	7.39
	C: title+all sent., filtered	.866	.894	5.12	7.39
64	A: title+text	.794	.894	.61	.42
	C: title+6 sent.	.848	.896	1.75	1.74
	C: title+6 sent., filtered	.853	.909	1.40	1.73
	C: title+all sent.	.870	.871	14.63	9.39
	C: title+all sent., filtered	.871	.900	11.10	9.39
128	A: title+text	.808	.894	.46	.53
	C: title+6 sent.	.855	.894	2.87	2.34
	C: title+6 sent., filtered	.854	.913	2.16	2.33
	C: title+all sent.	.882	.887	23.32	11.72
	C: title+all sent., filtered	.873	.906	17.90	11.71
256	A: title+text	.782	.896	.69	.94
	C: title+6 sent.	.858	.885	5.78	3.57
	C: title+6 sent., filtered	.865	.915	4.06	3.57
	C: title+all sent.	.878	.874	39.79	15.99
	C: title+all sent., filtered	.880	.888	30.23	15.96
512	A: title+text	.805	.895	1.16	1.62
	C: title+6 sent.	.863	.885	10.14	5.86
	C: title+6 sent., filtered	.852	.911	7.42	5.86
	C: title+all sent.	.878	.873	68.12	26.74
	C: title+all sent., filtered	.871	.899	56.95	26.72

Table 5: **AMI** results together with computational time on **UA-RU News** after dimensionality reduction via **UMAP model**. The column **dim.** stands for final dimensionality of dataset after the reduction (UMAP parameter `n_components`). The first set of results, labeled as *dim. = NO* serves as a baseline and represents the case where no UMAP was applied.

E Results of additional experiments

model	configuration	NMI		AMI	
		HDBSCAN	GCom	HDBSCAN	GCom
LaBSE	title	.729	.786	.371	.583
	title+text	.746	.887	.441	.837
	title+6 sent., filtered	.810	.905	.689	.861
mE5-large	title	.734	.857	.393	.774
	title+text	.786	.901	.583	.857
	title+6 sent., filtered	.815	.892	.698	.846
GTE	title	.735	.863	.399	.807
	title+text	.820	.901	.674	.860
	title+6 sent., filtered	.811	.899	.686	.862
BGE-M3	title	.737	.849	.407	.754
	title+text	.817	.900	.665	.858
	title+6 sent., filtered	.815	.885	.700	.819
nomic-v2	title	.736	.854	.400	.763
	title+text	.770	.902	.529	.861
	title+6 sent., filtered	.820	.907	.708	.869
Gemma-0.3B	title	.717	.803	.311	.700
	title+text	.775	.898	.545	.856
	title+6 sent., filtered	.772	.862	.631	.799
Qwen3-0.6B	title	.729	.824	.370	.759
	title+text	.779	.898	.563	.855
	title+6 sent., filtered	.817	.906	.701	.862
Qwen3-4B	title	.731	.796	.377	.743
	title+text	.776	.902	.550	.861
	title+6 sent., filtered	.820	.913	.706	.875
Qwen3-8B	title	.735	.858	.396	.794
	title+text	.769	.902	.530	.857
	title+6 sent., filtered	.821	.913	.709	.874

Figure 6: Summary of results on **Miranda**. NMI and AMI scores for each embedding model, in the three selected configurations: *title only*, *full text*, *configuration C* (bipartite clustering top-6 sentences after filtering). The best clustering configuration for each embedding is in bold.

F Failure examples

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2059	599	1748	370	0.566	0.755	0.996	0.964	0.722	0.847	0.441	0.785
Title + snippet	1885	213	1580	99	0.580	0.860	0.997	0.941	0.733	0.898	0.481	0.874
Title + text	1501	124	1191	32	0.612	0.902	0.997	0.941	0.759	0.921	0.564	0.905
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	777	286	591	133	0.688	0.841	0.947	0.955	0.797	0.894	0.697	0.864
Title + 6 sent., filtered	881	346	673	176	0.686	0.834	0.980	0.972	0.807	0.897	0.703	0.866
Title + all sent.	424	162	280	35	0.730	0.859	0.879	0.922	0.798	0.890	0.729	0.864
Title + all sent., filtered	480	202	313	69	0.742	0.861	0.936	0.930	0.828	0.895	0.763	0.869
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	480	140	341	54	0.732	0.941	0.912	0.885	0.812	0.912	0.745	0.896
Title + 6 sent., filtered	616	216	457	101	0.736	0.938	0.956	0.934	0.832	0.936	0.763	0.922
Title + all sent.	248	100	126	19	0.762	0.951	0.895	0.839	0.823	0.892	0.773	0.875
Title + all sent., filtered	306	127	176	39	0.777	0.954	0.932	0.869	0.847	0.910	0.800	0.894

Figure 7: Experimental Results **UA-RU News** dataset when BGE-M3 text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2019	189	1704	89	0.570	0.860	0.995	0.91	0.725	0.884	0.452	0.859
Title + snippet	2015	224	1722	114	0.572	0.867	0.996	0.933	0.726	0.899	0.457	0.875
Title + text	1501	124	1227	35	0.622	0.901	0.995	0.926	0.765	0.913	0.583	0.896
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	897	345	695	176	0.664	0.814	0.946	0.964	0.780	0.883	0.662	0.846
Title + 6 sent., filtered	995	413	781	230	0.671	0.809	0.981	0.974	0.797	0.884	0.678	0.845
Title + all sent.	440	164	300	35	0.717	0.858	0.868	0.917	0.785	0.887	0.712	0.860
Title + all sent., filtered	512	199	343	67	0.737	0.858	0.939	0.929	0.826	0.892	0.758	0.866
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	569	207	421	91	0.716	0.919	0.906	0.921	0.799	0.920	0.723	0.902
Title + 6 sent., filtered	699	282	525	150	0.730	0.916	0.966	0.956	0.831	0.935	0.757	0.919
Title + all sent.	282	104	154	18	0.752	0.951	0.891	0.834	0.815	0.889	0.760	0.871
Title + all sent., filtered	348	125	211	38	0.768	0.952	0.930	0.864	0.841	0.906	0.790	0.890

Figure 8: Experimental Results **UA-RU News** dataset when GTE-multilingual-base text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2019	502	1695	308	0.568	0.779	0.996	0.964	0.723	0.862	0.446	0.811
Title + snippet	1665	273	1354	135	0.601	0.850	0.997	0.964	0.750	0.904	0.534	0.877
Title + text	1541	124	1215	34	0.601	0.899	0.996	0.935	0.749	0.916	0.540	0.899
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	850	261	642	116	0.666	0.844	0.942	0.953	0.780	0.895	0.664	0.867
Title + 6 sent., filtered	983	307	753	152	0.666	0.849	0.982	0.966	0.793	0.904	0.671	0.876
Title + all sent.	451	177	283	51	0.712	0.851	0.885	0.921	0.789	0.885	0.713	0.857
Title + all sent., filtered	515	190	329	61	0.734	0.859	0.946	0.931	0.827	0.894	0.757	0.868
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	544	144	382	55	0.713	0.923	0.913	0.874	0.801	0.898	0.724	0.879
Title + 6 sent., filtered	701	207	519	98	0.722	0.930	0.962	0.923	0.825	0.926	0.747	0.911
Title + all sent.	270	115	138	24	0.747	0.932	0.896	0.813	0.815	0.869	0.759	0.847
Title + all sent., filtered	347	136	197	42	0.765	0.941	0.935	0.851	0.842	0.894	0.789	0.875

Figure 9: Experimental Results **UA-RU News** dataset when nomic-embed-text-v2-moe text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2008	248	1679	137	0.567	0.816	0.995	0.903	0.722	0.857	0.444	0.822
Title + snippet	1814	171	1486	74	0.580	0.852	0.996	0.914	0.733	0.882	0.483	0.855
Title + text	1497	80	1170	10	0.602	0.868	0.991	0.874	0.749	0.871	0.543	0.849
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	897	321	673	162	0.655	0.802	0.944	0.949	0.774	0.869	0.648	0.829
Title + 6 sent., filtered	1016	373	771	208	0.657	0.805	0.975	0.964	0.785	0.878	0.654	0.838
Title + all sent.	460	217	305	70	0.702	0.825	0.867	0.927	0.776	0.873	0.695	0.840
Title + all sent., filtered	553	241	366	89	0.721	0.830	0.937	0.933	0.815	0.878	0.739	0.846
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	571	180	419	84	0.692	0.897	0.892	0.873	0.779	0.885	0.692	0.862
Title + 6 sent., filtered	728	259	535	135	0.701	0.896	0.952	0.931	0.808	0.913	0.718	0.892
Title + all sent.	312	145	173	36	0.723	0.908	0.879	0.836	0.794	0.871	0.728	0.846
Title + all sent., filtered	393	164	236	60	0.739	0.918	0.929	0.855	0.823	0.885	0.760	0.864

Figure 10: Experimental Results **UA-RU News** dataset when multilingual-e5-large text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2724	407	2444	244	0.530	0.689	0.998	0.823	0.692	0.705	0.304	0.669
Title + snippet	2194	345	1943	201	0.567	0.754	0.995	0.874	0.722	0.810	0.439	0.753
Title + text	1844	102	1567	23	0.586	0.813	0.994	0.835	0.738	0.824	0.499	0.791
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	1319	710	1104	480	0.622	0.659	0.953	0.873	0.753	0.751	0.584	0.639
Title + 6 sent., filtered	1455	839	1225	571	0.616	0.666	0.976	0.912	0.755	0.770	0.573	0.655
Title + all sent.	814	455	629	260	0.642	0.681	0.869	0.854	0.739	0.758	0.612	0.668
Title + all sent., filtered	895	548	680	322	0.661	0.685	0.939	0.881	0.776	0.770	0.655	0.677
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	1016	344	856	193	0.640	0.698	0.898	0.749	0.747	0.722	0.613	0.648
Title + 6 sent., filtered	1179	556	974	351	0.649	0.717	0.960	0.846	0.774	0.776	0.635	0.696
Title + all sent.	559	342	402	110	0.663	0.722	0.858	0.757	0.748	0.739	0.647	0.659
Title + all sent., filtered	665	333	492	123	0.685	0.738	0.920	0.804	0.785	0.770	0.689	0.701

Figure 11: Experimental Results UA-RU News dataset when Gemma (with no additional prompt) text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2290	669	1972	442	0.548	0.692	0.993	0.892	0.706	0.779	0.380	0.687
Title + snippet	1998	372	1718	219	0.575	0.791	0.995	0.927	0.728	0.854	0.465	0.809
Title + text	1715	120	1419	33	0.593	0.860	0.996	0.884	0.743	0.872	0.518	0.847
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	1054	562	837	347	0.643	0.722	0.941	0.922	0.764	0.810	0.624	0.735
Title + 6 sent., filtered	1151	674	928	427	0.642	0.724	0.965	0.953	0.771	0.823	0.626	0.745
Title + all sent.	607	354	450	157	0.667	0.742	0.850	0.898	0.747	0.812	0.645	0.750
Title + all sent., filtered	602	432	434	203	0.689	0.751	0.893	0.929	0.778	0.830	0.685	0.769
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	698	284	538	149	0.679	0.820	0.900	0.851	0.774	0.835	0.675	0.794
Title + 6 sent., filtered	824	436	656	249	0.677	0.814	0.927	0.919	0.782	0.863	0.677	0.820
Title + all sent.	350	203	210	49	0.703	0.823	0.859	0.814	0.773	0.819	0.699	0.778
Title + all sent., filtered	394	265	251	92	0.726	0.840	0.901	0.880	0.804	0.860	0.736	0.823

Figure 12: Experimental Results UA-RU News dataset when Gemma (prompt="task: sentence similarity | query:") text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2394	187	2097	97	0.538	0.647	0.984	0.698	0.696	0.672	0.350	0.597
Title + snippet	2227	178	1943	86	0.555	0.719	0.991	0.758	0.712	0.738	0.406	0.682
Title + text	1798	138	1538	48	0.593	0.761	0.991	0.805	0.742	0.782	0.516	0.736
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	1124	1087	919	729	0.609	0.625	0.901	0.931	0.727	0.748	0.560	0.590
Title + 6 sent., filtered	1260	1329	1047	937	0.609	0.622	0.929	0.972	0.736	0.759	0.560	0.584
Title + all sent.	617	759	466	424	0.615	0.632	0.777	0.887	0.687	0.738	0.562	0.602
Title + all sent., filtered	704	937	529	568	0.629	0.640	0.831	0.935	0.716	0.760	0.589	0.619
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	709	738	558	458	0.618	0.662	0.812	0.871	0.701	0.752	0.573	0.637
Title + 6 sent., filtered	854	995	694	654	0.624	0.666	0.852	0.939	0.721	0.779	0.585	0.654
Title + all sent.	347	431	222	188	0.628	0.666	0.752	0.815	0.684	0.733	0.585	0.635
Title + all sent., filtered	406	651	270	361	0.642	0.681	0.79	0.891	0.708	0.772	0.609	0.668

Figure 13: Experimental Results UA-RU News dataset when Gemma (prompt="task: clustering | query:") text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2127	506	1800	325	0.560	0.752	0.997	0.921	0.717	0.828	0.422	0.766
Title + snippet	1905	223	1578	101	0.574	0.861	0.997	0.946	0.729	0.902	0.467	0.878
Title + text	1622	120	1291	35	0.593	0.895	0.996	0.922	0.743	0.908	0.521	0.890
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	945	256	710	117	0.654	0.833	0.956	0.937	0.777	0.882	0.648	0.849
Title + 6 sent., filtered	1096	311	857	165	0.651	0.829	0.981	0.948	0.782	0.884	0.644	0.851
Title + all sent.	465	183	307	57	0.701	0.844	0.854	0.908	0.770	0.875	0.689	0.845
Title + all sent., filtered	564	210	371	78	0.724	0.842	0.936	0.909	0.816	0.875	0.740	0.844
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	608	147	443	58	0.692	0.918	0.908	0.855	0.786	0.885	0.696	0.865
Title + 6 sent., filtered	763	205	573	103	0.708	0.915	0.966	0.905	0.818	0.910	0.731	0.891
Title + all sent.	318	117	173	23	0.733	0.920	0.892	0.834	0.805	0.875	0.742	0.853
Title + all sent., filtered	379	146	228	49	0.753	0.928	0.930	0.844	0.832	0.884	0.774	0.864

Figure 14: Experimental Results **UA-RU News** dataset when Qwen3-Embedding-0.6B text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2190	683	1869	459	0.556	0.737	0.997	0.953	0.714	0.831	0.409	0.759
Title + snippet	1672	342	1351	184	0.594	0.823	0.997	0.970	0.745	0.890	0.521	0.857
Title + text	1426	192	1104	72	0.613	0.884	0.996	0.967	0.759	0.924	0.569	0.905
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	893	296	668	140	0.670	0.825	0.958	0.954	0.789	0.885	0.673	0.852
Title + 6 sent., filtered	1042	367	812	198	0.664	0.820	0.983	0.964	0.793	0.886	0.667	0.850
Title + all sent.	463	198	302	61	0.702	0.841	0.869	0.920	0.777	0.879	0.696	0.849
Title + all sent., filtered	517	230	329	83	0.734	0.845	0.944	0.933	0.826	0.887	0.756	0.857
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	580	173	427	72	0.715	0.916	0.911	0.889	0.801	0.903	0.723	0.883
Title + 6 sent., filtered	731	237	552	116	0.722	0.925	0.963	0.932	0.825	0.929	0.747	0.912
Title + all sent.	290	134	148	31	0.744	0.932	0.897	0.876	0.813	0.903	0.755	0.885
Title + all sent., filtered	355	157	206	52	0.766	0.937	0.941	0.885	0.845	0.910	0.793	0.893

Figure 15: Experimental Results **UA-RU News** dataset when Qwen3-Embedding-4B text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	2051	460	1739	282	0.567	0.791	0.996	0.940	0.723	0.859	0.443	0.812
Title + snippet	1695	232	1380	107	0.591	0.857	0.996	0.953	0.742	0.902	0.513	0.878
Title + text	1331	150	1020	53	0.622	0.888	0.993	0.944	0.765	0.915	0.589	0.897
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	869	244	636	100	0.672	0.849	0.961	0.949	0.791	0.896	0.677	0.869
Title + 6 sent., filtered	1014	316	779	160	0.669	0.841	0.985	0.968	0.797	0.900	0.676	0.871
Title + all sent.	434	167	283	40	0.708	0.868	0.852	0.916	0.773	0.892	0.696	0.867
Title + all sent., filtered	505	201	319	64	0.736	0.852	0.938	0.924	0.825	0.887	0.756	0.859
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	562	145	406	57	0.708	0.922	0.907	0.877	0.795	0.899	0.715	0.880
Title + 6 sent., filtered	723	205	520	98	0.715	0.937	0.970	0.929	0.823	0.933	0.740	0.918
Title + all sent.	296	118	161	21	0.741	0.933	0.891	0.844	0.809	0.886	0.750	0.866
Title + all sent., filtered	345	139	198	44	0.761	0.945	0.935	0.873	0.839	0.908	0.785	0.891

Figure 16: Experimental Results **UA-RU News** dataset when Qwen3-Embedding-8B text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	7432	2972	6700	2264	0.586	0.749	0.994	0.981	0.737	0.849	0.407	0.754
Title + snippet	6388	1720	5705	1215	0.615	0.823	0.993	0.970	0.760	0.890	0.495	0.838
Title + text	4485	1205	3866	761	0.694	0.848	0.994	0.958	0.817	0.900	0.665	0.858
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	4674	2686	4132	2102	0.645	0.753	0.948	0.953	0.768	0.841	0.571	0.749
Title + 6 sent., filtered	5183	3061	4626	2426	0.644	0.759	0.976	0.984	0.776	0.857	0.569	0.767
Title + all sent.	2291	1497	1921	1064	0.727	0.793	0.870	0.902	0.792	0.844	0.686	0.775
Title + all sent., filtered	2663	1865	2244	1354	0.736	0.803	0.925	0.944	0.820	0.868	0.716	0.803
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	3014	1739	2680	1356	0.717	0.852	0.839	0.840	0.773	0.846	0.655	0.788
Title + 6 sent., filtered	3542	2288	3114	1763	0.732	0.854	0.919	0.934	0.815	0.892	0.700	0.841
Title + all sent.	1300	856	1122	566	0.805	0.897	0.796	0.783	0.800	0.836	0.736	0.790
Title + all sent., filtered	1715	1217	1461	874	0.818	0.904	0.865	0.850	0.841	0.876	0.779	0.835

Figure 17: Experimental Results **Miranda** dataset when BGE-M3 text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	7465	1388	6712	1003	0.583	0.815	0.993	0.918	0.735	0.863	0.399	0.807
Title + snippet	6215	1603	5579	1108	0.623	0.825	0.993	0.965	0.766	0.889	0.517	0.838
Title + text	4339	1165	3757	731	0.698	0.849	0.992	0.960	0.820	0.901	0.674	0.860
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	4972	1804	4407	1354	0.635	0.817	0.954	0.949	0.762	0.878	0.548	0.821
Title + 6 sent., filtered	5478	2030	4907	1559	0.635	0.814	0.981	0.966	0.771	0.884	0.549	0.826
Title + all sent.	2310	1100	1931	725	0.722	0.837	0.865	0.898	0.787	0.866	0.678	0.816
Title + all sent., filtered	2764	1359	2320	923	0.727	0.837	0.927	0.930	0.815	0.881	0.705	0.831
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	3092	1213	2741	923	0.705	0.901	0.840	0.822	0.767	0.860	0.642	0.817
Title + 6 sent., filtered	3707	1526	3215	1147	0.718	0.906	0.931	0.893	0.811	0.899	0.686	0.862
Title + all sent.	1271	659	1093	446	0.797	0.921	0.781	0.746	0.789	0.825	0.722	0.784
Title + all sent., filtered	1712	970	1444	689	0.803	0.928	0.855	0.826	0.828	0.874	0.761	0.838

Figure 18: Experimental Results **Miranda** dataset when GTE-multilingual-base text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	7497	2839	6738	2135	0.584	0.755	0.994	0.983	0.736	0.854	0.400	0.763
Title + snippet	5769	1774	5121	1232	0.636	0.815	0.992	0.978	0.775	0.889	0.548	0.834
Title + text	5842	1254	5101	798	0.628	0.847	0.994	0.965	0.770	0.902	0.529	0.861
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	4823	1952	4241	1470	0.642	0.804	0.959	0.955	0.769	0.873	0.564	0.811
Title + 6 sent., filtered	5273	2251	4699	1750	0.641	0.803	0.980	0.975	0.775	0.881	0.563	0.819
Title + all sent.	2261	1124	1873	730	0.733	0.838	0.878	0.912	0.799	0.874	0.697	0.824
Title + all sent., filtered	2688	1430	2258	976	0.735	0.843	0.926	0.943	0.819	0.890	0.715	0.843
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	2869	1327	2508	998	0.721	0.889	0.841	0.837	0.777	0.862	0.662	0.817
Title + 6 sent., filtered	3414	1685	2919	1237	0.735	0.895	0.927	0.919	0.820	0.907	0.708	0.869
Title + all sent.	1163	745	965	486	0.814	0.917	0.793	0.775	0.803	0.840	0.743	0.798
Title + all sent., filtered	1605	966	1316	674	0.820	0.924	0.859	0.831	0.839	0.875	0.777	0.838

Figure 19: Experimental Results **Miranda** dataset when nomic-embed-text-v2-moe text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	7498	2552	6697	1904	0.582	0.764	0.995	0.977	0.734	0.857	0.393	0.774
Title + snippet	6081	2080	5356	1480	0.620	0.803	0.996	0.985	0.764	0.885	0.510	0.824
Title + text	5194	1339	4485	832	0.650	0.840	0.995	0.971	0.786	0.901	0.583	0.857
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	4556	1995	3966	1485	0.648	0.794	0.955	0.946	0.772	0.863	0.577	0.796
Title + 6 sent., filtered	5097	2286	4487	1732	0.645	0.795	0.982	0.975	0.778	0.876	0.572	0.809
Title + all sent.	2189	1173	1803	790	0.724	0.818	0.865	0.898	0.788	0.856	0.681	0.799
Title + all sent., filtered	2601	1481	2153	1017	0.734	0.828	0.923	0.934	0.818	0.878	0.713	0.824
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	2803	1352	2454	1021	0.718	0.865	0.838	0.810	0.773	0.836	0.658	0.783
Title + 6 sent., filtered	3369	1775	2886	1300	0.726	0.874	0.928	0.910	0.815	0.892	0.698	0.846
Title + all sent.	1188	704	1007	456	0.778	0.901	0.770	0.749	0.774	0.818	0.703	0.772
Title + all sent., filtered	1629	995	1363	691	0.794	0.913	0.853	0.829	0.822	0.869	0.753	0.830

Figure 20: Experimental Results **Miranda** dataset when multilingual-e5-large text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	10706	544	10183	357	0.538	0.734	0.998	0.728	0.699	0.731	0.198	0.654
Title + snippet	8050	1969	7503	1541	0.589	0.790	0.993	0.922	0.739	0.851	0.409	0.782
Title + text	6721	1236	6070	850	0.610	0.838	0.991	0.932	0.755	0.882	0.478	0.835
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	6344	3553	5780	2880	0.606	0.633	0.970	0.843	0.746	0.723	0.474	0.537
Title + 6 sent., filtered	6810	4021	6257	3192	0.604	0.655	0.986	0.903	0.749	0.759	0.466	0.582
Title + all sent.	3628	2275	3194	1820	0.669	0.665	0.894	0.809	0.765	0.730	0.605	0.589
Title + all sent., filtered	4210	2523	3743	1903	0.673	0.684	0.945	0.853	0.786	0.759	0.622	0.624
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	5030	1722	4691	1419	0.645	0.685	0.887	0.637	0.747	0.660	0.550	0.550
Title + 6 sent., filtered	5533	2704	5107	2204	0.657	0.729	0.951	0.802	0.777	0.764	0.582	0.651
Title + all sent.	2467	1095	2291	557	0.717	0.769	0.797	0.631	0.755	0.693	0.645	0.595
Title + all sent., filtered	3151	1454	2876	989	0.736	0.783	0.887	0.755	0.804	0.769	0.696	0.687

Figure 21: Experimental Results **Miranda** dataset when Gemma (with no additional prompt) text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	8775	2309	8061	1787	0.561	0.73	0.994	0.892	0.717	0.803	0.311	0.700
Title + snippet	6557	2066	5945	1550	0.619	0.801	0.992	0.969	0.762	0.877	0.502	0.814
Title + text	5911	1232	5283	812	0.635	0.850	0.993	0.950	0.775	0.898	0.545	0.856
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	5152	2700	4603	2156	0.63	0.724	0.954	0.913	0.759	0.808	0.538	0.697
Title + 6 sent., filtered	5332	3214	4780	2564	0.631	0.737	0.967	0.971	0.763	0.838	0.539	0.733
Title + all sent.	2690	1494	2315	1107	0.693	0.753	0.862	0.862	0.768	0.804	0.639	0.718
Title + all sent., filtered	2862	1962	2474	1436	0.702	0.768	0.891	0.916	0.785	0.835	0.659	0.753
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	3583	1482	3291	1159	0.687	0.808	0.843	0.744	0.757	0.775	0.614	0.702
Title + 6 sent., filtered	3780	2247	3467	1741	0.692	0.830	0.873	0.897	0.772	0.862	0.631	0.799
Title + all sent.	1502	817	1365	475	0.751	0.845	0.756	0.680	0.754	0.754	0.671	0.687
Title + all sent., filtered	1755	1171	1586	837	0.768	0.865	0.802	0.802	0.785	0.832	0.705	0.779

Figure 22: Experimental Results **Miranda** dataset when Gemma (prompt="task: sentence similarity | query:") text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	8985	153	8294	39	0.557	0.714	0.992	0.663	0.714	0.688	0.297	0.618
Title + snippet	6829	747	6239	506	0.613	0.824	0.990	0.842	0.757	0.833	0.485	0.781
Title + text	6149	1211	5495	814	0.625	0.832	0.992	0.924	0.767	0.875	0.520	0.826
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	5186	4067	4661	3216	0.623	0.648	0.943	0.912	0.750	0.757	0.521	0.571
Title + 6 sent., filtered	5334	4716	4823	3753	0.628	0.660	0.957	0.976	0.758	0.787	0.532	0.602
Title + all sent.	2850	2294	2499	1717	0.676	0.677	0.839	0.842	0.749	0.750	0.607	0.613
Title + all sent., filtered	3128	3031	2750	2272	0.689	0.695	0.890	0.927	0.777	0.794	0.640	0.658
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	3558	2367	3351	1917	0.673	0.729	0.808	0.776	0.734	0.752	0.585	0.642
Title + 6 sent., filtered	3833	3335	3573	2624	0.680	0.749	0.855	0.924	0.758	0.827	0.608	0.722
Title + all sent.	1573	1165	1454	761	0.734	0.776	0.733	0.715	0.734	0.744	0.645	0.661
Title + all sent., filtered	1938	1835	1755	1363	0.751	0.795	0.805	0.847	0.778	0.820	0.689	0.745

Figure 23: Experimental Results **Miranda** dataset when Gemma (prompt="task: clustering | query:") text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	8083	1224	7387	892	0.576	0.795	0.993	0.854	0.729	0.824	0.370	0.759
Title + snippet	6399	1644	5703	1135	0.613	0.825	0.993	0.970	0.758	0.892	0.490	0.841
Title + text	5386	1297	4663	830	0.641	0.842	0.993	0.963	0.779	0.898	0.563	0.855
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	4632	1474	4070	1086	0.646	0.827	0.955	0.929	0.770	0.875	0.573	0.822
Title + 6 sent., filtered	5056	1711	4469	1289	0.646	0.834	0.983	0.951	0.780	0.889	0.576	0.839
Title + all sent.	2232	922	1855	590	0.726	0.835	0.863	0.876	0.789	0.855	0.682	0.804
Title + all sent., filtered	2563	1200	2138	795	0.741	0.839	0.925	0.916	0.823	0.876	0.723	0.827
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	2964	1676	2616	1277	0.714	0.860	0.845	0.872	0.774	0.866	0.655	0.815
Title + 6 sent., filtered	3497	2094	3053	1565	0.729	0.865	0.928	0.951	0.817	0.906	0.701	0.862
Title + all sent.	1190	587	1035	380	0.800	0.909	0.779	0.678	0.790	0.777	0.725	0.728
Title + all sent., filtered	1588	782	1347	525	0.822	0.926	0.857	0.769	0.839	0.840	0.779	0.799

Figure 24: Experimental Results **Miranda** dataset when Qwen3-Embedding-0.6B text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	7967	597	7274	422	0.578	0.814	0.994	0.780	0.731	0.796	0.377	0.743
Title + snippet	6404	1559	5687	1045	0.612	0.823	0.995	0.967	0.757	0.889	0.487	0.838
Title + text	5492	1208	4738	752	0.636	0.849	0.995	0.962	0.776	0.902	0.550	0.861
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	4579	2267	4001	1721	0.651	0.787	0.959	0.960	0.775	0.865	0.584	0.793
Title + 6 sent., filtered	5051	2568	4470	2012	0.649	0.783	0.982	0.980	0.782	0.870	0.582	0.797
Title + all sent.	2199	1513	1828	1010	0.729	0.811	0.859	0.929	0.789	0.866	0.684	0.805
Title + all sent., filtered	2543	1889	2127	1329	0.741	0.814	0.922	0.970	0.822	0.885	0.722	0.828
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	2988	1552	2661	1168	0.716	0.878	0.848	0.873	0.776	0.875	0.658	0.829
Title + 6 sent., filtered	3530	1941	3081	1448	0.730	0.883	0.935	0.946	0.820	0.913	0.706	0.875
Title + all sent.	1205	903	1037	613	0.810	0.905	0.786	0.832	0.798	0.867	0.736	0.827
Title + all sent., filtered	1634	1348	1379	956	0.826	0.910	0.863	0.912	0.844	0.911	0.784	0.878

Figure 25: Experimental Results **Miranda** dataset when Qwen3-Embedding-4B text encoder model used.

	# clusters		# singletons		completeness		homogeneity		NMI		AMI	
	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom	HDBSCAN	GCom
Configuration A: Single-step clustering, one vector per article												
Title	7691	1749	6997	1299	0.583	0.800	0.994	0.925	0.735	0.858	0.396	0.794
Title + snippet	6299	1950	5594	1390	0.614	0.812	0.994	0.977	0.759	0.887	0.494	0.830
Title + text	5659	1479	4891	983	0.627	0.839	0.995	0.975	0.769	0.902	0.530	0.857
Configuration B: Two-level clustering, k sentence representatives, projection on sentence communities												
Title + 6 sent.	4499	2248	3892	1712	0.652	0.790	0.960	0.964	0.776	0.868	0.586	0.799
Title + 6 sent., filtered	5012	2580	4412	2014	0.649	0.788	0.985	0.984	0.783	0.876	0.584	0.805
Title + all sent.	2192	1402	1823	931	0.735	0.814	0.871	0.921	0.797	0.864	0.696	0.805
Title + all sent., filtered	2602	1828	2169	1280	0.742	0.822	0.933	0.969	0.826	0.890	0.727	0.835
Configuration C: Two-level clustering, k sentence representatives, bipartite community detection												
Title + 6 sent.	2910	1525	2554	1154	0.723	0.878	0.853	0.864	0.782	0.871	0.668	0.824
Title + 6 sent., filtered	3399	1953	2921	1456	0.732	0.882	0.936	0.946	0.821	0.913	0.709	0.874
Title + all sent.	1162	849	980	574	0.806	0.906	0.782	0.819	0.794	0.861	0.731	0.821
Title + all sent., filtered	1568	1260	1309	891	0.822	0.912	0.855	0.896	0.838	0.904	0.778	0.870

Figure 26: Experimental Results **Miranda** dataset when Qwen3-Embedding-8B text encoder model used.

Document A	Document B
<i>European, Salvini: “Vote League to stop bombers like Macron”</i> “I call for a vote against the war, a vote to halt dangerous bombers such as Macron. Today and tomorrow, Italians can put an end to the winds of war.”	<i>UK’s Farage: West provoked Putin to invade Ukraine</i> Nigel Farage, leader of the United Kingdom’s anti-immigration Reform UK party, came under strong criticism after claiming that the West had provoked Russia’s invasion of Ukraine. Speaking to the BBC, Farage said, “we’ve provoked this war,” while referring to the conflict in Ukraine.

Case 1: Incorrect merging of different quotes by different entities

Configuration: *GCom*, sentence-level, with filtering on *LaBSE* embeddings.

Commentary: Both documents mention opposition politicians who express opposition to the war in Ukraine. They focus on different political actors and use different framing.

Document A	Document B
<i>Russia imprisons US soldier Gordon Black as it suspends prisoner swap deal</i> The sentence adds to the number of American citizens detained in Russia, as Moscow says it is awaiting a response concerning prisoner-exchange proposals. A Russian court sentenced a U.S. soldier to prison while Moscow said it was waiting for Washington’s response.	<i>Russian-American woman’s treason trial opens as tensions rise between Washington and Moscow</i> The trial of a Russian-American citizen accused of treason by Russia began as tensions increased between Washington and Moscow, including amid the arrests of two American journalists. The proceedings are being held behind closed doors in Yekaterinburg.

Case 2: Incorrect merging of different events.

Configuration: *GCom* on *title+text* using *Qwen3-8B* embeddings.

Commentary: The documents on the surface exhibit strong topical overlap (both refer to a penal court issue) while describing very different events regarding different individuals (for A, a court sentence, while for B, the beginning of a trial).

Document A	Document B
<i>Why a drone war in Asia would be different from the one in Ukraine</i> Twenty years ago, drones were seldom used in warfare. In 2003, during the first year of the Iraq war, the United States had 163 drones, accounting for 1% of its total aircraft fleet. Today, they have become dominant on the battlefield and have also expanded across the world.	<i>Romania is taking war drones from Germany</i> German defense company Quantum Systems, which manufactures unmanned aerial vehicles, announced that it had received a contract from Romania’s Ministry of Defense to provide Vector reconnaissance unmanned aerial vehicle systems. The same model is also used by Ukraine for surveillance of Russia.

Case 3: Incorrect merging of different types of articles.

Configuration: *GCom* on *title+text* using *Qwen3-8B* embeddings.

Commentary: Both documents contain mention of drone technology and mention Ukraine, but vastly differ in their content, focus and framing. A is an opinion article concerning future warfare, while B is a report about a drone purchase agreement.

Table 6: Qualitative examples of failure cases in clustering. In Cases 1–3, the pair of documents was clustered together whereas the ground-truth did not.

Document A	Document B
<i>Sen. Lindsey Graham coughs up the real reason why US is providing unlimited funding of Russia-Ukraine war</i> U.S. Senator Lindsey Graham (R-South Carolina) has become the first member of the Washington establishment to openly discuss what he describes as the real reason behind U.S. funding of the war between Ukraine and Russia, which he characterizes as a conflict between the U.S./NATO and Russia. Ukraine, he argues, is merely NATO’s proxy.	<i>In order not to give resources to Russia and China: Senator Graham explained why the United States would sponsor Ukraine</i> Washington should continue supporting Kyiv because the war could bring economic benefits to the United States and Western countries. U.S. Senator Lindsey Graham expressed this view on CBS News. According to him, if Ukraine wins, the West would gain “the best business partner”, and at the....

Case 4: Incorrect splitting of the same cluster

Configuration: *GCom*, sentence-level (bipartite on top-6 sentences) using *LaBSE* embeddings.

Commentary: Variation of the same article mentioning very similar information using different tone and framing. Both documents mention the same speech of the same entity, yet because of slight variations in phrasing the documents are split into separate clusters.

Table 7: Example of a failure case in clustering where the same cluster is split into two separate clusters.