FANATIC: FAst Noise-Aware TopIc Clustering

Ari Silburt, Anja Subasic, Evan Thompson, Carmeline Dsilva, Tarec Fares
Bloomberg, New York, New York
{asilburt, asubasic, ethompson64, cdsilva3, tfares1}@bloomberg.net

Abstract

Extracting salient topics from a collection of documents can be a challenging task when a) the amount of data is large, b) the number of topics is not known *a priori*, and/or c) "topic noise" is present. We define "topic noise" as the collection of documents that are irrelevant to any coherent topic and should be filtered out. By design, most clustering algorithms (e.g., k-means, hierarchical clustering) assign all input documents to one of the available clusters, guaranteeing any topic noise to propagate into the result. To address these challenges, we present a novel algorithm, FANATIC, that efficiently distinguishes genuine topic documents from those that are topic noise. We also introduce a new Reddit dataset to showcase FANATIC as it contains short, noisy data that is difficult to cluster using most clustering algorithms. We find that FANATIC clusters 500k Reddit titles (of which 20% are topic noise) in 2 minutes and achieves an AMI score of 0.59, in contrast with hdbscan (McInnes et al., 2017), a popular algorithm suited for this type of task, which requires over 7 hours and achieves an AMI of 0.03. Finally, we test FANATIC against a Twitter dataset and find again that it outperforms the other algorithms with an AMI score of 0.60. We make our code[1] and data publicly available.

1 Introduction

Every minute, millions of social media data such as Reddit comments, Tweets, Facebook comments, and other content are posted online (Marr, 2018). A cornucopia of value resides in this online information including product feedback, political beliefs, news, trending topics, and social interactions. However, these topics are generally needles in a haystack of topic noise and require suitable algorithms for extracting them.

[1] https://github.com/bloomberg/fast-noise-aware-topic-clustering

Coherent Topic subreddit: /r/Hair
How do I get this hairstyle?
No better feeling than freshly bleached roots!
Some fun Hair I did! (Haircut is not my work)
Is this hair possible for me?
Topic Noise subreddit: /r/TheSimpsons
The noble spirit embiggens the smallest man.
Surly Says: Don't Get Caught.. OR ELSE!
Guys, meet Lisa
And That's The End Of That Chapter

Table 1: Sample Reddit Post Titles From Topically Coherent (/r/Hair) and Noise (/r/TheSimpsons) subreddits

The ability to group short-text documents into topics is an increasingly relevant problem, yet few algorithms are effective because:

- the large numbers of documents can become computationally prohibitive;
- the number of topics is not known *a priori*;
- a large fraction of documents may be topically irrelevant or idiosyncratic, and should not be assigned to any topic. We henceforth refer to this phenomenon as "topic noise".

For social media data, clustering based methods are often favoured over more traditional topic models (Chinnov et al., 2015) like LDA (Blei et al., 2001), however, even within the clustering domain many algorithms struggle. For example, the standard k-means algorithm requires choosing the number of clusters ahead of time or finding the optimal number (which is an NP-hard problem, Mahajan et al., 2009). Therefore, time and/or compute restrictions make k-means infeasible for large datasets. Agglomerative clustering methods do not require specifying the number of clusters, but generally scale poorly with the number of documents, with runtimes of $O(n^2 \log n)$ (Gilpin et al., 2013).

Other clustering algorithms better suited to this task are gmeans (Hamerly and Elkan, 2003) and dpmeans (Kulis and Jordan, 2012); instead of

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 650–663
November 7–11, 2021.

specifying the number of clusters one needs only to specify criteria for adding or splitting clusters. gmeans starts with a single cluster and keeps splitting until the child clusters are less Gaussian than their parent. dpmeans specifies a distance length λ and creates new clusters when documents are greater than λ from any existing cluster (See Algorithm 1 from Kulis and Jordan, 2012).

However, most clustering algorithms, including those mentioned above, struggle with topic noise since every input document must be assigned to a cluster. As an example, for the Reddit[2] titles shown in Table 1, only one cluster should be produced - the /r/Hair subreddit captures a single, coherent topic while /r/TheSimpsons subreddit titles are irrelevant to any single topic and should be filtered out as noise.

One option for handling topic noise is to apply a pre-processing step to filter it out before clustering (e.g., Godfrey et al., 2014), but without proper care one can accidentally remove informative "hard-to-classify" documents and/or fail to remove all of the topic noise (Guyon et al., 1996). In addition, each dataset will have its own noise profile warranting a new, detailed analysis per dataset. Some works (e.g., Curiskis et al., 2020) have restricted their analyses to clustering on tight coherent topics with zero topic noise; however, such studies are unlikely to generalize to datasets where topic noise is present.

Instead, a more desirable approach is to add filtering capabilities directly into the clustering algorithm so that clustering and topic noise filtering can be handled together. A key algorithm designed for this purpose is hdbscan (McInnes et al., 2017), which we use as a benchmark for our new algorithm, FANATIC, in Section 5.

A significant challenge in developing clustering algorithms for social media data is acquiring reliable ground truth labels. In particular, obtained labels must reliably distinguish documents from coherent topics and those that are topic noise. However, a common practice when using Twitter data for example is to use the hashtag(s) as the ground truth label (e.g., Benevenuto et al., 2010; Rosa et al., 2011; Curiskis et al., 2020). Since many Twitter hashtags are generic (e.g., #TuesdayThoughts), tweets containing such hashtags can have very little in common with one another (Bruns and Burgess, 2011; Ferragina et al., 2015). For the Reddit domain, Table 1 illustrates how titles from the /r/Hair subreddit encapsulate a coherent topic while titles from /r/TheSimpsons subreddit are unrelated. Many studies (e.g., Rosa et al., 2011; Conover et al., 2011; Park and Conway, 2018; Curiskis et al., 2020) do not assess the topical coherency of the hashtag/subreddit used as the ground truth label, raising questions about how coherent the associated content is. In addition, collisions between nearby labels (e.g., #photooftheday and #picoftheday) will also downgrade performance since, from a metrics perspective, these identical topics would be considered separate.

Our contributions in this work are as follows:

- FANATIC, a clustering algorithm that is fast, does not require specifying the number of clusters *a priori*, and is robust to topic noise;
- a new Reddit-based dataset that reliably distinguishes documents from coherent topics and those that are topic noise;
- evaluation of FANATIC against current cluster algorithms suited to social media data: hdbscan, gmeans, dpmeans, and LDA.

2 FANATIC algorithm

2.1 Brief overview of dpmeans

FANATIC is built upon the original dpmeans algorithm (Kulis and Jordan, 2012), which works by specifying a cluster diameter, λ. The algorithm is initialized by creating a single cluster whose center is the mean of all of the documents. It then iterates over the documents and assigns each to either a) the nearest cluster provided the distance is less than λ, or b) creates a new cluster with the document's location as the cluster center. This process repeats until convergence.

FANATIC enhances dpmeans to ensure robustness to topic noise through several modifications to the original algorithm. A description of the modifications and associated parameters are described in the subsections below. The distance function, $\mathcal{D}$, is either cosine or Euclidean and convergence is achieved when the document-weighted average change in cluster centers falls below a specified threshold. The complete algorithm is outlined in Algorithm 1.

2.2 Minimum Token Probability

For text-based clustering it is typical to cluster on word embeddings, yet embeddings of rare words are ineffective and often clump together (e.g., Gong

[2] https://www.reddit.com/

Algorithm 1 FANATIC

INPUT: $d_1, .., d_n \in D$: set of n documents
PARAMETERS: λ: cluster size
L: token probability threshold
$\mathcal{D}$: distance function (cosine or Euclidean)
N_C: maximum number of clusters
S_C: minimum cluster size
M_R: number of cluster-merge rounds
M_d: merge distance between two clusters
OUTPUT: $y_1, .., y_n$: cluster assignments for each document, C: total number of clusters

1: Initialize: C=1, μ=$\{\mu_1\}$ s.t. μ_1= global mean
2: **while** True **do**
3: Randomly shuffle documents
4: Clear assignments: $y_1, .., y_n$ = Ø
5: **for all** $d \in D$ **do**
6: Compute the set of clusters, c, where $\mathcal{D}(\mu_c, d) < \lambda$ and $P_{c,d} \geq L$
7: **if** c is not empty **then**
8: $y_i = \text{argmin}_c(\mathcal{D}(\mu_c, d))$
9: **else if** $C < N_C$ **then**
10: Create new cluster: $C = C + 1$, $y_i = C$ $\mu_C = d$, add μ_C to μ
11: **end if**
12: **end for**
13: update cluster centers
14: Check for convergence (if so, break 2)
15: Merge clusters using M_R, M_d
16: **end while**
17: Remove clusters $< S_C$, reassign documents

et al., 2018). Thus, without proper care, disparate content can cluster together simply because they contain rare words. Social media data is particularly rife with rare words due to misspellings, abbreviations, acronyms, special characters, etc. (Chinnov et al., 2015; Curiskis et al., 2020).

Therefore, in addition to distance requirement λ for adding a document to a cluster, we add an additional token-based requirement that a document's tokens must be "sufficiently close" (defined in Equation 3) to the cluster's tokens. This feature encodes the intuition that, not only do we want to group documents that are close in embedding space, but additionally we want their raw tokens to be similar as well. This can significantly improve the purity of clusters as their formation no longer relies solely on the quality of the embedding space.

First we define $P_{c,t}$ to be the token probability for token t in cluster c,

$$P_{c,t} = \frac{\sum_{d \in D_c} \sum_{s \in T_d} \mathbb{1}_{s=t}}{\sum_{d \in D_c} |T_d|} \quad (1)$$

where D_c is the set of documents in cluster c, and T_d is the set of tokens in document d.

Defining $T_{c,d}$ as the set of common tokens between the documents in cluster c and a new document d, we then calculate the token probability of document d with respect to cluster c by summing the individual token probabilities of cluster c for each token in $T_{c,d}$, normalized by the total number of tokens in document d:

$$P_{c,d} = \sum_{t \in T_{c,d}} \frac{P_{c,t}}{|T_d|} \quad (2)$$

Finally, document d is only added to cluster c if

$$P_{c,d} \geq L \quad (3)$$

where $L \in [0, 1]$, the token probability threshold, is a tunable parameter. The token probabilities of a cluster, $P_{c,t}$, are re-calculated every time a new cluster is created or the cluster center is updated. Equation 3 is used during step 6 of Algorithm 1.

2.3 Cluster Merging

While iterating over the data, cluster centers can gradually move toward higher density space and find themselves within λ of other clusters. This can result in similar and/or duplicate clusters with arbitrary decision boundaries. Performance can be improved by merging such overlapping clusters.

Cluster merging proceeds in rounds, where the number of rounds, M_R, is a tunable parameter. At a high-level, each round commences by first finding all pairwise distances between clusters. Next, cluster pairs are greedily chosen in order of ascending pairwise distance. If the distance between the two clusters is less than λM_d, where $M_d \in [0, 1]$ is a tunable parameter, the clusters will be merged. A cluster may only be merged once per round, as allowing for multiple can result in merges cascading into a single (or several) large, ambiguous clusters. When a merge occurs the new cluster center becomes the document-weighted average of the two child clusters, while the cluster diameter remains fixed at λ. Cluster merging occurs during step 15 of Algorithm 1.

2.4 Post-Cluster Filtering of Small Clusters and Document Reassignment

After clustering is complete we filter out clusters that have fewer than S_C documents, a tunable parameter, under the intuition that they likely encap-

sulate highly specific and/or idiosyncratic topics. To ensure that we do not lose valuable documents during this filtering process, we perform a final assignment step where documents from filtered clusters can be re-assigned to a remaining cluster if the criteria in step 6 of Algorithm 1 is met. This serves as an additional method of cleaning up topic noise by removing small clusters, but taking relevant, topic-coherent documents out of them and adding them back into the large clusters they belong to. Cluster filtering and reassignment is done during step 17 of Algorithm 1.

2.5 Limiting Number of Clusters During Clustering

To accommodate the fact that, *a priori*, the true number of clusters in the dataset is unknown, we introduce the tunable parameter N_C, an upper bound on the total number of clusters. It allows for more flexibility than algorithms where number of clusters is fixed (e.g., `k-means`). Specifically, N_C:

- allows documents to be classified as topic noise/outliers since if a document doesn't belong to an existing cluster but N_C is reached, the document remains unassigned. Without N_C, a new cluster would always be created;
- acts as a form of regularization, forcing fewer clusters to find an optimal configuration;
- speeds up document assignment. Once N_C is reached the remaining documents can be assigned in parallel.

3 Data

We evaluate our algorithm's performance on the Pushshift Reddit dataset (Baumgartner et al., 2020), as it is publicly available and suitable for clustering. Specifically, the Reddit platform is organized into categories, or subreddits, which generally focus on a single topic, have a title, and contain a large number of user posts. We use the titles of posts from selected subreddits as input documents for clustering, while the cluster labels are derived from the subreddit via an annotation task described below.

3.1 Annotation Task to Extract Coherent Subreddits

As mentioned in Section 1, obtaining ground truth labels requires care due the fact that many subreddits are, topically speaking, very general (e.g., /r/Showerthoughts), and especially so when considering only the title of the post without additional context (see Table 1). Here we define an annotation task with the goal of identifying those subreddits which encapsulate a single "coherent" topic and those which do not, which we label as "noise".

3.1.1 Topic Definition

We acknowledge upfront that many valid definitions of "topic" exist, and future users are encouraged to try others as `FANATIC` is not tied to a particular one. In this work we follow Guille et al. (2013) who define a topic as "a coherent set of semantically related terms that express a single argument". We apply this definition to our annotation task (and downstream clustering) such that a topic must be characterized by a central noun (e.g., "sports", "cooking", "fitness"), and cannot be defined by a central adjective (e.g., "happy", "cute", "interesting").

It's possible that some of the subreddits we assign as "noise" are in fact coherent topics when viewed holistically on `www.reddit.com` (e.g., "/r/TheSimpsons"). However, importantly, in this work we only considered the title of each post and disregarded all other content (pictures, text body, comments, etc.). Therefore, since our dataset has been significantly mutated from the original content, it's possible that some annotation labels may deviate from human expectation.

3.1.2 Task Design

For this annotation task, we randomly sample 1000 subreddits. From each subreddit, we randomly sample forty posts and have six annotators evaluate random subsets of twenty posts from the selected forty posts. When presenting posts to the annotator we omit the subreddit label to avoid biasing the annotator (e.g., /r/Showerthoughts gives context to otherwise unrelated posts).

We ask annotators to evaluate topic coherency by answering two questions: a) Do the majority of the titles (sampled from a single subreddit) represent a coherent topic and, if so, b) provide a short summary for the topic.

We crowdsource our annotations using a leading commercial crowdsourcing platform where anonymized annotators are sampled randomly from around the world. We utilize quality control features which exclude low performing contributors on golden test questions, as well other quality control measures described in more detail in Appendix A.1.2.

3.1.3 Extracting Annotation Label

We limit the final set of subreddits to only those which were annotated consistently across all six annotators to increase reliability of our results. We define subreddits in which all annotators answered question a) with "yes" as "coherent" subreddits, and those in which all annotators answered question a) with "no" as "noise" subreddits. As an additional quality control step, we examine the annotator-provided summaries and manually filter out any subreddits whose summaries did not unanimously describe a single semantic topic.

Although we now have high confidence as to which subreddits encapsulate coherent topics and which are topical noise, we still have not accounted for the fact that subreddits can overlap in content, and a particular reddit post could (and often does) belong to many subreddits. It's important to account for this overlap when assigning cluster labels to avoid unfair penalization in downstream metrics.

Therefore, as the final filtering step, through a combination of TF-IDF analysis and manual vetting, we remove subreddits which are semantically similar (e.g., /r/Hair and /r/curlyhair), and always remove the smaller of the two subreddits.

3.1.4 Final Dataset

After the aforementioned annotation procedure, our dataset is finalized to 25 coherent subreddits and 67 noise ones, which are listed in Appendix A.1.1. For the remainder of this work we restrict to titles from these subreddits.

4 Methods

4.1 Preprocessing and Embeddings

All Reddit titles are preprocessed by a) normalizing urls, numbers, @mentions, emoticons, dollar amounts, emails and phone numbers, b) lowercasing, c) tokenizing and d) filtering out stopwords using NLTK's[3] standard stopword list.

Using a trained Word2Vec model (Mikolov et al., 2013), each title's tokens are embedded and averaged into a single vector. Although more sophisticated techniques exist for combining token-level embeddings into document-level embeddings (e.g., Arora et al., 2017; De Boom et al., 2016), these methods generally depend on term-frequency statistics which can be unreliable in noisy social media data (spelling mistakes, slang, etc.). Furthermore, a simple average often performs competitively on short texts (Wieting et al., 2016). The Word2Vec model was trained via `gensim` (Řehůřek and Sojka, 2010) on the RS_2017-08.bz2 - RS_2017-11.bz2 data files using a standard embedding size of 300 and window size of 5. We find downstream results insensitive to changes in Word2Vec hyperparameters, likely due to the short nature of each Reddit title.

4.1.1 Alternative Featurizations

Since `FANATIC` only relies on embeddings and tokens for clustering, future users are encouraged to featurize however they wish provided a static embedding vector and token set can be generated per document. For example, one could switch to use contextual embeddings (e.g., Reimers and Gurevych, 2019) instead of Word2Vec, and the code[4] has been specifically modularized to accommodate alternative preprocessings. Our choice of Word2Vec stemmed from a need for a strong baseline embedding model to showcase `FANATIC`'s potential. `FANATIC` should still perform regardless of featurization strategy.

4.2 Clustering Algorithms

The documents are then clustered using each of the following clustering algorithms until convergence:

- `FANATIC` (see Section 2)
- `dpmeans` (Kulis and Jordan, 2012)
- `gmeans` (Hamerly and Elkan, 2003)
- `hdbscan` (McInnes et al., 2017)
- `LDA` (Blei et al., 2001)

For `gmeans`, `dpmeans` and `LDA` we add an additional hyperparameter to filter out clusters smaller than S_c after the algorithm completes (`FANATIC` and `hdbscan` already have this feature.). Without this added feature `gmeans`, `dpmeans` and `LDA` would have no opportunity to filter out noise. We emphasize that when $S_c = 0$, this added feature is disabled and the algorithms return to their original implementations. If this scenario is preferred it should be selected during hyperparameter tuning.

4.3 Evaluation

4.3.1 Labeling Noise Documents

Unlike the supervised classification domain where standard metrics like precision, recall, and f1 are reliable measures of performance, there are no equivalent one-size-fits-all metrics for the clustering domain (Romano et al., 2016). This is especially

[3] https://www.nltk.org/

[4] https://github.com/bloomberg/fast-noise-aware-topic-clustering

true when considering topic noise, where Amigó et al. (2009) show that almost all clustering metrics fail the "rag bag" scenario[5], which occurs when the data contains a collection of disparate items that should not be grouped with the other items (think "miscellaneous", "other", or in our case, "topic noise").

To best handle topic noise in this work we assign the same `NOISE` label to all Reddit titles from "noise" subreddits. From a metrics perspective, this consolidates the rag bag of noise documents into a single cluster label, encouraging them to be grouped together. This is an ideal labeling scheme for filtering topic noise, however, as we will see in Section 5.1, it can also encourage disparate `NOISE` content to group together in clusters since they share the same label, which is not ideal.

An alternative noise labeling scheme could be to assign a unique label to each noise document; however this would dramatically increase the number of labels and result in extreme label imbalances, which is very challenging for cluster metrics to handle (e.g., de Souto et al., 2012).

4.3.2 Performance Metrics

To select "best" runs and measure how well similarly-labeled documents are grouped together, we use the well-established Adjusted-Mutual Information, or `AMI` (Vinh et al., 2010). We select `AMI` because its baseline is a) adjusted for chance, b) robust to changes in the number of clusters and/or documents (Vinh et al., 2010; Meilă, 2007), and c) fast to compute. Other metrics such as V-measure (Rosenberg and Hirschberg, 2007), Fowlkes-Mallows (Fowlkes and Mallows, 1983) and B-Cubed (Bagga and Baldwin, 1998) do not have such properties (Meilă, 2007; Gösgens et al., 2019; scikit-learn developers, 2020).

For each run we also measure:

- pseudo-precision, P^*, which tracks the contamination of topic noise in clusters.
- pseudo-recall, R^*, which tracks how well documents from coherent topics are retained in clusters vs. filtered out as noise.

These are calculated as:

$$P^* = \frac{tp^*}{tp^* + fp^*}, \quad R^* = \frac{tp^*}{tp^* + fn^*} \tag{4}$$

[5]Amigó et al. (2009) mention that B-Cubed (Bagga and Baldwin, 1998) is robust to the rag-bag scenario. However it couldn't be used in this work since it's not robust to changes in number and size of clusters (Gösgens et al., 2019) and scales as O(n^2) (Bagga and Baldwin, 1998)), taking hours for a single B-cubed calculation when testing.

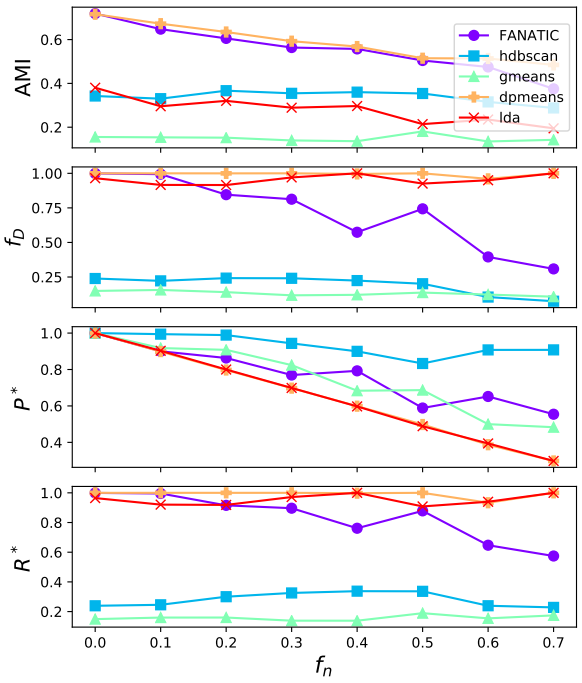

Figure 1: Noise fraction, f_n, vs. performance as measured by `AMI` (top row), the fraction of documents in clusters, f_D (second row), pseudo-precision, P^*, and pseudo-recall, R^* (Equation 4, bottom two rows).

Where tp^* is the set of documents from coherent topics that ended up in a cluster, fp^* is the set of noise documents that ended up in a cluster, and fn^* is the set of documents from coherent topics that did not end up in any cluster.

These are pseudo values since they only track whether a document ended up in *any* cluster vs. the *correct* cluster. However, since topic noise should not end up in *any* cluster, these metrics allow us to track the contamination of topic noise in clusters and determine how robust each clustering algorithm is at filtering it. A lower P^* implies that more noise documents are contaminating clusters, while a lower R^* implies that the more documents from coherent topics are being excluded from clusters.

5 Experiments and Results

5.1 Amount of Noise vs. Performance

In our first experiment we fix the number of documents, N_D, to 50k and vary the fraction of documents that are topic noise, f_n. The two questions we want to answer are how well each algorithm:

- groups similarly-labeled documents together;
- filters topic noise.

The first question is answered via the `AMI` score, while the second is answered via the P^* and R^*

scores, which (as mentioned in Section 4.3.2) monitor how an algorithm filters noise while retaining valid content in clusters.

At each (algorithm, f_n) combination we run 250 experiments randomly sampling over the algorithm's hyperparameters, and select the best run as the highest AMI score. Best results for each (algorithm, f_n) combination are displayed in Figure 1. See Appendix A.1.3 for additional experimental details.

5.1.1 FANATIC vs. dpmeans

The top panel of Figure 1 shows that FANATIC and dpmeans both have the highest AMI scores, indicating equal ability to group similarly-labeled documents together. We emphasize that the AMI score *includes* the grouping of noise documents as they all share the same NOISE label. As f_n increases, this grouping of topic noise becomes an increasingly dominant component of the overall AMI score (e.g., at $f_n = 0.5$, 50% of the clusterable content is topic noise).

The final three rows of Figure 1 show how, although dpmeans and FANATIC have equal AMI scores, FANATIC is superior at filtering topic noise from clusters for two key reasons:

- For all experiments the pseudo-precision, P^* (third row), is noticeably higher for FANATIC while still maintaining high pseudo-recall, R^* (last row). This indicates that FANATIC does a better job of filtering noise while keeping valid documents in clusters. In contrast dpmeans has the lowest P^* of any algorithm indicating poor ability to filter out noise.
- The fraction of documents clustered, f_D (second row), for dpmeans is approximately 1 regardless of the amount of noise present, f_n. This means that, although dpmeans can effectively group noise documents together (it has a high AMI score), this noise is contaminating clusters instead of being filtered. In contrast, for FANATIC f_D is proportional to f_n, illustrating how it filters more documents when more noise is present.

These findings are qualitatively highlighted in Tables 2 and 3 for the best performing $f_n = 0.4$ runs for FANATIC and dpmeans, respectively, and show eight randomly sampled documents from the cluster with the most "Hair" subreddit mentions. As can be seen, although the dpmeans cluster contains roughly equal number of NOISE and "Hair" labels (yielding a good AMI score), the cluster itself carries little topical coherency. In contrast, the FANATIC cluster clearly shows a valid "Hair" topic, and even the contamination (e.g., "giveaways" label) contains relevant content.

Label	Text
give-aways	Win $500 in Hair Essentials (11/22/2017)US
Hair	9 Best Fall Hair Color Ideas for 2017
Hair	how to get free from dreads with short hair
Hair	Optimal hair length: how long is too long?
Hair	How to find hair vendor? Contact me!
Hair	Advice on lightening dyed hair?
Hair	Smooth hair ponytail makes me look bald!
Hair	Essential Hair Growth – Hair Bloom

Table 2: FANATIC: Eight randomly sampled documents from the cluster with the most "Hair" documents for the best performing $f_n = 0.4$ run.

Label	Text
NOISE	Is a snake a neck or a tail?
Hair	ASAP Hair Dye Help!
Dentistry	Hawley retainer uneven bite
NOISE	Saltwater Fish Tank! Should i get a reef?
Hair	Any latinas/tanned girls with pastel pink hair?
NOISE	Red and white poppies.
NOISE	Why do mens even pull up their trousers?
Hair	Back to the color I love!

Table 3: dpmeans: Eight randomly sampled documents from the cluster with the most "Hair" label mentions for the best performing $f_n = 0.4$ run.

5.1.2 FANATIC vs. hdbscan, gmeans, LDA

FANATIC achieves better AMI scores at all f_n than hdbscan, gmeans and LDA, with the greatest performance difference occurring at $f_n = 0$. This indicates its superior ability to group similar documents together, especially in the absence of any topic noise. The other algorithms generally struggle to achieve the trifecta of high AMI, P^* and R^*, and also tend to cluster the same fraction of documents, f_D, independent of the amount of noise present, f_n (second row in Figure 1), indicating little sensitivity to filtering out topic noise.

Interestingly, hdbscan consistently achieves the highest P^* with very low R^*, indicating that it is a harsh filter - it can reliably filter noise documents but tends to discard relevant documents.

5.2 Number of Documents vs. Performance

We take the best performing runs from Section 5.1 at $f_n = 0.2$ and exponentially increase the number of documents, N_D, to answer how each algorithm:

- is affected by data perturbation;
- scales computationally with N_D.

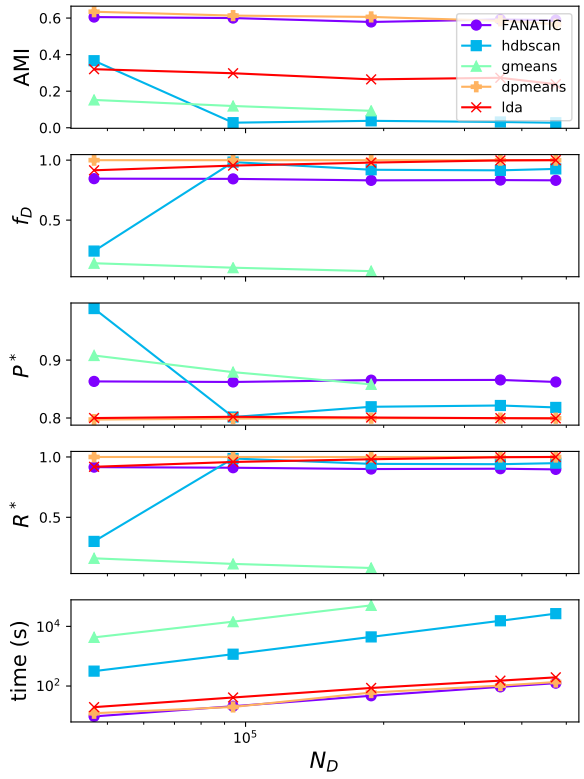

Figure 2: Number of documents, N_D vs. performance as measured by AMI (top row), fraction of documents in clusters, f_D (second row), and P^* and R^* (Equation 4, 3rd and 4th rows), and clustering time in seconds (bottom row). $N_D > 2 * 10^5$ results for gmeans are absent due to time restrictions.

Best results for each (N_D, algorithm) combination are displayed in Figure 2, and shows that FANATIC again performs best and is robust to changes in N_D. In particular:

- AMI, f_D, P^* and R^* do not change as a function of N_D indicating high stability.
- It has the highest AMI score (tied with dpmeans).
- The fraction of documents clustered, f_D is proportional to the fraction of noise (in particular, $f_D = 1 - f_n = 1 - 0.2 = 0.8$).
- P^* and R^* are both high, indicating it can filter noise while keeping valid documents in clusters.

All other algorithms show some additional drawback, including lower AMI score (gmeans, LDA, hdbscan), disproportionate cluster fraction, f_D (all other algorithms), lower pseudo-precision, P^* (LDA, dpmeans, mostly hdbscan), lower document-recall, R^* (gmeans, partially hdbscan), or instability of results as N_D changes (hdbscan).

5.2.1 Computational Efficiency

FANATIC, LDA and dpmeans all scale computationally very efficiently as shown in the bottom row of Figure 2 which plots clustering time (in seconds) vs. N_D. In particular FANATIC is two orders of magnitude faster than hdbscan, and at N_D=500k hdbscan takes over 7 hours while FANATIC takes 2 minutes. The slowness of hdbscan is likely due to the 300-dimensional embeddings (typical for the NLP domain, e.g., Pennington et al., 2014), and others in the community have also noticed that scaling for hdbscan degrades as embedding dimension increases (Leland McInnes, 2018).

5.3 Evaluation on Secondary Twitter Dataset

Algorithm	AMI	P^*	R^*
FANATIC	0.60	1	0.99
dpmeans	0.54	1	0.79
lda	0.37	1	0.97
hdbscan	0.29	1	0.37
gmeans	0.14	1	0.18

Table 4: Performance on the Twitter dataset. As explained in Section 5.3, P^*=1 for all algorithms since there is no topic noise.

To evaluate how FANATIC generalizes to other datasets, we briefly test on a collection of 20k tweets collected over 2019-12-18 to 2019-12-21 via the Twitter API[6]. These tweets span 20 hashtags (see Appendix A.2.1 for the list) which were manually vetted to be topically coherent and disparate from each other. Since each hashtag represents a coherent topic, in this experiment there is no "topic noise", and by default $P^* = 1$. Tweets are preprocessed into documents, clustered and evaluated in an identical manner to the Reddit data (see Section 4).

[6]https://developer.twitter.com/en/docs

The results in Table 4 show that FANATIC again has the highest AMI, highlighting its superior ability to cluster similar documents together. It also has the highest pseudo-recall, R^*, indicating that it classified almost no documents as noise, as it ideally should since no topic noise is present. In Appendix A.2.2 we show five randomly sampled documents from the cluster with the most "#crypto" mentions for each algorithm's best run. These samples qualitatively match the findings from Section 5.1: namely that FANATIC is best at grouping similarly-labeled documents together, followed by dpmeans. hdbscan again tends to act as a "harsh filter", yielding precise clusters at the cost of filtering significant amounts of valuable content.

6 Conclusion

In this paper we present the FANATIC algorithm that is capable of robustly extracting coherent topics, even in the presence of topic noise. We first showed that AMI scores for FANATIC were consistently high across three experiments, indicating general ability to group similar documents together. Second, we showed that pseudo-precision, P^*, and pseudo-recall, R^*, were consistently high, demonstrating its robustness to detect and filter topic noise. Third, we demonstrated that FANATIC's consistent performance as the noise fraction, f_n, increased over the total number of documents displayed its robustness to different scenarios. As an added advantage, FANATIC performed best with zero topic noise. Finally, we found FANATIC to be two orders of magnitude faster than hdbscan, demonstrating its scalability and efficiency in the NLP domain.

We particularly recommend FANATIC over other clustering algorithms if the number of documents is large and/or topic noise is present.

Acknowledgments

We thank Maria Pershina for her help throughout the research and paper writing process, Daniel Preotiuc-Pietro and Leslie Barrett for their invaluable feedback, and the anonymous reviewers for their constructive reviews.

References

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. *ArXiv preprint*, abs/2001.08435.

Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. 2010. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, page 12.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 601–608. MIT Press.

Axel Bruns and Jean E Burgess. 2011. The use of twitter hashtags in the formation of ad hoc publics. In *Proceedings of the 6th European consortium for political research (ECPR) general conference 2011*.

Andrey Chinnov, Pascal Kerschke, Christian Meske, Stefan Stieglitz, and Heike Trautmann. 2015. An overview of topic discovery in twitter communication through social media analytics. In *AMCIS*.

Michael D Conover, Jacob Ratkiewicz, Matthew R Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. 2011. Political polarization on twitter. *Icwsm*, 133(26):89–96.

Stephan A Curiskis, Barry Drake, Thomas R Osborn, and Paul J Kennedy. 2020. An evaluation of document clustering and topic modelling in two online social networks: Twitter and reddit. *Information Processing & Management*, 57(2):102034.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80:150–156.

scikit-learn developers. 2020. Adjustment for chance in clustering performance evaluation.

Paolo Ferragina, Francesco Piccinno, and Roberto Santoro. 2015. On analyzing hashtags in twitter. In *Ninth International AAAI Conference on Web and Social Media*. Citeseer.

Edward B Fowlkes and Colin L Mallows. 1983. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569.

Sean Gilpin, Buyue Qian, and Ian Davidson. 2013. Efficient hierarchical clustering of large high dimensional datasets. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 1371–1380. ACM.

Daniel Godfrey, Caley Johns, Carl Meyer, Shaina Race, and Carol Sadek. 2014. A case study in text mining: Interpreting twitter data from world cup tweets. *ArXiv preprint*, abs/1408.5427.

ChengYue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. FRAGE: frequency-agnostic word representation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1341–1352.

Martijn Gösgens, Liudmila Prokhorenkova, and Alexey Tikhonov. 2019. Systematic analysis of cluster similarity indices: Towards bias-free cluster validation. *ArXiv preprint*, abs/1911.04773.

Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A Zighed. 2013. Information diffusion in online social networks: A survey. *ACM Sigmod Record*, 42(2):17–28.

Isabelle Guyon, Nada Matic, Vladimir Vapnik, et al. 1996. Discovering informative patterns and data cleaning.

Greg Hamerly and Charles Elkan. 2003. Learning the k in k-means. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 281–288. MIT Press.

Brian Kulis and Michael I. Jordan. 2012. Revisiting k-means: New algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Steve Astels Revision Leland McInnes, John Healy. 2018. Hdbscan github issue 167.

Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. 2009. The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pages 274–285. Springer.

Bernard Marr. 2018. How much data do we create every day? the mind-blowing stats everyone should read.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11).

Marina Meilă. 2007. Comparing clusteringsan information based distance. *Journal of multivariate analysis*, 98(5):873–895.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Albert Park and Mike Conway. 2018. Harnessing reddit to understand the written-communication challenges experienced by individuals with mental health disorders: Analysis of texts from mental health communities. *Journal of medical Internet research*, 20(4):e121.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. 2016. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17(1):4635–4666.

Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2011. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM*, 63.

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.

Marcilio CP de Souto, André LV Coelho, Katti Faceli, Tiemi C Sakata, Viviane Bonadia, and Ivan G Costa. 2012. A comparison of external clustering evaluation indices in the context of imbalanced data sets. In *2012 Brazilian Symposium on Neural Networks*, pages 49–54. IEEE.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

A Appendices

A.1 Supplemental Material for Reddit

A.1.1 Dataset: List of Subreddits Used

Table 5 displays the list of “coherent" and “noise" subreddits in our Reddit dataset used throughout our experiments in Sections 5.1 and 5.2. “Coherent" subreddits encapsulate a single topic while “noise" subreddits lack identifiable topics. We remind the reader that these topic labels have been derived solely from Reddit post titles, with all additional content (pictures, text body, comments, etc.) discarded. See Section 3.1.1 for additional discussion.

Coherent subreddits
MathHelp, Cartalk, aws, flashlight, Hair, borrow, MovieSuggestions, gamingsuggestions, ufc, ios, coupons, Dragonballsuper, progresspics, Soccer_Goal, Xiaomi, tattoos, Dentistry, NubzSports, Animesuggest, StillBullsToMe, giveaways, ThePodcastFeed, christmas, CanadaWeedStocks, url
Noise subreddits
OhGirlIamInTrouble, longtail, shower_thoughts, Denver, Waxpen, NoStupidQuestions, orangecounty, economy, hatebeingpoor, ReefTank, mylittleandysonic1_ss, DippingTobacco, undelete, mashable, shittyaskscience, ImagesOfNewZealand, hockeyjerseys, BravoRealHousewives, bestoflegaladvice, nottheonion, mildly_interesting, traaaaaaannnnnnnnnns, asianamerican, ImagesOfTexas, devils, science, minecraftsuggestions, Screenwriting, FreeCompliments, uncensorednews, simracing, BSPN, JustTheTopNews, FiveYearsAgoOnReddit, TumblrInAction, theworldnews, crochet, personalfinance, Showerthoughts, raisedbynarcissists, AFL, Frei_Donald, DorsetNews, lotr, TheSimpsons, heroesofthestorm, HPfanfiction, whatisthisthing, feedthebeast, CrazyIdeas, bayarea, gamecollecting, ImagesOfGeorgia, fark, Right_Politics, guineapigs, firstworldproblems, MoviePassClub, IndiaSpeaks, britishproblems, SeattleWA, SCJerk, subredditSimulator, quityourbullshit, trailerparkboys, opieandanthony, whowouldwin

Table 5: All Coherent and Noise subreddits

A.1.2 Dataset: Quality Control

Although ∼90% of subreddits have been filtered (from the original set of 1000), potentially introducing bias, this is a worthwhile tradeoff as our final dataset enables reliable assessment of cluster quality in the presence of topic noise, a valuable measurement previously absent from the community.

Our downstream results assume that the labels obtained from our annotation task generalize to the entire subreddit. While in general this is an effective way to obtain labels for thousands of documents (which are infeasible to annotate individually), it is possible that some subreddits have been mislabelled.

We have tried to minimize this possibility by using the strictest possible filters as described in Section 3.1.3, namely unanimous annotator agreement and semantic agreement on the provided summary. We also restricted the data to the 2017 year to minimize potential distribution shift.

It is also possible that some fraction of the documents within a coherent subreddit are actually noise. During our annotation task we allowed annotators to select documents from coherent subreddits that did not belong. We found that, on average, 0.8 ± 2.1 of the 20 titles shown were selected, suggesting that the fraction of misannotated coherent documents is low. For our noise subreddits, since all 67 of them were uniquely annotated as noise and will be combined into a single `NOISE` label (see Section 4.3.1), no individual subreddit contributes greatly to the whole, mitigating risk of any one subreddit having been misannotated. In general, as long as the misannotated fraction is low its effect on downstream metrics will also be low.

Overall, we have taken considerable care in ensuring that the labels reflect the topical content.

A.1.3 Experimental Details

For the experiments in Sections 5.1 we use the RS_2017-11.bz2 data file from Pushshift[7] which contains Reddit data from November 2017, while for the experiments in Section 5.2 we use the RS_2017-01.bz2 - RS_2017-11.bz2 data files (January through November 2017 data), as needed, depending on the amount of required data for the experiments.

All experiments and derived clustering times

[7] https://files.pushshift.io/reddit/submissions/

were run with 4 CPU cores and and 50GB memory. Experiments that took over two days to run or required additional memory were not considered for the paper (in practice this only restricted the longest `gmeans` runs).

Table 6 shows the min/max range and scale for each `FANATIC` hyperparameter. "lin.", "log" and "int" correspond to linear, logarithmic and postitive integers. λ_{cos} and λ_{euc} correspond to λ when cosine and Euclidean distances were selected.

These ranges were set by a) getting initial results from very broad hyperparameter ranges and b) reducing the hyperparameter space to exclude ranges with consistently very poor results for improved efficiency.

	λ_{cos}	λ_{euc}	L	N_C	S_C	M_R	M_d
min	0.1	1	0	10^1	10^1	0	0
max	1	3.5	0.3	10^3	10^3	1	1
scale	lin.	lin.	lin.	log	log	int	lin.

Table 6: `FANATIC` hyperparameter ranges.

The best `FANATIC` hyperparameters are listed in Table 7 for each (N_D, f_n) combination. Numbers are rounded to two significant digits, and when $M_R = 0$ then $M_d = 0$ by default. For the distance function, $\mathcal{D}$, λ_{cos} and λ_{euc} correspond to λ when cosine and Euclidean distances were selected.

N_D	f_n	λ_{cos}	λ_{euc}	L	N_C	S_C	M_R	M_d
50k	0.0	.56	-	.00	57	36	1	.40
50k	0.1	.46	-	.00	645	316	1	.77
50k	0.2	.84	-	.012	22	378	0	0
50k	0.3	.44	-	.014	29	25	0	0
50k	0.4	.39	-	.04	37	379	1	.50
50k	0.5	.32	-	.01	50	70	1	.41
50k	0.6	.31	-	.03	557	364	1	.91
50k	0.7	-	2.3	.05	835	346	1	.80

Table 7: `FANATIC` hyperparameters of best runs.

A.2 Supplemental Material for Twitter

A.2.1 Dataset: List of Twitter Hashtags Used

Table 8 displays the list of 20 Twitter hashtags used for our experiment in Section 5.3. Each hashtag was manually vetted to be both topically coherent and disparate from other hashtags.

A.2.2 Example clusters

In Tables 9 - 13, for each algorithm's best run on the Twitter dataset (see Section 5.3), we show five randomly sampled documents from the cluster with the most "crypto" documents. In each table caption we also include the percent of #crypto documents that were filtered out as noise. "..." indicates truncation of the tweet's text due to space constraints.

Twitter Hashtags
#stocks, #isupportcaa_nrc, #crypto, #climatechange, #cybersecurity, #trump, #ai, #brexit, #microsoft, #demdebate, #nswfires, #oil, #starliner, #syria, #cdnpoli, #got7, #tesla, #hsbc, #soundcloud, #christmas

Table 8: Twitter Hashtags used for the experiment in Section 5.3.

Label	Text
#crypto	Crypto Rainbow XRB Airdrop thanks for the opportunity
#crypto	RT @ravikikan: Blockchain Statistics via @ravikikan Crypto cryptocurrency eth startup defstar5...
#crypto	$VET cup and handle forming. VeChain Crypto HODL https://t.co/j32S5MIHa5
#crypto	$BTC crypto $ETH Come on Honey badger, show us what you can do https://t.co/g6NMtiOFWe
#crypto	@NanoTipBot @nanillionaire How this awesome shoot me $NANO for my doggy crypto

Table 9: `FANATIC`: Six randomly sampled documents from the cluster with the most "crypto" documents for the best performing run on the twitter dataset. 0.2% of "#crypto" mentions were filtered out as noise.

Label	Text
#crypto	RT @ravikikan: Are You Ready For Blockchain ? via @ravikikan @Gartner_inc Crypto...
#ai	RT @MikeQuindazzi: 4G download speeds for Smartphone in the USA; @StatistaCharts via @MikeQuindazzi...
#crypto	RT @ravikikan: Blockchain Statistics via @ravikikan Crypto cryptocurrency eth startup...
#crypto	Check out the XTRABYTES SubReddit https://t.co/aLskB3D2sO crypto cryptocurrency tech technology...
#crypto	RT @ravikikan: Blockchain Statistics via @ravikikan Crypto cryptocurrency eth startup defstar5...

Table 10: `dpmeans`: Six randomly sampled documents from the cluster with the most "crypto" documents for the best performing run on the twitter dataset. 0.2% of "#crypto" mentions were filtered out as noise.

Label	Text
#crypto	RT @ravikikan: Blockchain Statistics via @ravikikan Crypto cryptocurrency eth startup defstar5 tech...
#crypto	Check out the XTRABYTES SubReddit https://t.co/oD6waciUcK crypto cryptocurrency tech technology $BTC $ETH...
#crypto	Check out the XTRABYTES SubReddit https://t.co/NpKLGpfHjy crypto cryptocurrency tech technology $SNGLS...
#crypto	Because Blockchain can be better. XTRABYTES Check out the website: https://t.co/7jXAkTjMo4 crypto...
#crypto	Because Blockchain can be better. XTRABYTES Check out the website: https://t.co/WTEoWQeIhj crypto...

Table 11: `hdbscan`: Six randomly sampled documents from the cluster with the most "crypto" documents for the best performing run on the twitter dataset. 49% of "#crypto" mentions were filtered out as noise.

Label	Text
#crypto	Because Blockchain can be better. XTRABYTES Check out the website: https://t.co/9IQRKlPKcL crypto...
#crypto	Because Blockchain can be better. XTRABYTES Check out the website: https://t.co/1blFeXMG9y crypto...
#isupportcaa_nrc	RT @nijunction: We walked right into a rally which was...
#crypto	@GSMAm4D @CGAP financialinclusion Mobile Crypto CALL to VOTE in TELCOIN POLLs before elapse POLL1:...
#isupportcaa_nrc	ISupportCAA_NRC Too much Hatred for Hinduism. Where is Secularism...

Table 12: `LDA`: Six randomly sampled documents from the cluster with the most "crypto" documents for the best performing run on the twitter dataset. 27% of "#crypto" mentions were filtered out as noise.

Label	Text
#crypto	Check out the XTRABYTES SubReddit https://t.co/jk8CUTnKDC crypto cryptocurrency tech...
#crypto	Check out the XTRABYTES SubReddit https://t.co/yUfhV9hqL9 crypto cryptocurrency tech...
#crypto	Check out the XTRABYTES SubReddit https://t.co/oD6waciUcK crypto cryptocurrency tech...
#crypto	Check out the XTRABYTES SubReddit https://t.co/yUfhV9hqL9 crypto cryptocurrency tech...
#crypto	Check out the XTRABYTES SubReddit https://t.co/cNZcrhLmYg crypto cryptocurrency tech...

Table 13: `gmeans`: Six randomly sampled documents from the cluster with the most "crypto" documents for the best performing run on the twitter dataset. 80% of "#crypto" mentions were filtered out as noise.