# Relevancer: Finding and Labeling Relevant Information in Tweet Collections

Ali Hürriyetoğlu[1,2(✉)], Christian Gudehus[3], Nelleke Oostdijk[2], and Antal van den Bosch[2]

[1] Statistics Netherlands, P.O. Box 4481, 6401 Heerlen, CZ, The Netherlands
a.hurriyetoglu@cbs.nl

[2] Centre for Language Studies, Radboud University, P.O. Box 9103, 6500 Nijmegen, HD, The Netherlands
{a.hurriyetoglu,n.oostdijk,a.vandenbosch}@let.ru.nl

[3] Faculty of Social Science, Ruhr University Bochum, 150 Building GB 04/148, 44801 Bochum, Germany
christian.gudehus@ruhr-uni-bochum.de

**Abstract.** We introduce a tool that supports knowledge workers who want to gain insights from a tweet collection, but due to time constraints cannot go over all tweets. Our system first pre-processes, de-duplicates, and clusters the tweets. The detected clusters are presented to the expert as so-called information threads. Subsequently, based on the information thread labels provided by the expert, a classifier is trained that can be used to classify additional tweets. As a case study, the tool is evaluated on a tweet collection based on the key terms 'genocide' and 'Rohingya'. The average precision and recall of the classifier on six classes is 0.83 and 0.82 respectively. At this level of performance, experts can use the tool to manage tweet collections efficiently without missing much information.

**Keywords:** Social media analysis · Event analysis · Data mining · Text mining · Machine learning · Social signal processing · Decision support systems · Genocide · Rohingya

## 1 Introduction

Keyword-based collections of tweets tend to be overly rich in the sense that not all tweets are relevant for the task at hand. Tweets can be irrelevant for a particular task, for instance because they are posted by non-human accounts, contain spam, refer to irrelevant events, or point to an irrelevant sense of an ambiguous keyword used in collecting the data. This richness has a number of dynamic characteristics, which can be present in a static or continuously updated collection, as well. There are no guarantees that tweet collections will have similar characteristics across different periods of time.

With the aim of managing tweet collections, we introduce the term *information thread* and our tool, Relevancer. An information thread characterizes a


E. Spiro and Y.-Y. Ahn (Eds.): SocInfo 2016, Part II, LNCS 10047, pp. 210–224, 2016.
DOI: 10.1007/978-3-319-47874-6_15

specific informationally coherent set of tweets. Relevancer is the tool we developed to analyze a tweet collection in terms of information threads.

Related sets of tweets (information threads) are initially detected using unsupervised machine learning. They are then confirmed by a human expert, and are used as training data in order to classify any remaining or new tweets using supervised machine learning. An expert can be anybody who is able to make knowledgeable decisions about how to annotate tweet clusters in order to understand a tweet collection in a certain context.

Relevancer enables an expert to analyze a tweet collection, i.e. any set of tweets that has been collected by using keywords. The tool requires expert feedback in terms of cluster annotation in order to complete the analysis. Experts can repeat the annotation process in case they collect new data with the same keywords. Alternatively they can decide to do another type of annotation once they understand the collection better after evaluating the first clusters. Our method advances the state of art in terms of efficient and complete understanding and management of a non-standard, rich, and dynamic data source.

This paper illustrates the functionality of Relevancer with a use case based on a particular tweet collection that we processed. It serves to illustrate the different steps in the description of our approach and the way that the Relevancer tool we developed supports it. The strength of our approach is the ability to scale to a large collection without sacrificing the precision or the recall by understanding intrinsic characteristics of the used key terms on social media. Finally, sharing the responsibility for completeness and precision with the users of the tool ensures they will achieve and preserve the target performance they require.

The remainder of the paper is structured as follows. In Sect. 2, we first describe related research. Section 3 introduces the concept 'information thread'. Then, in Sects. 4 and 5, we describe the structure of the tool and give information about the tweet collection used for the case study we did based on two key terms ('genocide' and 'Rohingya'). The results are presented in Sect. 6. Finally, Sect. 7 concludes this paper.

## 2 Related Studies

Identifying different uses of a word in different contexts is a word sense induction task [8]. This task is especially challenging for tweets, as they have a limited context [5]. Moreover, the diversity of the content on Twitter [3] and specific information need of an expert require a more flexible definition than a sense or topic of the content for tweet collections. We introduce the term information thread, which can be seen as contextualization of a sense.

Popular approaches of word sense induction on social media data are Latent Dirichlet allocation (LDA) [6,7], and user graph analysis [11]. The success of the former method depends on the given number of topics, which is challenging to determine, and the latter assumes the availability of user communities. Both methods provide one-fit-all solutions that are not flexible enough to allow users to customize the output based on a particular collection and the needs that an

expert or a researcher has. Therefore, we designed our tool Relevancer in such a fashion that it is not restricted by these assumptions. Relevancer makes it possible to discover information threads without any a priori restrictions.

Social science researchers have been seeking ways of utilizing social media data [4] and have developed various tools [1] to this end. Although these tools have many functions, they do not focus on identifying the uses of key terms. A researcher should continue to navigate the collection by specific key term combinations. Our study aims to enable researchers to discover specific or new uses, information threads, of the already used key terms and focus on the related tweets of a particular information thread.

A tweet collection management tool must take into account the characteristics of the social media data and achieve certain tasks. Available tools[1] have been designed for specific domains, languages, and use cases. Each of these suffers from one or more of the following restrictions: (a) they are restricted to analyzing tweets that contain at least a certain number of well-formed key terms or content words in specific languages; (b) they do not take into account tweet characteristics such as emoticons and personal language use; (c) they rarely use other attributes of a tweet text, such as mentions and URLs; and (d) they assume the availability of a group of annotators that are willing to label a sufficient number of tweets. We designed Relevancer[2,3] in such a way that it does not suffer from any of the aforementioned restrictions.

Enabling human intervention is crucial to ensuring high level performance in text analytics approaches [9]. Therefore we need to build a flexible pipeline that can facilitate human input in order to yield customized results [2]. Our approach responds to this challenge and need by providing the adaptive steps of analysis.

## 3 Information Threads

The task we address here is a specific case for collecting data using key terms from Twitter. The use and the interpretation of the key terms depends partly on the social context of a Twitter user and the point in time this word is used. Often, the senses and nuances or aspects that a word may have on social media cannot all be found in a dictionary. Therefore, we focus on the automatic identification of sets of tweets that contain the same contextualization of a sense, namely tweets that convey the same meaning and nuance (aspect). We name this kind of tweet groups as information threads.

For example, the word 'flood' has multiple senses; including being covered with water and filling somewhere with large amount of something[4]. A researcher

[1] For example, https://wiki.ushahidi.com/display/WIKI/SwiftRiver, https://github.com/qcri-social/AIDR/wiki/AIDR-Overview, and https://github.com/JakobRogstadius/CrisisTracker.

[2] https://bitbucket.org/hurrial/relevancer.

[3] http://relevancer.science.ru.nl.

[4] http://www.oed.com/view/Entry/71808.

working in the field of water management will want to focus on only the water-related sense. At the same time, he will want to discriminate between different nuances or aspects of this sense: past, current, up-coming events, effects, measures taken, etc. By incrementally clustering and labeling these, the collection is analyzed into different information threads.

The information thread concept allows a fine-grained management of all uses of a key word. In the case of this study, this approach enables the user of the tool to focus on uses of a key term at any level of granularity. For instance, tweets about a certain event, which takes place at a certain time and place, and tweets about a type of event, without a particular place or time, can be processed at the same level of complexity.

## 4 Tweet Collection Analysis with Relevancer

We retrieved a tweet collection from the public Twitter API[5] with the key terms 'genocide' and 'Rohingya' between May 25 and July 7 2015. We use this tweet collection to illustrate how Relevancer is used.

Relevancer begins by cleaning and exploring the tweet collection: it detects duplicates, extracts detailed features, and divides the collection into coherent subsets to which an expert can attach labels. Labeled tweets are then used to train an automatic classifier. This classifier can be used to analyze the remaining tweets or a new collection.

The analysis steps after duplicate and near-duplicate elimination of the tool are presented in Fig. 1. The details are explained in the following subsections.

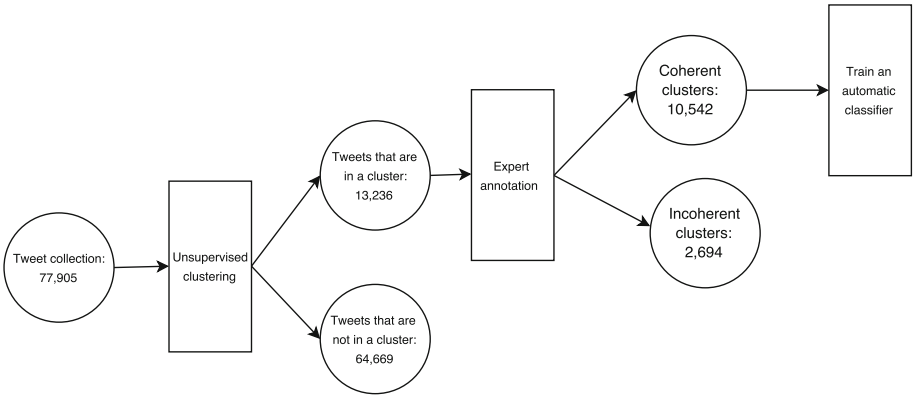


**Fig. 1.** Phases in the analysis process with the number of tweets at each step starting after duplicate and near-duplicate elimination

[5] https://dev.twitter.com/rest/public.

### 4.1 Pre-processing

Any tweet that has an indication of being a retweet is excluded. We use two types of retweet detection in a tweet: (a) the retweet identifier of the tweet's JSON file, (b) when the tweet starts with "RT @".

We proceed with normalizing the user names and URLs to "usrusrusr" and "urlurlurl" respectively. The normalization eliminates the noise introduced by huge number of different user names and URLs and preserves the abstract information that a user name or a URL is present.

Finally, we detect and exclude exact duplicate tweets. The importance of this step can be appreciated when we identify tweets such as exemplified in (1) and (2) below were posted 5,111 and 2,819 times respectively. However, we have to note that short tweets that signal the same information in different contexts are eliminated as well.

1. *usrusrusr The 2nd GENOCIDE against #Biafrans as promised by #Buhari has begun,3days of unreported aerial Bombardment in #Biafraland*
2. *usrusrusr New must read by usrusrusr usrusrusr A genocide & Human trafficking at a library in Sweden urlurlurl*

### 4.2 Feature Extraction

Any token that occurs in a tweet text is used as a feature in similarity calculations and in machine learning. Tokens are any space-delimited sequences of alphanumeric characters, emoticons, and sequences of punctuation marks. Features can be words, hashtags, letters, numbers, letter and number combinations, and emoticons. Punctuation mark sequences are treated differently. Sequences of punctuation marks comprising two, three or four items are considered as one feature. If the punctuation marks sequence is longer than four, we split them from left to right in tokens of length 4 by ignoring the last part if it is a single punctuation mark. The limit of length 4 is used for punctuation mark combinations, since longer combinations can be rare, which may cause them not to be used at all.

Detected tokens are used as unigram and bigram features. We apply a dynamically calculated threshold to the features. The threshold is half of the log of the number of tweets in a collection.

The motivation for using all aforementioned features is that this makes it possible to capture any nuance that may be present in groups of tweets. As a result, we can detect and handle many stylistic and textual characteristics properly. Moreover, using just the space for determining the features enables the tool to operate on any language that uses the blank space as token separator. Since none of the steps contain any language-specific optimization and the language and task related information is provided by a human user, we consider our method to be language-independent.

### 4.3 Near-Duplicate Detection

Most near-duplicate tweets occur because the same quote is used, the same message is tweeted, or the same news article is shared with other people. Since duplication does not add information and may harm efficiency and performance of the basic algorithms, we remove near-duplicate tweets by keeping only one of them in the collection.

The near-duplicate tweet removal algorithm is based on cosine similarity of the tweets. If the pairwise cosine similarity of two tweets is larger than 0.8, we assume that these tweets are near-duplicates of each other. In case the available memory does not allow to process all tweets at once, we apply a recursive bucketing and removal procedure. The recursive removal starts with splitting the collection into buckets of tweets of a size that can be handled efficiently. After removing near-duplicates from each bucket, we merge the buckets and shuffle the remaining tweets, which are unique in their respective bucket. We repeat the removal step until we can no longer find duplicates in any bucket.

For instance, the near-duplicate detection method recognizes the tweets (1) and (2) below as near-duplicates and leaves just one of them in the collection.

1. *Actor Matt Dillon puts rare celebrity spotlight on Rohingya urlurlurl#news*
2. *urlurlurl Matt Dillon puts rare celebrity spotlight on Rohingya urlurlurl*

### 4.4 Information Thread Detection

The information thread detection step aims at finding available information threads related to the key terms used to collect a tweet collection. In case the tweets in the collection do not share a certain key term, this step will still find related groups of tweets that are about certain uses of the words and word combinations in this collection. This groups will represent information threads.

We think that any approach that tries to find a relation between all tweets will fail to some extent. Because twitter data is extremely rich in a sense that it is not realistic to think that every tweet can be related to particular tweets. Therefore, our method aims at detecting only related group of tweets and ignoring tweets that do not present any clear relation to other tweets. These outlier tweets will be available to be analyzed by the classifier or manually at the end of the process.

The clustering aims to identify coherent clusters of tweets in order to understand the collection in terms of clusters, which constitute an information thread. We run a basic algorithm, K-Means for small collections and MiniBatch K-Means for large ones[6]. We repeat the clustering and cluster selection steps in iterations until we reach the requested number of coherent clusters, which is provided by the expert. Clusters that fit our coherence criteria are picked for annotation and the tweets in them are not included in the following iteration of clustering.

[6] We used scikit-learn v0.17.1 for all machine learning tasks in this study http://scikit-learn.org.

### 4.5 Relevancer Parameters

Formulas for parameter value assignment were identified empirically with several principles in mind. Since tweet collections do not demonstrate a clear separation of tweet clusters for the whole set, we assume that optimizing the parameters is not feasible. Moreover, having expert feedback for the selected coherent clusters allows us not to spend relatively excessive amounts of time on optimizing the clustering parameters for yielding a better automatic clustering. This generic approach aims at finding only main coherent clusters of tweets.

Relevancer facilitates two levels of parameters: clustering and coherence parameters. The clustering parameters depend on the collection size (n) and the time spent on searching for clusters. The cluster coherence parameters control the selection of clusters for annotation. These parameters are updated at each iteration (i) based on the requested number of clusters (r). A requested number of clusters is given as a stopping criterion for the exploration and as an indicator of the adaptation step for the value of the other parameters at each cycle.

**Clustering parameters.** There are two clustering parameters. K is the number of expected clusters and t is the number of initializations of the clustering algorithm before it delivers its result. These parameters are adjusted at each iteration.

**Coherence parameters.** The second layer of the parameters contains the number of clusters that should be generated for the expert (r) and the cluster coherence criteria. Although these parameters have default values, they can be set by the expert as well. Adaptation of the cluster coherence criteria steps is small if we are close to our target.

The value of k at each iteration is assigned based on Eq. 1. The parameter k is equal to half of the square root of the tweet collection size at that iteration plus the number of previous iterations times the difference between the requested number of clusters and the detected number of clusters (a). This adaptive behavior ensures that if we do not have many clusters after several iterations, we will be searching for smaller clusters at each iteration.

The result of the clustering, which is evaluated by the coherence criteria, is the best score in terms of inertia after initializing the clustering process (t) times in an iteration. As provided in Eqs. 1 and 2, (t) is log of the tweet collection at the current iteration plus the number of iterations performed until that point times (t), Eq. 2. This formula ensures that the more time it takes to find coherent clusters, the more the clustering will be initialized before it delivers any result.

$$k = \frac{\sqrt{n_i}}{2} + (i \times (r_i - a_i)) \tag{1}$$

$$t = \log_{10} n_i + i \tag{2}$$

If the requested number of clusters is too high for the collection, the adaptive relaxation of the coherence parameters stops at a level any cluster may qualify

as a coherent. We think it is unrealistic to expect relatively good clusters in such a situation. In such a case, the available clusters are returned before they reach the number requested by the expert. On the other hand, in case the algorithm exceeds the number of requested clusters in an iteration, it completes the search for coherent clusters and yields all detected clusters.

### 4.6 Cluster Annotation

Automatically selected clusters are presented to an expert for identifying clusters that present certain information threads[7]. After the annotation, each thread may consist of several clusters. In other words, similar clusters should be labeled with the same label. Clusters that are not clear and fall out outside the focus of a study should be labeled as incoherent and irrelevant respectively. This decision is taken by a human expert. The tweets that are in an incoherent labeled cluster are treated as the tweets that are not placed in any coherent cluster.

Sample tweets, the closest and the farthest to the cluster center, from coherent and incoherent clusters are presented in Table 1. A coherent cluster (CH) tweets have a clear relation that allows us to treat this group of tweets as pertaining the same information thread. Incoherent clusters are summarized under IN1 and IN2. In the former group, tweets do not have any relation between the first and last tweet[8]. The latter group contains a meta-relation that can be considered as an indication of being about a video. However, if the expert is interested in the content, this cluster should be annotated as incoherent.

**Table 1.** Tweets that are the closest and the farthest to the cluster center for coherent (CH), incoherent type 1 (IN1) and type 2 (IN2) clusters

| | |
|---|---|
| CH | – myanmar rejects'unbalanced' rohingya remarks in oslo (from usrusrusr urlurlurl<br>– shining a spotlight on #myanmar's #rohingya crisis: usrusrusr remarks at oslo conf on persecution of rohingyas urlurlurl |
| IN1 | – un statement on #burma shockingly tepid and misleading, and falls shortin helping #rohingya says usrusrusr usrusrusr urlurlurl<br>– usrusrusr will they release statement on bengali genocide 10 months preceding'71 ? |
| IN2 | – i liked a usrusrusr video urlurlurl rwanda genocide 1994<br>– i liked a usrusrusr video urlurlurl fukushima news genocide; all genocide paths lead to vienna and out of vienna |

For instance, clusters that contain tweets like "plain and simple: genocide urlurlurl" and "it's genocide out there right now" can be gathered under the

[7] The annotation is designed to be done or coordinated by a single person in our setting.

[8] The expert may prefer to tolerate a few different tweets at the end of the group in case majority of the tweets are coherent and treat the cluster as coherent.

same label, e.g., actual ongoing events. If a cluster of tweets is about a recent event, a label can be created for that particular event as well. For instance, the tweet "the plight of burma's rohingya people urlurlurl" is about a particular event related to the Rohingya people. If we want to focus on this event related to Rohingya, we should provide a specific label for this cluster and use it to specify this particular context. We can use 'plight' as a label as well. In such a case, the context should specify cases relevant to this term.

In case the expert is not interested in or does not want to provide a specific label to a CH, the expert should attach the label irrelevant, which behaves as an umbrella label for all tweet groups that are not the present focus of the expert.

We developed a web-based user interface for presenting the clusters and assigning a label to the presented cluster. We present all tweets in a cluster if the number of tweets is smaller than 20. Otherwise, we present the first and the last 10 tweets in terms of the distance to the cluster center. This setting provides an overview and enables spotting incoherent clusters effectively. A cluster can be skipped without providing a label for it. In such a case, that cluster is not used in the following steps.

At the end of this process, an expert is expected to have defined the information threads for this collection and have identified the clusters that are related to these threads. Tweets that are part of a certain information thread can be used to understand the related thread or to create an automatic classifier that can classify new tweets, e.g., ones that were not included in any selected cluster at the clustering step, in classes based on detected and labeled information threads.

### 4.7 Creating a Classifier

Creating classifiers for tweet collections is a challenge that is mainly affected by label selection of the expert annotator, nature of the key word, and time of the collection. Consequently, the classes are or prone to be imbalanced.

The labeled tweet groups are used as training data for building automatic classifiers that can be applied 'in the wild' to any previous or new tweets, particularly if they are gathered while using the same query.

Relevancer facilitates the Naive Bayes algorithm for creating a classifier, which is one of the most basic supervised machine learning algorithms. We prefer this classifier due to its short training time and comparable performance to sophisticated machine learning algorithms [10] for text classification. We need time efficiency in order to be able to re-train a classifier in case an expert prefers to update the current classifier with new data or create another classifier after observing the results of a particular classifier.

Parameters of the Naive Bayes algorithm are optimized by using a grid search on the training data. The performance of the classifier is based on 15 % of the training data, which was held out and not used at the training step.

### 4.8 Scalability

Relevancer applies various methods in order to remain scalable independent of the number of tweets and features used. High number of tweets and features force this tool to have the scalability at the center of its design.

Large amount of tweets and a wide variety of features are processed by using basic and fast algorithms. Depending on the size of the collection, K-means or MiniBatch K-Means algorithms are employed in order to rapidly identify candidate clusters. The main parameter k, number of clusters parameter for K-Means, in these algorithms is increased at each iteration in order to target finding more and smaller clusters than previous iteration. Targeting more and smaller clusters increase chance of identifying coherent clusters.

Tweets in coherent clusters are excluded from the part of the collection that enter the subsequent clustering iteration. This approach shrinks the collection size at each iteration. Moreover, the criteria for coherent cluster detection is relaxed at each step until certain thresholds are reached in order to prevent the clustering from being repeated.

The Naive Bayes classifier was chosen in order to create and evaluate a classifier at a reasonable time. The speed of this step enables users to decide whether they will use a particular classifier or need to generate and annotate additional coherent clusters immediately.

This optimized cycle enables experts to be able to provide feedback frequently without having to wait too long. As a result, the quality of the results increases with minimal input and time for a particular task.

## 5 Tweet Collection

We collected 363,959 tweets with the key terms ‘genocide’ and/or ‘Rohingya’ between May 25 and July 7 2015. The number of tweets that contain only ‘genocide’ and only ‘Rohingya’ are 269,131 and 137,319 respectively; 12,889 tweets contain both terms.

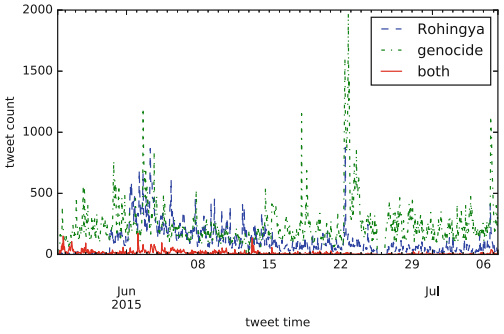


**Fig. 2.** Tweet distribution per key term

Figure 2 shows the distribution of the tweets for each subset. We can already observe that there are many peaks and a constant tweet ratio for each key term. Our analysis of the collection aims at understanding the constantly mentioned content and the peaks separately.

## 6 Results

We start the analysis by preprocessing the tweets in the collection. As a first step, we excluded 198,049 tweets, which are all retweets, of the 363,959 tweets, leaving 165,910 tweets in the collection. Then the exact duplicates were excluded from the collection, leaving 103,987 tweets in the collection. Finally 26,082 near-duplicate tweets were removed.

Figure 3 illustrates the final tweet distribution. We observe that the distribution was changed after we eliminated the repetitive information. Large peaks in the 'genocide' data were drastically eliminated and some of the small peaks disappeared. Thus, only relatively important peaks in the 'genocide' data remain and the peaks in tweets containing the key term 'Rohingya' becomes apparent.

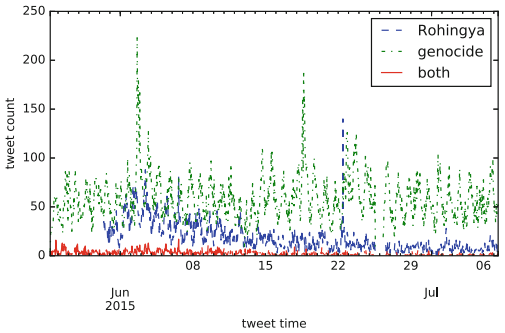


**Fig. 3.** Tweet distribution per key term after removing retweets, duplicates, and near-duplicates

The summary of the evolution of the size of the tweet collection is presented in Fig. 4. At each step, a large portion of the data is detected as repetitive and excluded. This cleaning phase shows us how size of the collection depends on the preprocessing. Decreasing the size of the data without loosing information[9] enables us to apply sophisticated analysis techniques to social media data.

After removing the duplicates and near-duplicates, we clustered the remaining 77,905 tweets. Although the requested number of clusters was 100, the number of generated clusters was 145, which contain 13,236 tweets. The cluster parameters were set to begin with the following values: (i) the distances of the closest

[9] We note that the repetition pattern analysis is valuable in its own right. However, this information is not within the scope of the present study.

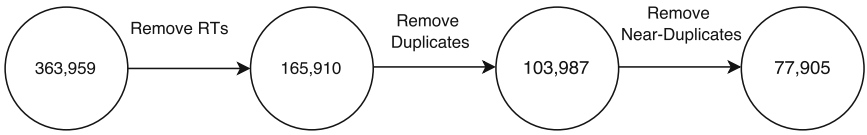


**Fig. 4.** Tweet number change at each step of the preprocessing in the collection

and farthest tweets to the cluster center have to be less than 0.725 and 0.875 respectively; and the difference of the distance to the cluster center between the closest and the farthest tweets in a cluster should not exceed 0.4.

The annotation of the 145 clusters by a domain expert yielded the results in Table 2. This process yielded 8 labels: Actual cases (AC), Cultural genocide (CG), Historical cases (HC), Incoherent (IN), Indigenous people genocide (IPG), Irrelevant (IR), Jokes (JO), and Mass migration (MM).

This step enabled the domain expert to understand the data by annotating only 17 % of the preprocessed tweets, which is 0.03 % of the complete collection, without the need of having to go over the whole set. Furthermore, annotating tweets in groups as suggested by the clustering algorithm improved the time efficiency of this process.

**Table 2.** Number of labeled clusters and total number of tweets for each of the labels

| | # of clusters | # of tweets |
|---|---|---|
| AC | 48 | 4,937 |
| CG | 7 | 375 |
| HC | 22 | 1,530 |
| IN | 32 | 2,694 |
| IPG | 1 | 109 |
| IR | 30 | 3,365 |
| JO | 1 | 30 |
| MM | 4 | 226 |
| Total | 145 | 13,236 |

We used the Relevancer to create an automatic classifier by using the annotated tweets. We merged the tweets that are under the JO (Jokes) label with the tweets under the Irrelevant label, since their number is relatively small compared to other tweet groups for generating a classifier for this class. Moreover, the incoherent clusters were not included in the training set. This leaves 10,552 tweets in the training set.

We used the same token characteristics explained in the feature extraction step to create the features used by the classifier. We added token trigrams to the unigrams and bigrams as features. The parameter optimization and the training

was done on 85 % of the labeled data and the test was performed on the remaining 15 %. The only parameter of the Naive Bayes classifier, $\alpha$, was optimized on the training set to be 0.105 by testing equally separated 20 values between 0 and 2.

The performance of the classifier is summarized in Tables 3 and 4 as a confusion matrix and evaluation summary. We observe that classes that have a clear topic, e.g., HC and CG, perform reasonably well. However, classes that are potentially a combination of many sub-topics, such as AC and IR, which contain JO labeled tweets as well, perform relatively worse. Detailed analysis yielded that the HC thread contains only a handful past events that were referred to directly. On the other hand, there is a lot of discussions in addition to clear event reference in the AC thread. As a result, labels that contain specific language use work better than the labels contain diverse language use.

**Table 3.** The rows and the columns represent the actual and the predicted labels of the test tweets. The diagonal provides the correct number of predictions.

| | AC | CG | HC | IPG | IR | MM |
|---|---|---|---|---|---|---|
| AC | 586 | 3 | 5 | 1 | 158 | 1 |
| CG | 1 | 42 | 0 | 0 | 3 | 0 |
| HC | 26 | 0 | 198 | 0 | 7 | 1 |
| IPG | 2 | 1 | 0 | 9 | 3 | 0 |
| IR | 62 | 1 | 3 | 0 | 441 | 1 |
| MM | 8 | 0 | 0 | 0 | 0 | 19 |

**Table 4.** Precision, recall, and F1-score of the classifier on the test collection. The recall is based only on the test set.

| | Precision | Recall | F1 | Support |
|---|---|---|---|---|
| AC | .86 | .78 | .81 | 754 |
| CG | .89 | .91 | .90 | 46 |
| HC | .96 | .85 | .90 | 232 |
| IPG | .90 | .60 | .72 | 15 |
| IR | .72 | .87 | .79 | 508 |
| MM | .86 | .70 | .78 | 27 |
| Avg/Total | .83 | .82 | .82 | 1,582 |

The result of this step is an automatic classifier that can be used to classify any tweet in the aforementioned six classes. Although the performance is relatively low for a class that is potentially a mix of various subtopics, the average scores, which are 0.83, 0.82, and 0.82 for the precision, recall, and F1 respectively, are sufficient for using this classifier in a real scenario.

## 7 Conclusion and Future Research

We presented Relevancer, our tweet collection management tool, and its performance on a collection collected with the key terms 'genocide' and 'Rohingya'. Each step of the analysis process was explained in quite some detail in order to shed light both on the tool and the characteristics of this collection.

The results show that the requested number of clusters should not be too small. Otherwise, missing classes may cause the classifier not to perform well on tweets related to information threads not represented in the final set of annotated clusters. Second, the reported performance was measured on the held out subset of the training set. This means that the test data has the same distribution with the subset of the training data that is used for the actual training. Therefore, generalization of those results to the actual social media context should be a concern of a further study.

At the end of the analysis steps, the expert successfully identified repetitive information, which is 79 % of the whole collection, analyzed coherent clusters, defined the main information threads, annotated the clusters, and created an automatic classifier that has 0.82 F1 score.

We plan to improve the feature extraction by including skip-grams and posting user dimensions and to introduce more sophisticated cluster evaluation metrics. Moreover, we can get feedback from the experts about which users or hashtags should best be ignored or included. This information can be used to update and continuously evaluate the classifiers. The posting user and hashtags have the potential to provide information about certain information threads. This information can enable the annotation step to be expanded to the tweets that are not in any cluster but contain information thread specific hashtags or users.

We do not carry out any post processing on the clusters. But we can identify outlier tweets in a cluster in terms of the distance to the cluster center in order to refine the clusters. This will enable the tool to have cleaner clusters and to find required clusters in fewer iterations.

Finally, the temporal distribution of the tweets in an information thread can guide us in defining its scope. An information thread over a short period will be treated in a manner different from a more persistent one.

**Acknowledgements.** This research was funded by the Dutch national research programme COMMIT. We gratefully acknowledge the contribution by Statistics Netherlands.

## References

1. Borra, E., Rieder, B.: Programmed method: developing a toolset for capturing and analyzing tweets. Aslib J. Inf. Manage. **66**(3), 262–278 (2014). http://dx.doi.org/10.1108/AJIM-09-2013-0094
2. Chau, D.H.: Data mining meets hci: Making sense of large graphs. Technical report, DTIC Document (2012)

3. Choudhury, M.D., Counts, S., Czerwinski, M.: Find Me the Right Content! Diversity-based sampling of social media spaces for topic-centric search. ICWSM, pp. 129–136 (2011). http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewPDFInterstitial/2792/3290
4. Felt, M.: Social media and the social sciences: How researchers employ Big Data analytics. Big Data Soc. **3**(1), 1–15 (2016). http://bds.sagepub.com/lookup/doi/10.1177/2053951716645828
5. Gella, S., Cook, P., Baldwin, T.: One sense per tweeter... and other lexical semantic tales of twitter. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pp. 215–220 (2014). http://www.aclweb.org/anthology/E14-4042
6. Gella, S., Cook, P., Han, B.: Unsupervised word usage similarity in social media texts. In: Second Joint Conference on Lexical and Computational Semantics (*SEM), Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, vol. 1, pp. 248–253 (2013). http://www.aclweb.org/anthology/S13-1036
7. Lau, J.H., Cook, P., McCarthy, D., Newman, D., Baldwin, T., Computing, L.: Word sense induction for novel sense detection. In: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012), pp. 591–601 (2012)
8. Mccarthy, D., Apidianaki, M., Erk, K.: Word sense clustering and clusterability. Comput. Linguist. **42**(2), 4943 (2016)
9. Tanguy, L., Tulechki, N., Urieli, A., Hermann, E., Raynal, C.: Natural language processing for aviation safety reports: from classification to interactive analysis. Comput. Ind. **78**, 80–95 (2016)
10. Wang, S., Manning, C.: Baselines and Bigrams: simple, good sentiment and topic classification. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics vol. 94305(1), pp. 90–94 (2012)
11. Yang, Y., Eisenstein, J.: Putting Things in Context: Community-specific embedding projections for sentiment analysis. CoRR abs/1511.0 (2015). http://arxiv.org/abs/1511.06052