Noname manuscript No.
(will be inserted by the editor)

Effective Social Post Classifiers on Top of Search Interfaces

Ryan Rivas · Vagelis Hristidis

Received: date / Accepted: date

Abstract Applying text classification to find social media posts relevant to a topic of interest is the focus of a substantial amount of research. A key challenge is how to select a good training set of posts to label. This problem has traditionally been solved using active learning. However, this assumes access to all posts of the collection, which is not realistic in many cases, as social networks impose constraints on the number of posts that can be retrieved through their search APIs. To address this problem, which we refer as the *training post retrieval over constrained search interfaces problem*, we propose several keyword selection algorithms that, given a topic, generate an effective set of keyword queries to submit to the search API. The returned posts are labeled and used as a training dataset to train post classifiers. Our experiments compare our proposed keyword selection algorithms to several baselines across various topics from three sources. The results show that the proposed methods generate superior training sets, which is measured by the balanced accuracy of the trained classifiers.

Keywords Text classification · Social media · Search interfaces · Data mining

Declarations

Funding. This work was supported by NSF grants IIS-1838222 and IIS-1901379.

Conflicts of interest/Competing interests. Not applicable.

R. Rivas
University of California, Riverside
Riverside, CA, USA
Tel.: +1 951-827-2838
E-mail: rriva002@ucr.edu

V. Hristidis
University of California, Riverside
Riverside, CA, USA
Tel.: +1 951-827-2478
E-mail: vagelis@cs.ucr.edu

Availability of data and material. The datasets used in our experiments were collected from DailyStrength, Reddit, and [17].

Code availability. The code used in our experiments is available from: https://github.com/rriva002/Training-Post-Retrieval.

1 Introduction

Text classification in social media is an area of active research. Examples of its application include analyzing the demographics of health-related discussions [24], inferring event attendance from non-geotagged posts [7], and detecting posts promoting extremist ideologies [1]. Training a text classification model requires a large and high-quality training set of labeled posts, where by "high quality" we generally mean that the posts should have a good coverage of the various classes, and even coverage of the various post variants within a class.

Labeling posts is generally the hardest and most expensive step in text classification applications. There has been much work on active learning, which studies how to select a good set of posts to label to achieve high training set quality. Active learning techniques assume that we have access to all documents (posts) in a collection, and iteratively select some of them to label next. However, this is not a realistic assumption in many applications, where posts' access is conducted via a constrained application programming interface (API).

In this paper, we study the *training post retrieval over constrained search interfaces problem*, which attempts to generate a high-quality posts training set, given a user-defined topic and a labeling budget. The topic is described by a few keywords that the user provides, for example for the topic "suicide" the user may provide keywords "suicide," "depressed," and "kill."

As an example application of the training post retrieval over constrained search interfaces problem, consider trying to create a personal classifier for each user to filter social media posts. A user could provide a few initial keywords of interest and then the method would use these keywords to return posts for labeling. This labeling can be implicit, e.g. via clickthrough. Labeled posts can then be used to train a classifier and use it to filter further posts. Note that the initial set of keywords provided by the user are just a rough description of their latent interest profile, that is, we cannot just assume that every post that contains these keywords is relevant or that posts without these keywords are irrelevant.

The training post retrieval over constrained search interfaces problem presents several challenges. The first challenge is the constrained search interface. In contrast to active learning, a method that addresses this problem does not have access to all of the available data. Instead, it must make API search queries that retrieve a limited number of posts, thus selecting the keywords that retrieve the most useful results is of key importance. Another challenge is that if we use keyword queries to generate our training set we incur coverage bias as we only get positive and negative (for binary classifiers) examples that match these queries. This problem is generally not present with active learning, where a post is picked based on how hard it is for the current classifier to classify it and not based on keywords. A third challenge is that the user-provided keywords are not perfect; there is no guarantee that any keyword provided will give 100% relevant results when used to query an

API. For example, the keyword "vote" provided by a user interested in US politics may retrieve posts relevant to US politics, but may also retrieve posts relevant to voting in another country, voting on posts and/or comments on Reddit, etc.

A successful *keyword selection algorithm (KSA)* must overcome these challenges. We propose several KSAs, with the most effective being the *Top Positives Random Negatives Keyword Selection Algorithm (TPRN-KSA)*, which progressively creates keyword queries to retrieve posts, which are labelled and added to the training set. TPRN-KSA tries to achieve two goals: (a) *balance*, i.e. the number of positives and the number of negatives in the training dataset should be as close to equal as possible; and (b) *diversity*, meaning it should cover a wide range of posts within each class to properly model the data with respect to the topic of interest. We show how an algorithm has to have diversity in both the positive and the negative posts to achieve good performance.

In summary, TPRN-KSA has the following steps: First, for each input keyword it retrieves a small set of posts, which are labeled to estimate the percentage of positives for the keyword. Based on these estimations, a second portion of the budget is spent to retrieve and label more posts from the most promising (higher rate of positives) of these keywords. Finally, to address the problem of bias especially in the negative class, TPRN-KSA spends a third portion of the budget to retrieve and label a set of random posts from the API, which replace some of the biased negative posts that were retrieved during the first two steps. We carefully compute the budget for each step of the algorithm and for each keyword query to achieve the goals of balance and diversity.

A key finding in this work is that achieving diversity of the negative posts in the training set is more important that the diversity of the positive posts. Another finding is that it is not enough to add some random negative posts to the training set, but we have to also remove many or all of the biased negative posts. This may sound a little counterintuitive as more training data should be better than less. Detailed experimental evaluation shows that training a text classifier with the training set generated by TPRN-KSA outperforms state-of-the-art baseline keyword selection methods. The summary of our contributions is as follows:

- We formulate the training post retrieval over constrained search interfaces problem.
- We propose a suite of principled keyword selection algorithms, including TPRN-KSA, to solve the training post retrieval over constrained search interfaces problem.
- We perform comprehensive experiments on three real datasets, which show that our proposed algorithms outperform existing baselines.
- We study the underlying reasons why the training set generated by our methods is of higher quality than the baselines. We measure the training set's balance, and the diversity of the labeled posts in both the positive and negative classes. We show how these quantities affect the quality of the training set, that is, how they are correlated to the classifier's performance.

The remainder of the paper is organized as follows: Sect. 2 discusses prior work. Sect. 3 defines the training post retrieval over constrained search interfaces problem. Sect. 4.1 introduces two baseline KSAs and discusses their limitations. Sect. 4.2 describes our initial TP-KSA method, which addresses one of the shortcomings of the baseline KSAs. Sect. 4.3 presents TPRN-KSA, which further refines

TP-KSA to also achieve diversity among the negative posts and also balance between positives and negatives. Sect. 5 explains the experimental evaluation of our proposed method and presents the results of our experiments. We conclude in Sect. 6.

2 Related Work

Applying information retrieval techniques to analyze social media posts has been employed in several applications. Shen et al. developed a method to retrieve disaster event data from Twitter and other social media platforms based on event-specific hashtags [27]. Balsamo et al. proposed an information retrieval algorithm to mine data from users on Reddit by identifying subreddits relevant to opioid abuse [2]. Rao et al. proposed a neural network model specifically designed for ranking short social media posts, e.g. tweets [22]. Our proposed method differs from these in two ways. First, its goal is to retrieve a dataset of both positive and negative posts for training a text classifier rather than only posts relevant to some topic. However, the pipeline consisting of our method and a classifier could be considered an information retrieval framework in itself. Second, our proposed method is task-agnostic, i.e. it is not specifically made for any one topic or platform. We show in our experiments that our method works well across several topics and sources.

Previous work has examined collecting text data from a constrained interface using a classifier. Ruiz et al. studied how to maximize the number of relevant retrieved items using a rule-based classifier [25]. Li et al. proposed a data platform to continuously monitor the Twitter streaming API for tweets relevant to some topic using a classifier trained to detect such tweets [15]. These works are complementary to ours, as their frameworks are built around a trained text classifier which is then leveraged to gather relevant documents, whereas our work studies how to build such a classifier.

Several papers have used sets of "seed" words, phrases, or documents to find matching documents. Li et al. used seed words related to some topic and a dataset of unlabeled documents to perform dataless text classification [12]. Wang et al. proposed a technique to identify more relevant search keywords starting from an initial set of keywords for retrieving social media posts related to some topic [30]. Sadri et al. proposed a system that adapts to changes in a topic on Twitter over time by iteratively selecting phrases to track [26]. Proskurnia et al. developed a framework to extract patterns from reference documents to identify microposts related to a specific topic [21]. Li et al. developed a model for estimating the relevance between a document and a set of seed words relevant to a category using pre-trained word embeddings [13]. A limitation of relying on query keywords is that they can be a poor representation of information need [6]; we show in our experiments that our proposed method has better performance than other keyword-based methods.

Pool-based active learning uses a classifier to iteratively determine which samples from among a large dataset, e.g. a corpus of documents, are the most informative and asks a human labeler to assign a class label to them. Goudjil et al. proposed an active learning method for text classification that uses a set of SVM classifiers to determine the average posterior probability of each document within

subsets of the unlabeled data [9]. Zhang et al. argued that active learning with a convolutional neural network (CNN) text classifier should focus on documents that have the most effect on the word embedding space in contrast to traditional methods such as classifier uncertainty [31]. Pool-based active learning is inapplicable to the problem we address in this paper, as the constrained search interface prevents the full dataset from being evaluated for informativeness.

Stream-based active learning involves evaluating data points (e.g. social media posts) one at a time and deciding whether to use a classifier or a human labeler to assign a class label to each one. Smailovic et al. used an SVM classifier initially trained on a Twitter sentiment dataset to perform active learning on financial-related tweets [28]. Pohl et al. proposed a stream-based active learning method to train a classifier to detect social media posts related to crises while limiting the number of queries to human labelers [20]. Zhang et al. proposed a method to address the issue of imbalanced data in determining whether to query the human labeler by exploiting samples' second-order information [32]. This work is notably similar to ours, but one shortcoming of this method, as well as stream-based active learning methods in general, is that it relies on streaming data, e.g. Twitter's streaming API, and thus has little control over the number of relevant posts being evaluated during the active learning process.

Positive-unlabeled (PU) learning trains a binary classifier with a dataset consisting of positive-labeled samples and additional unlabeled samples, which may be positive or negative. Li and Liu applied PU learning to text classification by combining the Rocchio method with an SVM [16]. Li et al. used PU learning to identify fake reviews on the Chinese business review website Dianping [14]. PU learning is complementary to our work, as a dataset generated by a KSA can be supplemented with additional unlabeled posts to train a classifier with PU learning instead of using traditional machine learning. However, we also note that many PU learning methods rely on the assumption that the positives are selected randomly [8], which is not applicable in this scenario.

3 Problem Definition

In this section, we define the *training post retrieval over constrained search interfaces* problem. Given the following inputs

- List of keywords K relevant to a latent topic t
- Supervisor S, a human who labels each post as relevant or not relevant to t
- Labeling budget m, the number of posts that we submit to S for labeling

a *Keyword Selection Algorithm (KSA)* selects n pairs of (keyword query q_i, number of results r_i), $P = \{(q_1, r_1), \cdots, (q_n, r_n)\}$, to submit to the social media API, where $r_1 + \cdots + r_n = m$, as shown in Fig. 1. Let $s_1, s_2, \cdots, s_n$ be the sets of posts returned by the n queries, respectively. The obtained training set is $T = s_1 \cup \cdots \cup s_n$, which is labeled by S and then used to train classifier c. The goal is to pick the P that maximizes the performance of c. *Picking the best classification method is outside the scope of this paper; we use standard SVM and CNN [10] text classifiers in our experiments.*

Note that n is not an input; rather it is dictated by the KSA, e.g. the algorithm may use each keyword in K to perform a query or may derive another list of

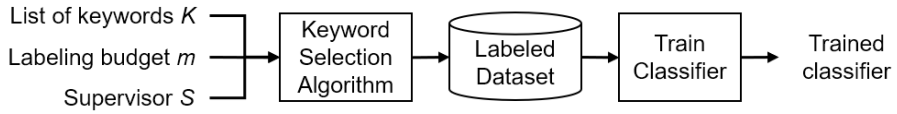

Fig. 1 Process of addressing the training post retrieval over constrained search interfaces problem.

keywords K' from K (e.g. a subset of K) that may be used to perform queries. P may also contain a special keyword query $q_i = \emptyset$ to mean that we obtain a random sample of posts. This does not come from a keyword in K, but is instead used by some of our methods. Support exists for retrieving random posts in a real-world setting, e.g. via the Reddit API's `/random` endpoint, but non-random sources of posts such as the Reddit API's `/new` endpoint may also suffice. Functionality similar to `/new` may be the only option for APIs on other social media. In general, the labeling budget m is not a constraint on the number of posts retrieved from an API, but rather the number of posts presented to the human labeler, as retrieving a large number of posts is trivial for most APIs. However, many of the KSAs evaluated in this study, including our proposed methods, retrieve only m posts.

Example: Consider a simple KSA that retrieves $\frac{m}{|K|}$ posts for each keyword in K. A user interested in posts discussing suicide provides $K = \{$"suicide", "depressed", "kill"$\}$ and $m = 600$ to the KSA, which selects $P = \{($"suicide", $200), ($"depressed", $200), ($"kill", $200)\}$. Then, for each $(q_i, r_i) \in P$, the KSA retrieves a set of posts s_i where $|s_i| = r_i$ and each post in s_i contains q_i. Next, the KSA presents the final set of posts $T = s_1 \cup s_2 \cup s_3$ to the user for labeling. The KSA then returns the labeled dataset, which is used to train a text classifier.

4 Methods

4.1 Baseline Keyword Selection Algorithms

In this section, we describe two simple baseline KSAs for comparison to our proposed method.

All-Keywords KSA. As described in the example above, this KSA uses every keyword in K along with a labeling budget m and supervisor S. For each keyword k, it retrieves $\frac{m}{|K|}$ posts that contain k from a keyword search API and queries S for their class labels. After all of the keywords' posts have been retrieved and labeled, it returns the labeled dataset. We expect the diversity of the negatives in this dataset to be low because every negative contains a keyword in K.

50-50 KSA. This KSA attempts to add diversity in the form of randomly retrieved posts. Like All-Keywords, it uses every keyword in K. However, instead of retrieving $\frac{m}{|K|}$ posts for each keyword k, it retrieves $\frac{m}{2|K|}$ posts that contain k for each keyword k and $\frac{m}{2}$ random posts. For example, given $K = \{$"suicide", "depressed", "kill"$\}$ and $m = 600$, 50-50 selects $P = \{($"suicide", $100), ($"depressed", $100), ($"kill", $100), (\emptyset, 300)\}$ (recall that the special keyword query $\emptyset$ retrieves random posts from an API).

Limitations. As previously mentioned, All-Keywords lacks diversity in its negatives. 50-50 attempts to address this by adding random posts, but both methods are highly dependent on the relevance of the keywords in K to achieve good balance. With All-Keywords, half of the posts retrieved by using keywords to query the API must be relevant, but with 50-50, all (or nearly all) of them should be relevant. As it is difficult to guarantee the relevance of a keyword, these KSAs often generate datasets that have poor balance.

4.2 KSA with Balance

In this section, we describe our initial Top Positives Keyword Selection Algorithm (TP-KSA). The goal of this method is to address the balance issue due to dependence on keyword relevance experienced in All-Keywords and 50-50. Specifically, TP-KSA selects the most descriptive keywords in K, that is, keywords whose posts have a relatively high ratio of positives. It then splits its budget equally across these keywords. We also present a variant, TPP-KSA, which retrieves posts proportionally to the rate of positives for each keyword. Table 1 summarizes the notation used by our methods.

Table 1 Notation used by our methods

Notation	Description
K	Initial list of keywords.
m	Labeling budget, i.e. the number of posts to present to S for labeling.
S	Supervisor; a human labeler to whom retrieved posts are presented for labeling.
X	List of posts to which retrieved posts are added.
y	List of class labels, where y_i is the class label for X_i.
p	List of percent positive, where p_i is the percentage of positives for keyword K_i.
K'	List of keywords from K retained after determining which ones to remove.
p'	List of percent positive, where p'_i is the percentage of positives for keyword K'_i.
b	Budget allocated equally to each keyword in K' (Eqn. 1).
X_k	List of posts retrieved by querying API with keyword k.
y_k	List of class labels; y_{ki} is the class label for post X_{ki} as determined by S.
s	Number of sample posts to retrieve for each keyword in K (Eqn. 2).
p_k	The percentage of positive posts for keyword k.
b_i	Budget allocated proportionally to keyword K'_i (Eqn. 3).

4.2.1 Top Positives KSA (TP-KSA)

This method determines which keywords in K to retain according to their relevance as determined by the ratio of positive posts that they retrieve. The algorithm first expends some of its budget m to call the `SampleKeyWordPosts` subroutine to retrieve and label a small sample of posts from each keyword in K. It then calls the `SelectKeywords` subroutine to determine which keywords to retain. Next, the algorithm evenly distributes the remainder of labeling budget m, which is m minus the total number of sample posts retrieved $|X|$, among the retained keywords in

K'. The budget b allocated to each remaining keyword is defined in Eqn. 1.

$$b = \left\lfloor \frac{m - |X|}{|K'|} \right\rfloor \tag{1}$$

Then, for each keyword k in K', retrieve b posts that contain k from a keyword search API, ask supervisor S to label each post retrieved, and add the posts and labels to the final dataset. When this process is complete, the algorithm returns the final labeled dataset. The complete TP-KSA method is described in Algorithm 1.

Algorithm 1 TP-KSA(list of keywords K, labeling budget m, supervisor S)

```
X, y, p := SampleKeywordPosts(K, m, S)
K', p' := SelectKeywords(K, p)
b := ⌊(m−|X|)/|K'|⌋
for all keywords k in K' do
    X_k := b posts returned by querying API with keyword k
    y_k := labels from S corresponding to posts in X_k
    Add each post in X_k to X
    Add each label in y_k to y
end for
return X, y
```

Initial sampling. Keywords given to our method are first evaluated for relevance. For each keyword k in a list of keywords K, we query a keyword search API for a sample of posts that contain k but not contain any of the keywords previously sampled. We note that this creates the possibility that the sample posts for a keyword K_i may also contain one or more keywords K_j, where $j > i$. However, we do not exclude these posts from our sampling as this would decrease the likelihood of retrieving a positive (intuitively, posts with more than one keyword are more likely to be positive). The number of posts s in the sample is determined by the labeling budget m as shown in Eqn. 2. For $m > 150|K|$, this formula allocates a budget of $\lfloor 0.2m \rfloor$ for all samples, which is distributed evenly between all keywords in K. For $m \leq 150|K|$, we set a minimum sample size of 30, which is considered the minimum sample size in statistics as a "rule of thumb." In the unlikely case where $m < 30|K|$, the sample size would exceed m, so we only sample keywords until the total number of posts sampled is $0.8m$. This threshold guarantees that the entire budget m will not be spent on sampling. These values (the total sampling budget $\lfloor 0.2m \rfloor$, the minimum sample size 30, and the maximum sampling size $0.8m$) were arbitrarily selected, but our experiments show that using our proposed method with these values achieves good results for this problem.

$$s = \max\left(30, \left\lfloor \frac{0.2m}{|K|} \right\rfloor\right) \tag{2}$$

We then use supervisor S to determine whether each post in the sample is positive (relevant) or negative (not relevant). With these labels, we determine the percentage of positives for each keyword. The posts in each sample and their corresponding labels are added to the final dataset. The process of sampling keywords and determining their relevance is shown by Algorithm 2.

Algorithm 2 SampleKeywordPosts(list of keywords K, labeling budget m, supervisor S)

$s := \max\left(30, \left\lfloor \frac{m}{5|K|} \right\rfloor\right)$
$X, y, p :=$ empty lists
for all keywords k in K **do**
  if $s + |X| > 0.8m$ **then**
    break
  end if
  $X_k := s$ posts returned by querying API with keyword k, excluding posts that contain any previous k
  $y_k :=$ labels from S corresponding to posts in X_k
  $p_k :=$ percentage of positive-labeled posts in X_k
  Add each post in X_k to X
  Add each label in y_k to y
  Add p_k to p
end for
return X, y, p

Keyword selection. Using the percentage of positive posts for each keyword, we can then determine which keywords to retain. To accomplish this, we use a method inspired by the elbow method used in clustering to determine the appropriate number of clusters in a dataset. The elbow method considers some measure, e.g. the average distance between members of a cluster [29], as a function of the number of clusters k, then chooses the k corresponding to the "elbow of the curve," i.e. the point at which the curve visibly flattens with respect to the horizontal axis, with the intuition that higher values of k offer little marginal gain. Our method takes inspiration from this approach by finding the "elbow" of a curve defined by a list of keywords and their corresponding percentages of positive posts.

Given a list of keywords K and a list of percentages p, where p_i is the percentage of positive posts in a sample of posts containing keyword K_i, this method first sorts K and p according to the values of p in descending order. Next, the method plots (i, p_i) for each i and calculates the distance between each of these points and a line drawn between the first and last point, noting the index j corresponding the the point with the greatest distance under the line. The method then retains only keywords in K with a corresponding percentage of positives $p_i > p_j$, i.e. all K_i with $i < j$. In cases where $j = 1$, which indicates that all keywords except the first one and the last one are on or above the line, the method retains every keyword in K except the last one.

As an example, consider $K = \{$Keyword 1, Keyword 2, Keyword 3, Keyword 4, Keyword 5$\}$ and $p = \{0.8, 0.75, 0.4, 0.2, 0.16\}$. As shown in Fig. 2 (left), we plot (i, p_i) for each K_i in K and draw the red dotted line from the point corresponding to Keyword 1 to the point corresponding to Keyword 5 (i.e. $(1, 0.8)$ to $(5, 0.16)$). We then calculate the distance from each of these points to the red dotted line and find that $(4, 0.2)$ corresponding to Keyword 4 has the greatest distance below the line (highlighted by the blue line perpendicular to the red dotted line). We thus retain only keywords before Keyword 4.

This method of keyword selection is shown by Algorithm 3. The algorithm sorts K and p in descending order according to p, then finds the j that maximizes the distance function. The distance function calculates the distance between a point (i, p_i) and the line that intersects $(1, p_1)$ and $(|K|, p_|K|)$ such that points below

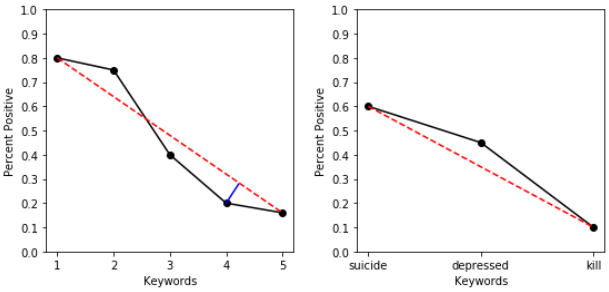

Fig. 2 TP-KSA keyword selection. Left: Keyword 4 has the greatest distance under the curve, thus the first 3 keywords are retained. Right: No keyword is under the curve, so all but the last are retained.

the line have a positive distance, points above the line have a negative distance, and points on the line have a distance of 0. Next, the algorithm sets $j = |K|$ if no point was below the line (signified by $j = 1$). The algorithm returns K' and p', which contain all K_i and p_i with a p_i higher than p_j. Note that the value p' (the percentages of positive posts corresponding to each keyword in K') is not used by TP-KSA, but is instead used in the variant methods described in Sects. 4.2.2 and 4.3.2.

Algorithm 3 SelectKeywords(list of keywords K, list of percentages p)

1: Sort K and p by descending order of the values in p
2: $j := \underset{i \in \{1,2,\ldots,|K|\}}{\text{argmax}} \frac{(p_{|K|}-p_1)(i-1)+p_i(1-|K|)+p_1(|K|-1)}{\sqrt{(p_{|K|}-p_1)^2+(1-|K|)^2}}$
3: **if** $j = 1$ **then**
4: $\quad j := |K|$
5: **end if**
6: $K' := \{K_i \ \forall \ i \in \{1, 2, \ldots, j-1\}\}$
7: $p' := \{p_i \ \forall \ i \in \{1, 2, \ldots, j-1\}\}$
8: **return** K', p'

Example: Given $K = \{$"suicide", "depressed", "kill"$\}$ and $m = 600$, TP-KSA first samples each keyword with $s = 40$ via the `SampleKeywordPosts` subroutine and determines that "suicide" is 60% positive, "depressed" is 45% positive, and "kill" is 10% positive (i.e. $p = \{0.6, 0.45, 0.1\}$). TP-KSA then continues with the `SelectKeywords` subroutine and determines $j = 3$ corresponding to the keyword "kill" as shown in Fig. 2 (right). Thus $K' = \{$"suicide", "depressed"$\}$ and each keyword in K' is allocated a budget $b = 240$, so $P = \{$("suicide", 40), ("depressed -suicide", 40), ("kill -suicide -depressed", 40), ("suicide", 240), ("depressed", 240)$\}$. "k_i -k_j" denotes a keyword query for posts that contain keyword k_i but do not contain keyword k_j.

4.2.2 Top Positive Proportional KSA (TPP-KSA) Variant

We also propose a variant to TP-KSA that aims at retrieving more positive posts by allocating a proportionally higher budget to keywords with higher percentages

of positive sample posts instead of allocating the same budget to each keyword in K'. More specifically, the method uses the list of percentages p' to determine the budget b_i for a keyword K'_i as shown in Eqn. 3.

$$b_i = \left\lfloor p'_i \frac{m - |X|}{\sum_{i=1}^{|K'|} p'_i} \right\rfloor \tag{3}$$

For each keyword K'_i in K', we then use this new budget b_i to retrieve posts containing K'_i similarly to TP-KSA. The TPP-KSA method is shown in Algorithm 4.

Algorithm 4 TPP-KSA(list of keywords K, labeling budget m, supervisor S)

```
X, y, p := SampleKeywordPosts(K, m, S)
K', p' := SelectKeywords(K, p)
for all keywords K'_i in K' do
    b_i := ⌊ p'_i (m − |X|) / (Σ_{i=1}^{|K'|} p'_i) ⌋
    X_k := b_i posts returned by querying API with keyword K'_i
    y_k := labels from S corresponding to posts in X_k
    Add each post in X_k to X
    Add each label in y_k to y
end for
return X, y
```

4.3 Extend TP-KSA to Add Diversity on the Negative Samples

While TP-KSA increases the diversity in the positive portion of its generated dataset, the posts in the negative portion each contain at least one keyword. This bias results in low diversity in that portion of the dataset. TP-KSA also does not sufficiently address the balance problem that All-Keywords and 50-50 have; the relevance of the keywords it is given remains the most significant factor in determining how well-balanced the resulting dataset is. To resolve these issues, we created TPRN-KSA, which builds upon TP-KSA by (a) discarding negative posts containing keywords to eliminate the source of bias and replacing them with randomly selected posts to add diversity to the negative samples, and (b) aiming for the same number of positives and negatives in the final dataset.

4.3.1 Random Negatives Variant of TP-KSA (TPRN-KSA)

We first assume that, for retrieval purposes, all randomly selected posts are negative. Then, to balance the positives and negatives (recall that we discard negatives returned by keyword queries), we need to compute the total number m_k of posts that should be retrieved using keywords versus the number $m - m_k$ of posts that should be retrieved randomly. Hence, we have:

$$m_k \cdot \text{average}(p') = m - m_k \tag{4}$$

where the left side of Eqn. 4 represents the target number of positive posts in the final dataset, while the right side is the number of negative (random) posts. m

is our overall labeling budget, `average`(p') is the average of all the values in p', and m_k is the budget for posts containing a keyword. Eqn. 5 shows the derived formula for m_k, which also incorporates a maximum value of $0.8m$ to ensure we avoid $m_k = m$, i.e. no budget remains for random posts.

$$m_k = \left\lfloor \min\left(\frac{m}{1 + \text{average}(p')}, 0.8m \right) \right\rfloor \tag{5}$$

We now describe our TPRN-KSA method. Like TP-KSA, it first calls the `SampleKeyWordPosts` and `SelectKeywords` subroutines. Then, using the list of percentages of positive posts p' returned by `SelectKeywords`, we calculate m_k using Eqn. 5. The algorithm then retrieves and elicits labels for b posts for each of the keywords in K' as in TP-KSA, but with a value of b that incorporates m_k:

$$b = \left\lfloor \frac{m_k - |X|}{|K'|} \right\rfloor \tag{6}$$

Next, the posts with negative labels are removed, and replaced with $m - m_k$ random posts retrieved from a keyword search API. While these random posts were assumed to be negative in our derivation of m_k, there may be positives among them in practice, thus they must be labeled by supervisor S. After these posts are labeled, they are added to the final dataset. TPRN-KSA is further described in Algorithm 5. It is also important to note that TPRN-KSA returns fewer than m labeled posts. However, we show in our experiments that despite generating dataset that is smaller compared to TP-KSA and the baselines, TPRN-KSA generates a dataset that leads to better classifier performance than those methods.

Algorithm 5 TPRN-KSA(list of keywords K, labeling budget m, supervisor S)

$X, y, p :=$ SampleKeywordPosts(K, m, S)
$K', p' :=$ SelectKeywords(K, p)
$m_k := \left\lfloor \min(\frac{m}{1+\text{average}(p')}, 0.8m) \right\rfloor$
$b := \left\lfloor \frac{m_k - |X|}{|K'|} \right\rfloor$
for all keywords k in K' **do**
    $X_k :=$ b posts returned by querying API with keyword k
    $y_k :=$ labels from S corresponding to posts in X_k
    Add each post in X_k to X
    Add each label in y_k to y
end for
Remove all negative labels from y and remove their corresponding posts from X
$X_\emptyset :=$ $m - m_k$ posts returned by querying API with $\emptyset$
$y_\emptyset :=$ labels from S corresponding to posts in $X_\emptyset$
Add each post in $X_\emptyset$ to X
Add each label in $y_\emptyset$ to y
return X, y

Example: Given $K = \{$"suicide", "depressed", "kill"$\}$ and $m = 600$, TPRN-KSA first determines K' and p'. Then, it calculates $m_k = 393$ and each keyword in K' is allocated $b = 136$ to retrieve additional posts with those keywords. Of all the retrieved posts (including those retrieved by `SampleKeywordPosts`), 180 are positive. The remaining 213 posts are removed from X and their corresponding class

labels are removed from y. TPRN-KSA then retrieves $m - m_k = 207$ random posts for a total dataset size of 387 and $P = \{(\text{“suicide”}, 40), (\text{“depressed -suicide”}, 40), (\text{“kill -suicide -depressed”}, 40), (\text{“suicide”}, 136), (\text{“depressed”}, 136), (\emptyset, 207)\}$.

4.3.2 TPPRN-KSA: Add Random Negatives to TPP-KSA

Our experiments also include a variant that combines both the TPP-KSA and TPRN-KSA variants called TPPRN-KSA. As with TPRN-KSA, this method calculates a labeling budget m_k for posts with a keyword. It then proportionally allocates budgets to each keyword in K' as in TPP-KSA, but uses m_k instead of m when calculating these budgets. For brevity, we do not include the pseudocode of this variant, but it is composed of lines 1-3 of Algorithm 5, then lines 3-9 of Algorithm 4 (with m_k replacing m in line 4), followed by lines 11-16 of Algorithm 5.

5 Experiments

In this section, we describe our experimental evaluation of our proposed method compared to several baselines.

5.1 Data

We use data from three sources in our experiments. In each experiment, posts from one of 15 message boards (DailyStrength) or one of 20 subreddits (Reddit), or news headlines from one of 35 categories (The Huffington Post) are labeled positive; all other posts/headlines in our collected data from that source are labeled negative. We discuss further details on our data collection and datasets in the Appendix. Note that while the datasets for our experiments have been downloaded in advance, this is not a requirement of our proposed method. We use these datasets in our experiments to simulate keyword search APIs from which posts are retrieved.

5.2 Baselines

We compare our proposed method to five baselines. As described in our problem definition, these methods take a list of keywords K, a labeling budget m, and a supervisor S as input. For brevity, we use “posts” here and in the remainder of the paper to refer to both posts from DailyStrength or Reddit, and news headlines and their article summaries from the Huffington Post.

- **All-Keywords:** As described in Sect. 4.1.
- **50-50:** As described in Sect. 4.1.
- **Double Ranking:** This method uses the Double Ranking method proposed by Wang et al. [30]. The keywords in K are first used to retrieve and label sample posts with Algorithm 2. The posts are then split according to their labels into X_{pos} and X_{neg}. These two datasets, along with K and a list of stopwords defined by Gensim [23], are passed to the Double Ranking algorithm, which

runs for two iterations to produce a new list of keywords K', where $|K'| = |K|$. These keywords are used to retrieve and label additional posts as in lines 3-10 of Algorithm 1. An important distinction with this method is that it retrieves much more than m posts, as the Double Ranking algorithm retrieves 300[1] posts per keyword in each iteration. However, only m posts are presented to S for labeling and added to the final dataset.

- **Active Learning:** This method uses pool-based active learning with entropy as an uncertainty measure. The initial dataset is made by iteratively querying each of the keywords in $K \cup \emptyset$ for one post one at a time until the dataset has at least one post of each label. Then, using a pool of 100,000 posts, each step of the active learning process presents the $\frac{m}{10}$ posts in the pool with the highest entropy to S for labeling and adds them to the dataset until the dataset has m labeled posts.
- **Random:** Retrieve m by querying an API with $\emptyset$ (i.e. retrieve m random posts) and present them to S for labeling, then return the labeled dataset.
- **Ideal:** Retrieve $\frac{m}{2}$ positive posts and $\frac{m}{2}$ negative posts from an API. As this method is capable of retrieving posts based on their label, it cannot be applied to a real-world setting. Instead, it serves as an approximate upper limit for comparison to the other methods.

5.3 Experiment Setup

Each experiment takes the following input:

- A KSA M
- A dataset D consisting of positives from one message board, subreddit or topic, and negatives from the remaining data from the same source
- A labeling budget m
- A number of keywords n (in all of our experiments, $n = 5$)

Each experiment is run as follows:

1. Remove a random sample from D of size $0.2|D|$ for use as test data.
2. Determine the top n words according to their information gain according to the data and labels in D; these words will be used as a list of keywords K.
3. Create a simulated keyword search API A using the data from D.
4. Run M with m and K as input. The supervisor S required by M will be simulated by the dataset's existing labels.
5. Train a text classifier C using the labeled data from M.
6. Use C to classify the test data (see Step 1) and record the results.

We then combine all of the results for one source and one method and report the average balanced accuracy [5], e.g. the average balanced accuracy across all subreddit datasets in the Reddit data for TP-KSA, for $m = 100, 200, \cdots, 1000$. This metric was chosen because accuracy can be misleadingly high when applied to the results produced from our test datasets, which are highly imbalanced toward the negative class.

These experiments are performed with both a CNN text classifier and a linear SVM with TF-IDF vectorization. Our CNN classifier splits the labeled data into

[1] This value was used by the experiments in [30].

training (80% of the data) and validation (20% of the data) datasets, performs 50 training epochs with a batch size of 100, uses window sizes of 3, 4, and 5 with 100 filters each, and uses 300-dimension fastText [4] embeddings pre-trained on Wikipedia. As we are using pre-trained embeddings, the posts selected by each KSA in our experiments do not affect the embedding space.

Our SVM experiments use the implementation in Scikit-learn [19] with 1000 features representing n-grams of sizes 1-3 with a minimum document frequency of 3% and stop words removed. All other parameters are set to their default values in Scikit-learn. Additionally, our SVM experiments are performed with 5-fold cross-validation (i.e. D is split into 5 disjoint subsets, the experiment process described above is performed 5 times with one of the subsets used as the test data, and the results are averaged).

5.4 Results – Balanced Accuracy

We split our experimental evaluation into three steps. We first compared the balanced accuracy of classifiers built from training data generated by each of the four variants of our proposed method. We then compared the best of these to the baselines. Finally, using the results of these experiments, we observe the effect of balance and diversity on classifier performance and report our findings in Sect. 5.5.

Comparison to Baselines. We first compared the 4 variants of our proposed method (see Appendix) and selected TPRN-KSA for comparison to the baseline methods. The results of the experiments with TPRN-KSA and the baselines are shown in Fig. 3. From these results we see that TPRN-KSA has higher balanced accuracy than all real-world baselines in all three sources with both SVM and CNN. The only method with higher balanced accuracy is the Ideal baseline, which has the benefit of being able to retrieve posts by their class label. All-Keywords, 50-50, Double Ranking, and Active Learning have similar balanced accuracy. Among these four baselines, All-Keywords tends to perform the best, while Active Learning tends to perform the worst. Unsurprisingly, the Random method has the lowest balanced accuracy. In the following section, we will further analyze the performance of these methods within the context of balance and diversity.

5.5 Results – Effect of Balance and Diversity on Classifier Performance

Recall that we previously stated that a classifier training dataset must achieve both balance and diversity to maximize classifier performance. More precisely, given a binary classifier trained on a training dataset T, the classifier's balanced accuracy on a test dataset should positively correlate to a combination of the balance and diversity of T. To determine this correlation, we must first quantify balance and diversity. To do so, we will use 3 additional metrics:

- **Percent Positive.** The ratio of positive posts in T.
- $\mathbf{KL_{pos}}$ and $\mathbf{KL_{neg}}$**.** The Kullback-Leibler divergence [11] of the positives (or negatives) in T from an equal number of positives (or negatives) taken from D. A bag-of-words model is used to represent each thread in these calculations.

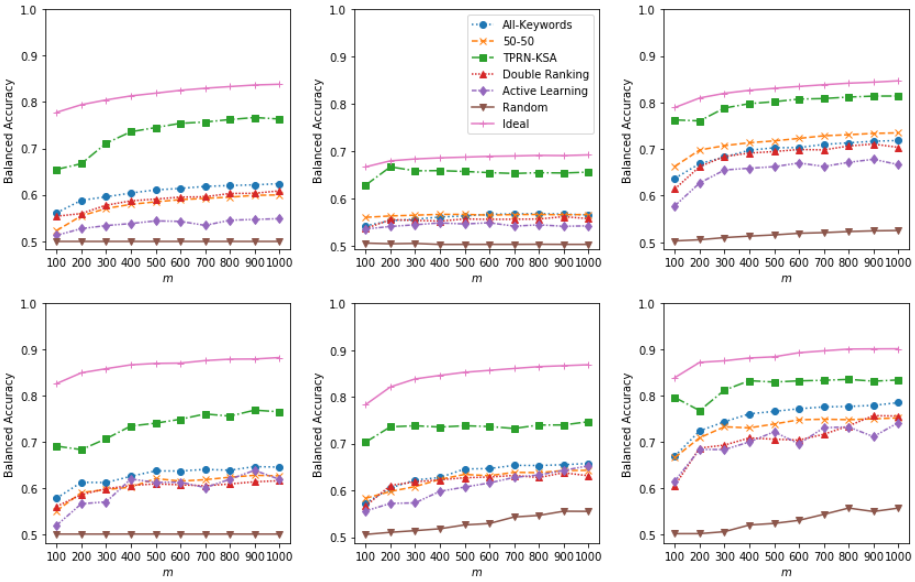

Fig. 3 Balanced accuracy of TPRN-KSA vs. baselines using SVM (top) and CNN (bottom). Left: DailyStrength. Middle: The Huffington Post. Right: Reddit.

> The divergence of part of T from a random sample with the same label acts as a measure of diversity in T, where lower values correspond to more diversity in that part of T.

Unless otherwise specified, our discussion of these metrics focuses on the SVM experiments, as the use of cross-validation makes these results more reliable. We report the averages of these metrics for each source in Fig. 4 for SVM as determined by the results of our experiments in Sect. 5.4. Note that with the exception of Active Learning, which incorporates a classifier into its selection of posts, these results differ from the CNN results only due to the use of cross-validation in the SVM experiments. We further discuss the CNN results in the Appendix.

When comparing the percent positive of the datasets generated by each method we see that TPRN-KSA is the closest to 50% positive with data from DailyStrength and The Huffington Post. With the Reddit data, the Double Ranking method is closer for some values of m and All-Keywords is above, but also close to, 50% positive. The percent positive of the Random method is very low, which is likely the most substantial contributing factor to its low balanced accuracy.

We also observed that each method's percent positive relative to other methods is generally opposite the same method's relative $\mathrm{KL}_{\mathrm{pos}}$, i.e. if method a has a higher percent positive than method b, then method a will tend to have a lower $\mathrm{KL}_{\mathrm{pos}}$ than method b. Notably, two exceptions to this are TPRN-KSA and Double Ranking, which tend to have slightly higher $\mathrm{KL}_{\mathrm{pos}}$ than All-Keywords and 50-50, respectively. This may be due to Double Ranking's retrieved positive posts becoming more topical (and thus less diverse) due to the refined keywords it uses, while the TP-KSA methods' use of a subset of K sacrifices some of the positive diversity provided by All-Keywords for better balance. The $\mathrm{KL}_{\mathrm{pos}}$ for the Random

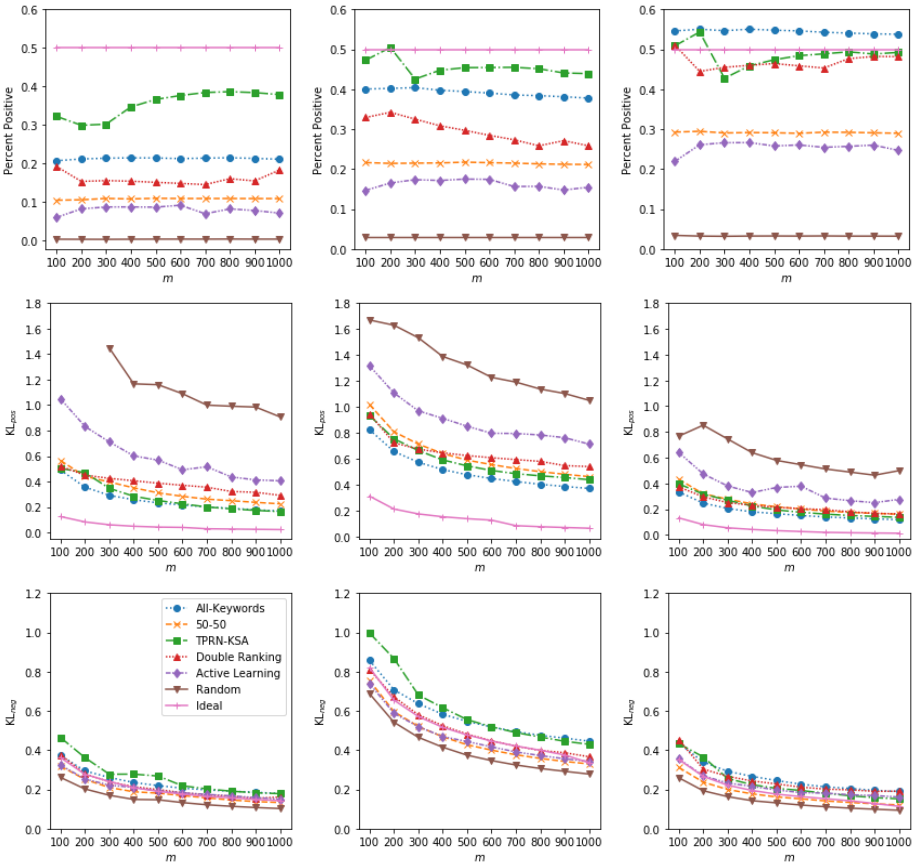

Fig. 4 Percent Positive (top), $KL_{\mathbf{pos}}$ (center), and $KL_{\mathbf{neg}}$ (bottom) as determined in the SVM experiments. Left: DailyStrength. Middle: The Huffington Post. Right: Reddit.

method is high due to its aforementioned low percentage of positive posts; with the DailyStrength dataset, the Random method's KL_{pos} is not given for $m < 300$ as there were an insufficient number of positives to calculate a meaningful result with these values of m.

The KL_{neg} of each generated dataset tends to be directly proportional to that dataset's percent positive because a higher number of positive posts means the dataset has fewer negatives and thus less opportunity for diversity. TPRN-KSA also follows this trend for smaller values of m, but its KL_{neg} falls below one or more other methods as m increases, particularly with the Reddit data. This suggests that the benefit conferred by TPRN-KSA's use of random negatives is more pronounced as m increases. We also observed that 50-50 tends to have slightly lower KL_{neg} than Active Learning, which may be explained by the fact that the negatives retrieved by 50-50 are random, while those retrieved by Active Learning are instead selected according to their entropy.

Next, we combine these three metrics and study their correlation to classifier performance. We first normalize them and then take the harmonic mean. The

intuition behind using their harmonic mean is that we want to show that a classifier tends to perform well when all three of these metrics are high, but performs worse when one or more is low. Because the harmonic mean gives more weight to smaller values, it tends to be lower when one value is low compared to the arithmetic or geometric means. The normalized balance is $B = 1-2|0.5-p|$ where p is the percent positive and has range $[0, 1]$. The normalized diversity, which is defined separately for the positive and negative portions of a generated dataset, is $D_x = e^{-\mathrm{KL}_x}$, where x is either "pos" or "neg," and has range $[0, 1]$. The harmonic mean of these three values is then defined as follows:

$$H = \left(\frac{B^{-1} + D_{\mathrm{pos}}^{-1} + D_{\mathrm{neg}}^{-1}}{3} \right)^{-1} \tag{7}$$

Using this definition, we calculated each experiment's H and compared it to the same experiment's classifier balanced accuracy to study their correlation.

Balanced Accuracy vs. Balance and Diversity: We plotted the results of each of our previous experiments for each source in two dimensions. Each experiment is represented by coordinates (H, A), where H is the harmonic mean of the balance, the diversity of the positives, and the diversity of the negatives of the training dataset generated in that experiment and A is the balanced accuracy the trained classifier achieved on the test data in that experiment. Experiments where the harmonic mean is undefined are excluded from our analysis. We show the plotted SVM experiments in Fig. 5. We also show the Pearson correlation coefficient [18] of balanced accuracy and the harmonic mean of balance and diversity for each source in Table 2. All three sources show that a classifier's balanced accuracy is strongly correlated with the balance and diversity of the dataset that was used to train the classifier. However, we also note that these correlations vary substantially between sources and classifiers; this suggests that while balance and diversity clearly play a significant role in classifier performance, it may be affected by other factors as well.

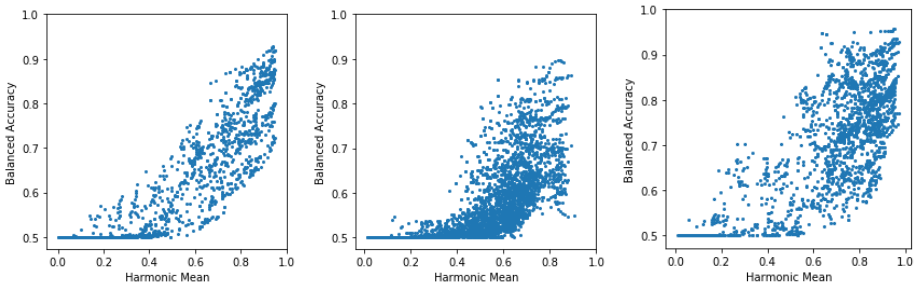

Fig. 5 Correlations between balanced accuracy and the harmonic mean of balance and diversity with SVM. Left: DailyStrength. Middle: The Huffington Post. Right: Reddit.

Limitations: Despite the performance of TPRN-KSA, it has several limitations. First, its sampling of posts to determine the percentage of positives for each keyword (Algorithm 2) does not account for the fact that a keyword may appear in a post that was sampled using another keyword. Accounting for this could reduce

Table 2 Correlation of classifier balanced accuracy and the harmonic mean of balance and diversity of classifier training data

Source	n	r (SVM)	p (SVM)	r (CNN)	p (CNN)
DailyStrength	1500	0.8629	< 0.001	0.8223	< 0.001
The Huffington Post	3500	0.6610	< 0.001	0.7715	< 0.001
Reddit	2000	0.7411	< 0.001	0.6807	< 0.001

the amount of the budget used for sampling. Another issue with the sampling of posts is its use of arbitrary values; specifically, reserving 20% of m for sampling, applying the "rule of 30" for determining minimum sample sizes, and imposing a maximum sample size of $0.8m$ for small values of m. However, the results of additional experiments in the Appendix suggest that the choice of these values within reasonable ranges generally do not have a significant impact on classifier performance. Finally, TPRN-KSA's keyword selection (Algorithm 3) is inspired by the elbow method, which is a generally unreliable means of determining the number of clusters in a dataset. A more principled approach to this method may lead to better results.

6 Conclusions

We proposed the training post retrieval over constrained search interfaces problem. We also proposed a method to address this problem, TPRN-KSA. This method is built on the assumption that balance and diversity in a training dataset positively affect the balanced accuracy of a classifier trained with the data. TPRN-KSA outperformed all other variant methods and several baselines in our experiments. For $m = 1000$, TPRN-KSA has an improvement of 13.96%, 8.95%, and 7.91% with SVM and 11.90%, 8.94%, and 4.92% with CNN over the best baseline for DailyStrength, The Huffington Post, and Reddit, respectively. We followed up these experiments with an analysis on the correlation between classifier balanced accuracy and the harmonic mean of the balance and diversity of training data. We found that they were positively correlated, supporting our initial assumption. Future work may address the limitations in TPRN-KSA as discussed above by improving the sampling behavior and proposing new methods for keyword selection, e.g. by incorporating other metrics.

References

1. Ahmad S, Asghar MZ, Alotaibi FM, Awan I (2019) Detection and classification of social media-based extremist affiliations using sentiment analysis techniques. Hum Cent Comput Inf Sci 9:24.
2. Balsamo D, Bajardi P, Panisson A (2019) Firsthand opiates abuse on social media: Monitoring geospatial patterns of interest through a digital cohort. In: Proc WWW 2019, pp 2572–2579.
3. Bissoyi S, Mishra BK, Patra MR (2016) Recommender systems in a patient centric social network—A survey. In: Proc SCOPES 2016, pp 386–389.
4. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5:135-46.

5. Brodersen KH, Ong CS, Stephan KE, Buhmann JM (2010) The balanced accuracy and its posterior distribution. In: Proc ICPR 2010, pp 3121–3124.
6. Croft WB, Metzler D, Strohman T (2010) Search engines: Information retrieval in practice. Addison-Wesley, Boston.
7. de Lira VM, Macdonald C, Ounis I, Perego R, Renso C, Times VC (2019) Event attendance classification in social media. Inform Process Manag 56(3):687–703.
8. Elkan C, Noto K (2008) Learning classifiers from only positive and unlabeled data. In: Proc SIGKDD 2008, pp 213–220.
9. Goudjil M, Koudil M, Bedda M, Ghoggali N (2018) A novel active learning method using SVM for text classification. Int J Automat Comput 15(3):290–298.
10. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proc EMNLP 2014, pp 1746–1751.
11. Kullback S, Leibler R (1951) On information and sufficiency. Ann Math Statist 22(1):79–86.
12. Li C, Xing J, Sun A, Ma Z (2016) Effective document labeling with very few seed words: A topic model approach. In: Proc CIKM 2016, pp 85–94.
13. Li C, Zhou W, Ji F, Duan Y, Chen H (2018) A deep relevance model for zero-shot document filtering. In: Proc 56th Annu Meeting ACL, pp 2300–2310.
14. Li H, Liu B, Mukherjee A, Shao J (2014) Spotting fake reviews using positive-unlabeled learning. Comput Sist 18(3):467–475.
15. Li R, Wang S, Cheng KC (2013) Towards social data platform: Automatic topic-focused monitor for Twitter stream. Proc VLDB Endowment, 6(14):1966–1977.
16. Li X, Liu B (2003) Learning to classify texts using positive and unlabeled data. In: Proc IJCAI 2003, pp 587–592.
17. Misra R (2018) News Category Dataset. ResearchGate. https://doi.org/10.13140/RG.2.2.20331.18729
18. Pearson K (1895) Note on regression and inheritance in the case of two parents. Proc. Royal Soc. London 58(347-352):240–242.
19. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830.
20. Pohl D, Bouchachia A, Hellwagner H (2018) Batch-based active learning: Application to social media data for crisis management. Expert Syst Appl 93:232–244.
21. Proskurnia J, Mavlyutov R, Castillo C, Aberer K, Cudre-Mauroux P (2017) Efficient document filtering using vector space topic expansion and pattern-mining: The case of event detection in microposts. In: Proc CIKM 2017, pp 457–466.
22. Rao J, Yang W, Zhang Y, Ture F, Lin J (2019) Multi-perspective relevance matching with hierarchical convnets for social media search. In: Proc 33rd AAAI Conf Artif Intell, pp 232–240.
23. Řehůřek R, Sojka P (2010) Software framework for topic modelling with large corpora. In: Proc LREC 2010 Workshop New Challenges NLP Frameworks, pp 45–50.
24. Rivas R, Sadah SA, Guo Y, Hristidis V (2020) Classification of health-related social media posts: Evaluation of post content classifier models and analysis of user demographics. JMIR Public Health Surveill 6(2):e14952.
25. Ruiz E, Hristidis V, Ipeirotis PG (2014) Efficient filtering on hidden document streams. In: Proc ICWSM 2014.
26. Sadri M, Mehrotra S, Yu Y (2016) Online adaptive topic focused tweet acquisition. In: Proc. CIKM 2016, pp 2353–2358.
27. Shen S, Murzintcev N, Song C, Cheng C (2017) Information retrieval of a disaster event from cross-platform social media. Inf Discov Deliv 45(4):220–226.
28. Smailovic J, Grcar M, Lavrac N, Znidarsic M (2014) Stream-based active learning for sentiment analysis in the financial domain. Inf Sci 285:181–203.
29. Thorndike RL (1953) Who belongs in the family? Psychometrika 18(4):267–276.
30. Wang S, Chen Z, Liu B, Emery S (2016) Identifying search keywords for finding relevant social media posts. In: Proc 30th AAAI Conf Artif Intell, pp 3052–3058.
31. Zhang Y, Lease M, Wallace BC (2017) Active discriminative text representation learning. In: Proc 31st AAAI Conf Artif Intell, pp 3386–3392.
32. Zhang Y, Zhao P, Cao J, Ma W, Huang J, Wu Q, Tan M (2018) Online adaptive asymmetric active learning for budgeted imbalanced data. In Proc SIGKDD 2018 pp 2768–2777.