Optimizing Search API Queries for Twitter Topic Classifiers
Using A Maximum Set Coverage Approach

Kasra Safari
University of Toronto
kasra.safari@mail.utoronto.ca

Scott Sanner
University of Toronto
ssanner@mie.utoronto.ca

ABSTRACT
Twitter has grown to become an important platform to access immediate information about major events and dynamic topics. As one example, recent work has shown that classifiers trained to detect topical content on Twitter can generalize well beyond the training data. Since access to Twitter data is hidden behind a limited search API, it is impossible (for most users) to apply these classifiers directly to the Twitter unfiltered data streams (“firehose”). Rather, applications must first decide what content to retrieve through the search API before filtering that content with topical classifiers. Thus, it is critically important to query the Twitter API relative to the intended topical classifier in a way that minimizes the amount of negatively classified data retrieved. In this paper, we propose a sequence of query optimization methods that generalize notions of the maximum coverage problem to find the subset of query terms within the API limits that cover most of the topically relevant tweets without sacrificing precision. We evaluate the proposed methods on a large dataset of Twitter data collected during 2013 and 2014 labeled using manually curated hashtags for eight topics. Among many insights, our analysis shows that the best of the proposed methods can significantly outperform the firehose on precision and F1-score while achieving high recall within strict API limitations.

CCS CONCEPTS
• Information systems → Web and social media search;

KEYWORDS
Social Media Search; Data Collection; Query Optimization; Greedy Heuristics; Mixed Integer Linear Programming

1 INTRODUCTION
Recent work has shown that machine learning classifiers trained to detect topical content on Twitter (e.g., content relevant to “natural disasters”) can generalize well beyond the training data and provide stable performance over long time horizons [8]. Such trained topic classifiers can be easily used to filter and rank content relevant to the given topic. For example, consider the use case that a user wants to query a social media platform such as Twitter to follow content on a specific topic (e.g. natural disasters). Every day or even more frequently, the user would like to check for the latest tweets relevant to the topic in real-time and they would like to do so with high recall (i.e., they do not want to miss important content) as well as high precision (i.e., to make best use of their limited time to browse ranked search results).

Under real-time requirements where one does not have access to the Twitter “firehose” (full, unfiltered data streams) and is limited by query restrictions of the Twitter’s free APIs, one is faced with two main choices: filter a relatively small — roughly 1% of the Twitter content (assumed random) [19] — data sample from the Twitter streaming API, or query for a subset of tweets using the Twitter search API.

But how can one define search queries for a given topic classifier to achieve high content coverage (i.e., recall)? A common way to collect tweets is to use a few manually selected terms or hashtags to construct the search queries [20, 24]. However, such predefined queries are subjective in nature and often lead to low recall [26].
We use a two-stage learning framework to address this problem for topical classifiers (Fig. 1). At the first stage, we either have access to the full unfiltered data (which we refer to as the “firehose” data) or the data is retrieved by querying the rate-limited Twitter search API\(^1\). The API allows queries of 500 characters maximum which includes boolean OR operators. In the second stage, a classifier model is trained on the data retrieved in the first stage to produce the final classifications and rankings of the tweets. The first stage aims for a high recall recall of relevant results, and the retrieved data is used to train a model which ranks the relevant tweets with the goal of maximizing precision at the top ranks. Our main focus in the paper is to improve the quality of the data retrieved in the first stage in a limited and constrained setting (limitations in the amount of retrieved data and the size/rate of the queries by the API). It has been shown that improving the quality of the first stage retrieval can have significant results on the performance of the classifier [4].

To address the question of optimal query construction, we use a sequence of query optimization methods that generalize notions of the maximum coverage problem [9] to find the subset of query terms within the API limits that cover most of the topically relevant tweets. We formulate the methods as Integer Linear Programs (ILP) and solve the ILPs to get the desired queries. To evaluate our proposed methods we run experiments over a large corpus of Twitter data collected during 2015 and 2014 with 8 different trained topic filters. We use the entire dataset as the firehose and simulate a boolean filtering search API interface to the data for evaluation purposes. We show simple greedy strategies perform nearly as well as optimal ILP based solutions and that the best of the proposed methods can significantly outperform the firehose on precision and F1-score for the first stage while achieving high recall within strict API limitations compared to the unlimited firehose API. We also show that while using our proposed methods results in significantly smaller data, the performance of the second stage classifier is similar to that of a classifier trained on the complete firehose data.

While this work provides methods for high precision and recall optimization of Twitter search API queries w.r.t. topic classifiers, we remark that it also has many potential applications to other data services that do not make their complete data available but also provide search APIs for limited access to their available content. Most social media platforms support keyword search for obtaining information but the main challenge is that there are limits imposed on the amount and frequency of data that can be obtained from these APIs [2]. Even in a situation where there are no limitations on the API, it is more desirable to start any task with a dataset with higher quality, i.e., having a medium-sized dataset with balanced positive-negative classes instead of a huge dataset containing a small minority of positive data. As a final remark, we note the proposed methods only require a labeled subset of the data to optimize the query API keywords and hence are agnostic to the classification technique.

\(^1\)https://developer.twitter.com/en/docs/tweets/search/overview

2 BACKGROUND

2.1 Task Definition

Our main focus in this paper is the improving the task of the first stage in the two-stage classification framework, which can be defined as follows: Given a collection of binary labeled tweets, we wish to derive a small set of keywords \(K = \{k_1, k_2, k_3, \ldots, k_n\}\) (where \(n\) is the maximum number of keywords allowed by the API) that when used in a boolean OR query, will match a balanced set of tweets with a high recall of the positive labeled tweets. The query keywords can be account usernames or terms, hashtags and locations of tweets. Since topics of interest are usually dynamic and cover a broad range of subjects and events, it is important for our methods to be scalable and adaptive to the changes in the nature of the relevant content.

The importance of the recall in the first stage is that while we can expect the second stage classifier to weed out the irrelevant data, failure to retrieve enough relevant data in the first stage will affect the performance of the second stage negatively. However, we can’t neglect the importance of precision at the first stage because there are rate limits \(^2\) for these APIs that prevent us from retrieving large amounts of data in a short time. So it is also important to consider a relatively high precision as a secondary objective in the first stage filtering.

2.2 Maximum K-Coverage Problem

Our main goal of retrieval of positively labeled tweets given the imposed limitations of the query API is similar to that of the well-known maximum k-coverage problem from combinatorial optimization, which we now review.

Definition 2.1 (Maximum k-Coverage Problem). Given a collection \(S\) of sets over a domain of elements \(E\), find a subset \(S' \subseteq S\) of sets, such that \(|S'| \leq k\) and the number of covered elements

\[
\left| \bigcup_{S_i \in S'} S_i \right|
\]

is maximized.

The task of querying a limited API can be reduced to the maximum k-cover problem. For the rest of the paper, we may refer to the possible keywords (terms, hashtags, etc.) in a query as features and use these terms interchangeably. \(F\) is the set of features in the available tweet corpus and for each feature \(F_i \in F\), we show the tweets which contain \(F_i\) (i.e. the tweets that would be returned by a boolean filtering API if we only queried for \(F_i\)) with \(\text{cov}(F_i)\).

Definition 2.2 (Maximum K-Coverage of Tweets). Given an integer \(k > 0\), \(F' \subseteq F\) is a maximum k-cover of tweets if \(|F'| \leq k\) and the number of covered tweets \(\text{cov}(\bigcup_{F_i \in F'} F_i)\) is maximized.

By modeling the set of tweets as the element set and the coverage of features as the sets, we can formulate the filtering task as a maximum set coverage problem and find solutions to this task by solving the corresponding max cover problem.

\(^2\)https://developer.twitter.com/en/docs/basics/rate-limits
3 MILP FORMULATIONS FOR QUERY OPTIMIZATION

The maximum k-cover problem is an NP-hard problem and can be approximated using either greedy or more sophisticated methods [9]. We use two ways to solve the maximum coverage problems in this paper:

(1) Linear Programming Solvers One option is to use the available optimization tools for linear programming problems to solve the ILP formulation. These solvers generally use sophisticated algorithms such as branch and bound [11] to give the most accurate results for the problems.

(2) Greedy Algorithm in which the solution is constructed in an iterative manner by selecting the feature with the most value for the greedy heuristic function (e.g. maximum coverage of uncovered tweets). This algorithm is known to give a $1 - \frac{1}{e}$-approximation solution [7] with an advantage of faster running time due to the simplicity of the algorithm. A high level description of the implemented algorithm is shown in Algorithm 1.

Algorithm 1 Greedy Algorithm for Maximum k-Coverage

1: $i \leftarrow 0$
2: output $\leftarrow []$
3: while $i < k$ do
4: $F^* \leftarrow$ Feature with Maximum Increase In Objective Function
5: Remove All Tweets Covered By $F^*$ From the Coverage Data
6: Add $F^*$ to output
7: $i \leftarrow i + 1$

Our main goal is to maximize the coverage of the topical tweets and therefore, topical tweet coverage should be the main element in the objective functions across all possible formulations. However, does using additional information like the coverage of non-topical tweets by each feature and the mutual information score of each feature help us in getting queries with higher qualities? To test this, we defined different formulations to observe the effect of each type of information in the formulations.

We use $P$ to indicate the presence of tweets labeled positively (topical tweets) in the dataset. Similarly, $N$ is used for non-topical tweets. Binary variables $p_i$ and $n_i$ are used to indicate the presence of each tweet in the queried data. If $p_i = 1$ topical tweet $P_i \in P$ is covered by the query. The same relationship holds for $n_i$ and non-topical tweet $N_i \in N$. To represent the selected features in the solution, we use binary variable $f_i$ in the formulations. if $f_i = 1$ then feature $F_i \in F$ is present in the query.

3.1 Coverage-based ILP Formulation (CILP)

For the simplest formulation, we only use the coverage of topical tweets by each feature:

\[
\text{maximize } f \sum_{i=1}^{[P]} p_i \text{ subject to } \bigvee_{(j : p_i \in \text{cov}(F_j))} f_j \leq k \quad p_i \in \{0, 1\}, \ i = 1, \ldots, [P] \\
\quad f_j \in \{0, 1\}, \ j = 1, \ldots, [F]
\]

The first constraint is for defining the relation between each covered tweet and the features. For a tweet $P_i$ to be covered in the solution, at least one of the features that covers the tweet should be selected in the query, hence the disjunction.

3.2 Weighted ILP Formulation with Mutual Information Scores (WILP)

While the CILP formulation offers the most simplicity, a possible issue is that we have a considerable amount of information on non-topical data not used. A feature may be selected by this model that covers a lot of the topical tweets while also covering a lot of negative tweets which is not desirable.

Mutual Information [3] is a quantified measure of similarity between two random variables in the data. In the context of retrieving topical tweets, a high value of mutual information score between the presence of a feature in each tweet and the label of that tweet can be an indicator of the quality of retrieved data by querying on that feature. In other words, we can use the mutual information score of a feature to determine if using that feature in the query can help us retrieve a larger number of topical tweets without getting a lot of non-topical data.

In this formulation, we use the mutual information score of the features selected in the solution as an additional term in the decision function. In this way, selecting features with high mutual information scores is rewarded by an increase in the decision function. The probability distributions needed to compute the mutual information scores are estimated using the available data.

The only challenge is to normalize and weight the two different metrics (coverage and mutual information) to make the addition in the decision function effective. We use a normalized value of the score which scales the scores to values between 0 (no mutual information) to 1 (perfect correlation). We also normalize the total coverage of the solution by dividing by the total number of tweets ($[P]$) so that both of the terms in the objective function be in range $[0, 1]$.

Still, the coverage data and the mutual information scores may not have equal effects on the solution. We introduce an additional hyperparameter $\lambda$ in the objective function which determines the relative importance of the mutual information scores in the objective function compared to the positive coverage. Below is the resulting ILP formulation:
To include the non-topical tweets. The resulting formulation is:

$$\text{maximize} \quad f \left( \frac{\sum_{i=1}^{p} p_i}{|P|} + \lambda \sum_{j=1}^{f_j} M_{ij} \right)$$

subject to $$\bigvee_{\{j : p_i \in \text{cov}(F_j)\}} f_j = p_i, \quad \bigvee_{\{j : n_i \in \text{cov}(F_j)\}} f_j = n_i,$$

$$\sum_{j=1}^{f_j} f_j \leq k, \quad p_i \in \{0, 1\}, \quad i = 1, \ldots, |P| \quad n_i \in \{0, 1\}, \quad i = 1, \ldots, |N|$$

In which $M_{ij}$ is the normalized mutual information score of feature $F_j$.

Determining the value of $\lambda$ is a hyperparameter tuning task. In our case, we tuned $\lambda$ to maximize the F1-score on the validation data.

### 3.3 Coverage/Anti-coverage Based ILP Formulation (CAILP)

Calculating the mutual information score for all the features can be an expensive and time-consuming task and as the problem space grows, the required time to calculate the scores increases. We can either continue using the mutual information scores or we can use less expensive alternatives with similar or improved results.

One of these alternatives is to use the coverage of non-topical tweets by each feature the same way we utilized the topical coverage. Instead of estimating the importance of a feature with the mutual information score, we tackle the problem of determining the feature importance head-on using each feature’s coverage of positive and negative labeled tweets. Selecting a feature that covers a lot of non-topical tweets is penalized by subtracting the non-topical coverage of the feature from the objective function.

For a more tidy formulation, we expand the definition of $\text{cov}(F_j)$ to include the non-topical tweets. The resulting formulation is:

$$\text{maximize} \quad f \left( \frac{\sum_{i=1}^{p} p_i}{|P|} - \lambda \sum_{i=1}^{n} n_i \right)$$

subject to $$\bigvee_{\{j : p_i \in \text{cov}(F_j)\}} f_j = p_i, \quad \bigvee_{\{j : n_i \in \text{cov}(F_j)\}} f_j = n_i,$$

$$\sum_{j=1}^{f_j} f_j \leq k, \quad p_i \in \{0, 1\}, \quad i = 1, \ldots, |P| \quad n_i \in \{0, 1\}, \quad i = 1, \ldots, |N|$$

Similar to the WILP formulation, the $\lambda$ hyperparameter is used to properly balance the two metrics in the objective function for best results.

### 4 EXPERIMENTAL SETUP

#### 4.1 Data Description

The data used in the following experiments is the crawled Twitter data retrieved using the Twitter streaming API for two years from 2013 to 2014.

Five general types of features present in each tweet are used as possible keywords for querying: (1) From the username of the tweet sender (2) Location provided by the users in their profiles (3) Hashtag set of hashtags in each tweet starting with "#" (if any) used as a metadata tag by the users (4) Mention usernames of the users referenced in the tweet text with the "@" sign (5) Term terms in each tweet which are not either mentions or hashtags. Since we have implemented a boolean filtering API module with these features in mind, we use all the mentioned features in the evaluation stage to observe the importance of each feature type in the solutions. However, it should be noted that at the time of writing this paper, the Twitter Search API does not accept the location feature in this form. We can limit the solvers to omit certain feature types that are not currently supported by the official API.

Since the data is not labeled in its raw form, we need a method to properly label the data to be used with the proposed methods. Considering the large size of the data manually labeling the tweets is practically impossible. Similar to the labeling methods in [8, 14, 17] the data is labeled for each of eight different topics using a set of user-curated hashtags (referred to as "labeling hashtags") specific to each topic. For each topic, a set of labeling hashtags are selected by four independent human annotators and each hashtag requires an inner-annotator agreement between three annotators to be included in the labeling hashtags collection used for labeling the tweets. A subset of the hashtags used for labeling the tweets for each topic is shown in Table 1 alongside the number of positive labeled tweets for each topic. It should be noted that our proposed methods are independent of the labeling strategy and can be applied to any labeled dataset with a different labeling method.

During the preprocessing stage, duplicate tweets are removed from the dataset since they do not add any new information to the solution and are redundant. At this stage, we are mainly interested in English tweets and features (for more interpretability in the selected keywords) which is why we remove any non-English tweets from the data during the preprocessing. Since we use hashtags to label the tweets, tweets without any hashtags are also removed from the data since it’s not possible to label them. Our dataset contains 135, 910, 871 English tweets after the preprocessing stage.

Due to the large size of the dataset, the size of the candidate keyword set for the queries can become huge (17, 996, 431 unique features). Since most of these features occur sparingly in the tweets, they would not be useful in our problem. It is important to apply frequency thresholds on the candidate query keywords to pare down the size of the candidate keyword set to a more reasonable size without losing the important features for maximizing the coverage of the tweets. We found that a frequency threshold of 100 resulted in a reasonable feature set size of 73, 887 without removing the important features.

---

1. https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html
Table 1: Topics with five labeling hashtags and the resulting statistics of positive labeled tweets. The hashtags are selected by four independent human annotators, requiring an inner-annotator agreement of three annotators to permit a hashtag to be assigned to a topic set.

| Topics          | #TopicalTweets |
|-----------------|----------------|
| Natural Disasters | 89,440         |
| Social Issues    | 374,710        |
| Space            | 409,817        |
| Soccer           | 1,377,787      |
| Human Disasters  | 792,268        |
| Tennis           | 86,108         |
| Health           | 401,362        |
| LGBT             | 632,882        |

Sample Hashtags
- #julio
- #tsunami2004
- #chileearthquake
- #houston
- #fifa
- #halamadrid
- #englandsoccercup
- #redefinenigeria
- #nadal
- #wimbledon2013
- #uk
- #chanyeolvirussday
- #uniteblue
- #chikungunya
- #active
- #maratynadal
- #drought13
- #44millionabortions
- #asteroids
- #beckham
- #mh17
- #notimynname
- #usa
- #malaysiaairlines
- #obamaina
- #bombsquad
- #malaysianairlines
- #naturaldisasters
- #soccer
- #terrorism
- #astronauts
- #beckham
- #ebolaresponse
- #equalitynow
- #messi
- #mh17
- #malaysiaairlines
- #nadal
- #notimynname
- #usa
- #naturaldisasters
- #soccer
- #terrorism
- #astronauts
- #beckham
- #ebolaresponse
- #equalitynow
- #messi
- #mh17
- #malaysiaairlines
- #nadal
- #notimynname
- #usa
- #naturaldisasters
- #soccer
- #terrorism
- #astronauts
- #beckham
- #ebolaresponse
- #equalitynow
- #messi
- #mh17
- #malaysiaairlines
- #nadal
- #notimynname
- #usa

Figure 2: Overview of the evaluation framework. The data is randomly divided into five splits. The smaller split used for deriving the query and evaluating the classifier and the rest of the data filtered and used for training the classifier.

4.2 Evaluation Framework

Twitter recommends using ten keywords in each query. Given the fact that the maximum length of a query can be 500 characters and considering the fact that the API rejects overly complex queries, we set $K = 20$ as the maximum number of features selected by our methods (25 characters per keyword). We use 5-fold cross validation to evaluate the performance of our proposed methods. The tweets are randomly divided into five splits. At each iteration, we use one of the splits as the input for the query optimization component in Fig. 1. A query is derived using the data in this split which is used to filter the data in the rest of the data (consisting of the four remaining splits). The performance of the classifier is evaluated using the data from the initial smaller split. An overview of the evaluation process is depicted in Fig. 2. We use a logistic regression classifier as the ranking model in the second stage.

Recall = \(# \text{ of Topical Tweets Retrieved} \) / \(# \text{ of Topical Tweets In Test Split} \)

Precision = \(# \text{ of Topical Tweets Retrieved} \) / \(# \text{ of Tweets Retrieved} \)

F1 Score = \[ \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \]

In addition to the three proposed methods, we evaluated the performance on the unfiltered data ("firehose") and also on an additional method filtering the data using the top weighted features in a logistic regression classifier trained on the unfiltered firehose data (referred to as the "TopK" method) as the baselines for comparing the performance results of the ILP-based methods.

We use average precision and precision@100 to evaluate the performance in the second stage of the classifier. Since the output of the second stage classifier is presented to the user, it is important to have high quality results in the top ranks of the output which is why we evaluate the performance at this stage with more rank-centric metrics. On the other hand, we use boolean-based evaluation metrics to evaluate the performance in the first stage since the ordering of the results is not important as much as the presence of relevant data in the filtered data.

5 RESULTS AND DISCUSSION

5.1 Performance Comparison of Query Optimization Methods

The results for the filtered data obtained using the methods at the first stage are shown in Table 2. The values in Table 2 reflect the 95% confidence interval calculated by applying each method on the five splits of our tweet data. The ILP problems for the formulations are solved using the Gurobi solver which has the best performance among the available LP solvers.

Observing the results, the performance on the firehose data has perfect recall with very low precision (mean precision of 0.004 across all topics) which results in small F1-scores (0.007 mean F1-score across all topics). These results are expected when using all

5http://www.gurobi.com
Table 2: Performance of the Query Optimization Methods for Different Topics (Decimals rounded to 3 digits). Firehose does not use a restricted Query API and is simply offered for benchmark comparison; best results among TopK, CILP, WILP, and CAILP are bolded.

| Topic                  | TopK       | CILP       | WILP       | CAILP      |
|------------------------|------------|------------|------------|------------|
| Natural Disaster        | 5.24       | 0.088      | 0.068      | 0.057      |
| Topical Retrieved (%)   | 0.057      | 0.091      | 0.090      | 0.083      |
| Total Retrieved (%)     | 0.134      | 0.136      | 0.136      | 0.136      |
| Recall (%)              | 0.048      | 0.048      | 0.048      | 0.048      |
| Precision (%)           | 0.486      | 0.486      | 0.486      | 0.486      |
| F1-Score                | 0.057      | 0.057      | 0.057      | 0.057      |
| Social Issues           | 0.001      | 0.000      | 0.000      | 0.000      |
| Firehose                | 0.159      | 0.160      | 0.160      | 0.160      |
| Topical Retrieved (%)   | 0.083      | 0.083      | 0.083      | 0.083      |
| Total Retrieved (%)     | 0.134      | 0.134      | 0.134      | 0.134      |
| Recall (%)              | 0.048      | 0.048      | 0.048      | 0.048      |
| Precision (%)           | 0.486      | 0.486      | 0.486      | 0.486      |
| F1-Score                | 0.057      | 0.057      | 0.057      | 0.057      |
| Sports                  | 0.061      | 0.061      | 0.061      | 0.061      |
| Firehose                | 0.252      | 0.252      | 0.252      | 0.252      |
| Topical Retrieved (%)   | 0.068      | 0.068      | 0.068      | 0.068      |
| Total Retrieved (%)     | 0.136      | 0.136      | 0.136      | 0.136      |
| Recall (%)              | 0.048      | 0.048      | 0.048      | 0.048      |
| Precision (%)           | 0.486      | 0.486      | 0.486      | 0.486      |
| F1-Score                | 0.057      | 0.057      | 0.057      | 0.057      |

5.2 Features Selected By Methods

A better way to understand the behaviour of each method is to look at the top features each method selects as the query keywords.

We show the most frequent keywords showing up when deriving the keywords using different splits for each method in Table 4 for a subset of topics (we could not show all of the topics due to the space limitations).

Looking at the results, the keywords selected by the TopK are generally those that have the highest correlation with the target label being positive. The fact that most of the top keywords learned by the classifier are relevant to the topic is a good sign that our classifier does actually learn about the target topic for the classification. However, high correlation with the topic doesn’t necessarily result in a high coverage of the relevant data. As an example, in the case of the topic Soccer, we see that @lfc and #ynwa are the top ranked features. The mentioned account and the hashtag are famously associated with Liverpool FC 6 and using both of these keywords in a limited query would not result in a diverse filtered dataset. The problem is that we want a more generalized selection of the features to ensure the retrieval of relevant content at any time.

The top features for the CILP method show us the reason for the coverage statistics in Table 2. Since we only consider the coverage of the positive-labeled tweets in this method, the method naturally picks up common words such as "the", "in". These keywords do cover a high number of tweets labeled topical (resulting in high recalls) but they also cover a lot of non-topical data (resulting in low precisions). One might argue that we can remove these known keywords in the data preprocessing stage, but we expect any reasonable filtering method to filter these keywords automatically. Additionally, there are cases where the stopwords become domain specific e.g. "rt" (indicator for retweeted tweets) is a common keyword specific to Twitter.

https://en.wikipedia.org/wiki/Liverpool_F.C.
Table 3: Performance of the query optimization methods in the second stage of the two-stage classification framework for different topics (decimals rounded to 3 digits). Best results among the methods are bolded.

| Model      | Test AveP  | P@100      |
|------------|------------|------------|
| **Natural Disaster** |            |            |
| Firehose   | 0.417 ± 0.004 | 0.772 ± 0.022 |
| TopK       | 0.373 ± 0.004 | **0.928 ± 0.044** |
| CILP       | 0.417 ± 0.004 | 0.750 ± 0.032 |
| WILP       | 0.352 ± 0.101 | 0.754 ± 0.096 |
| CAILP      | **0.434 ± 0.009** | 0.830 ± 0.057 |
| **Social Issues** |            |            |
| Firehose   | 0.678 ± 0.003 | 0.730 ± 0.035 |
| TopK       | 0.574 ± 0.005 | **0.741 ± 0.029** |
| CILP       | **0.684 ± 0.003** | 0.733 ± 0.035 |
| WILP       | 0.603 ± 0.004 | 0.523 ± 0.018 |
| CAILP      | 0.624 ± 0.005 | 0.714 ± 0.026 |
| **Space**  |            |            |
| Firehose   | 0.432 ± 0.002 | **0.746 ± 0.014** |
| TopK       | 0.533 ± 0.019 | 0.665 ± 0.016 |
| CILP       | **0.871 ± 0.003** | 0.662 ± 0.008 |
| WILP       | 0.862 ± 0.005 | 0.652 ± 0.022 |
| CAILP      | 0.794 ± 0.018 | 0.669 ± 0.013 |
| **Soccer** |            |            |
| Firehose   | 0.646 ± 0.001 | 0.700 ± 0.028 |
| TopK       | 0.599 ± 0.006 | 0.798 ± 0.037 |
| CILP       | 0.648 ± 0.001 | 0.704 ± 0.029 |
| WILP       | **0.651 ± 0.001** | 0.702 ± 0.036 |
| CAILP      | 0.599 ± 0.010 | **0.858 ± 0.016** |
| **Human Disasters** |         |            |
| Firehose   | 0.734 ± 0.002 | 0.602 ± 0.027 |
| TopK       | 0.524 ± 0.014 | 0.672 ± 0.016 |
| CILP       | 0.379 ± 0.002 | 0.602 ± 0.033 |
| WILP       | **0.746 ± 0.003** | 0.614 ± 0.026 |
| CAILP      | 0.694 ± 0.006 | **0.684 ± 0.017** |
| **Tennis** |            |            |
| Firehose   | 0.851 ± 0.004 | 0.910 ± 0.015 |
| TopK       | 0.682 ± 0.021 | 0.922 ± 0.027 |
| CILP       | **0.853 ± 0.004** | 0.910 ± 0.012 |
| WILP       | 0.840 ± 0.039 | 0.904 ± 0.007 |
| CAILP      | 0.787 ± 0.026 | **0.940 ± 0.025** |
| **Health** |            |            |
| Firehose   | 0.532 ± 0.002 | 0.732 ± 0.033 |
| TopK       | 0.439 ± 0.012 | 0.782 ± 0.016 |
| CILP       | **0.537 ± 0.001** | 0.738 ± 0.032 |
| WILP       | 0.532 ± 0.002 | 0.738 ± 0.036 |
| CAILP      | 0.478 ± 0.009 | **0.792 ± 0.010** |
| **LGBT**   |            |            |
| Firehose   | 0.647 ± 0.002 | 0.592 ± 0.010 |
| TopK       | 0.504 ± 0.010 | **0.630 ± 0.065** |
| CILP       | 0.651 ± 0.002 | 0.594 ± 0.011 |
| WILP       | **0.654 ± 0.001** | 0.596 ± 0.007 |
| CAILP      | 0.544 ± 0.015 | 0.592 ± 0.027 |

Table 4: Selected Features by the Methods. Keyword type prefixes= (@:mention, #:hashtag, loc:location and the rest are terms). All the keywords show up in the five splits except for the italicized ones for the WILP method which show up in 4/5 of the splits.

| TopK | CILP | WILP | CAILP |
|------|------|------|-------|
| **Natural Disaster** | | | | |
| storm | rt | rt | philippines |
| hurricane | the | #yolandaph | storm |
| earthquake | of | alaska | typhoon |
| philippines | to | islands | california |
| magnitude | in | the | magnitude |
| **Social Issues** | | | | |
| police | rt | rt | police |
| @deray | the | police | black |
| protesters | to | @lfc | protesters |
| @natedrug | in | @deray | justice |
| cops | is | protesters | @deray |
| **Space** | | | | |
| @nasa | rt | rt | @nasa |
| @tokiohotel | the | the | space |
| moon | to | @nasa | star |
| science | in | to | moon |
| @30secondstomars | for | sky | loc: houston tx |
| **Soccer** | | | | |
| @lfc | and | rt | cup |
| #ynwa | at | the | liverpool |
| #brazil | by | @lfc | goal |
| soccer | for | @fifaworldcup | match |
| @mcfc | from | to | football |
| **Human Disasters** | | | | |
| #isis | rt | rt | israel |
| @camilacabello97 | the | in | #iraq |
| israel | in | #iraq | gaza |
| gaza | to | #isis | war |
| underattack | of | gaza | #isis |

The WILP method manages to prevent some of the high coverage general keywords from CILP to show up in the top selected keywords; however, some of the general keywords do show up in the top keywords featured by the WILP methods which explains the similar filtering performance compared to the CILP method.

The presence of general terms such as "rt" in the solution can be used as a measure to approximate how much each method is improving the precision in the retrieved data. General terms such as "rt" and the are included in the CILP and WILP solutions for all of the topics, while CAILP manages to filter "rt" from the solutions across all topics. This shows that using the negative coverage in CAILP utilizes the information from the non-relevant tweets to
we ran the greedy solver across all topics for each method and when compared between methods. As an example, the greedy CAI LP we used the greedy solver discussed in part 3. To compare the methods (except for the gurobi CAILP).

method achieves a significantly higher F1-score than all the other plotted in Figure 4 alongside all the previously calculated results recorded the performance metrics of the results. The results are solution for the ILP methods doesn’t exactly match those of the the pare the results of the greedy solver with those of the Gurobi solver, it is also important to get ideal results for the queried data. To com-
mments, it took the Gurobi solver 82 minutes to terminate for an the problem grows in nearly exponential manner. From our exper-
iment, the Gurobi solver 82 minutes to terminate for an ILP with 300,000 tweets. On the other hand, the greedy solver consistently terminates with constant runtime ($O(k)$).

While it is important to get a solution within a reasonable time, it is also important to get ideal results for the queried data. To com-
pare the results of the greedy solver with those of the Gurobi solver, we ran the greedy solver across all topics for each method and recorded the performance metrics of the results. The results are plotted in Figure 4 alongside all the previously calculated results for comparison.

The plots show that while the queried data by the greedy so-
lution for the ILP formulations outputted by the Gurobi solver can lead to good results on the filtered data. The problem with using a solver such as Gurobi is that each additional tweet adds a new constraint to the LP and the problem space will grow as the input data grows in size. Defining this constraint space and solving the linear pro-
gramming problem using Gurobi under these constraints can quickly reach infeasible times and make the methods unscale-
able. In order to solve the problem for bigger datasets in faster time, we used the greedy solver discussed in part 3. To compare the solving time between the two solvers, we ran them on the same data (for topic "soccer") with increasing data sizes and recorded the solve time. The result is shown in Figure 3.

The Gurobi solver performs better for smaller datasets, but as the size of the problem data increases, the time for Gurobi to solve the problem grows in nearly exponential manner. From our experi-
ments, it took the Gurobi solver 82 minutes to terminate for an ILP with 300,000 tweets. On the other hand, the greedy solver consistently terminates with constant runtime ($O(k)$).

While it is important to get a solution within a reasonable time, it is also important to get ideal results for the queried data. To com-
pare the results of the greedy solver with those of the Gurobi solver, we ran the greedy solver across all topics for each method and recorded the performance metrics of the results. The results are plotted in Figure 4 alongside all the previously calculated results for comparison.

The plots show that while the queried data by the greedy so-
lution for the ILP formulations doesn’t exactly match those of the the Gurobi solution, but still it manages to get similar performance when compared between methods. As an example, the greedy CAILP method achieves a significantly higher F1-score than all the other methods (except for the gurobi CAILP).

To summarize all the results, out of all the proposed methods, the CAILP formulation gives us the best results. Increasing the recall of the TopK method while managing to maintain a high precision in the filtered data. Comparing the solvers, using the greedy solver gives us less precise results in faster time. Choosing which solver to use is a trade off between the precision of the results and the processing time.

6 RELATED WORK

There has been a considerable amount of work to train complex topical classifiers and event detection systems for Twitter. However, the main focus in most of these works is on the work done on a cached set of data previously acquired by querying the API under more simple rules and queries which results in the retrieval of a lot of non-relevant content instead of applying a more sophis-
ticated data collection method.

Phuvipadawat and Murata [21] and Hajjem and Latiri [5] use prede
dined search queries to retrieve tweets from the Twitter streaming and search APIs and apply online clustering to find similar tweets based on tf-idf to detect breaking news from Twitter data. Kim et al. [10] use tweets from verified popular news outlets to build a news classifier.

Iman et al. [8], Lin et al. [14], Yang et al. [27] and Magdy and El-
sayed [15] propose methods for following topics on Twitter; how-
ever, all of them query the API for raw and unfiltered data which wastes bandwidth and resources to retrieve large volumes of unre-
related content.

As for the work on searching microblog services such as Twitter, Hao et al. [6] introduce methods using query expansion to retrieve related tweets for a users interest based on a tweet selected by the user as a seed and generating new queries based on this tweet. Li et al. [13] have implemented a crawler in their work which uses query expansion methods to iteratively retrieve tweets relevant to an specific topic (crime and disaster event tweets). New queries are generated based on the results of previous queries until termina-
tion. While the results show improvements over querying the API for raw data, Multiple queries can result in waste of bandwidth - by retrieving redundant data multiple times - and may have problems with the API rate limits for certain topics.

Zheng et al. [28] use a semi-supervised method which maintains a
dynamic set of keywords along the development of an event. The importance of each keyword for an event is measured by a score determined by specific properties of each word. Wang et al. [26] start with a manually selected set of keywords and dynamically add hashtags showing up in the results to the keywords set. Becker et al. [1] generate dynamic keyphrases by using term fre-
quency analysis and event-related concept extraction from exter-
nal sources. While these methods work in a small setting involving a user, in a larger scope it is hard to generate an effective query to retrieve all the relevant tweets from one seed tweet since the tweet does not necessarily have overlaps with all the relevant tweets and there may be disjoint sets of tweets for topics.

Li et al. [12] use an interactive relevance feedback [23] which iteratively enhances a query based on the feedback from a human user. This approach requires constant supervision of a human user.

Figure 3: Solving time for greedy and Gurobi solvers as data size grows.
which isn’t completely compatible in our use case which works with dynamic topics over time.

Magdy and Elsayed [16] present an unsupervised adaptive method for filtering topics similar to our methods. However, the filtering method in this paper is heavily dependent on a classifier and may suffer from model misspecification in different situations. The novelty of our proposed method is in the fact that it can operate and filter independently of a classifier as long as we have a sample of the labeled dataset.

Despite the importance of the set covering and maximum coverage problems in various fields such as operations research, machine learning and data mining, there has not been much work done on utilizing these problems for learning query rules. The only work we are aware of that uses max coverage in a similar context is done by Saha and Getoor [22], which defines a maximum coverage problem to find blogs which have maximal relevancy to a list of topics. In addition to the difference in context and the size of data, their work tries to solve the problem in an online streaming environment while we are using previously labeled data to derive efficient queries.

7 CONCLUSION

In this paper, we introduced strategies based on maximum coverage of positive data to optimize the results of querying a limited search-based API. We proposed three general strategies utilizing different information about the data: a basic strategy using only the coverage of relevant data, a more sophisticated strategy maximizing the mutual information scores of the selected features and a third strategy using the negative coverage of tweets as a penalty in the maximization objective.

The proposed methods provided strong improvements in precision and recall over the baseline TopK approach and provide a bandwidth limited approximation of the gold standard Firehose results. Overall, the CAILP ILP formulation performed best by maximizing the relevant filtered data covered while minimizing the retrieval of non-relevant data.
We also implemented a simple greedy solver for getting approximated solutions to the formulated problem in a faster time. The greedy solver did perform in a noticeably faster time although the results were not as good as the MILP, as to be expected. However, the greedy solver used in this experiment can likely be improved in the future by introducing more sophisticated heuristics.

In concluding, we remark that there are a variety of potential applications of this work beyond Twitter topic classifiers. Overall, as the demand for free data faces the reality of limited APIs, the need for ways to retrieve content relevant to application needs will only grow. In cases where the relevant data is in a small minority, we need sophisticated filtering methods to retrieve most of the relevant data while filtering the non-relevant data. Our proposed method is not limited to the context of tweets and Twitter; it can be directly adapted to any API with a query interface to retrieve relevant data with high recall while filtering non-relevant content given a proper labeled dataset.

REFERENCES

[1] Hila Becker, Dan Iter, Mor Naaman, and Luis Gravano. 2012. Identifying content for planned events across social media sites. In Proceedings of the fifth ACM international conference on Web search and data mining. ACM, 533–542.

[2] Yan Chen, Hadi Amiri, Zhoujun Li, and Tat-Seng Chua. 2013. Emerging Topic Detection for Organizations from Microblogs. In Proceedings of the 9th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’13). ACM, New York, NY, USA, 43–52.

[3] Thomas M. Cover and Joy A. Thomas. 2006. Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience.

[4] Van Dang, Michael Bendersky, and W. Bruce Croft. 2013. Two-Stage Learning to Rank for Information Retrieval. In Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR’13) Springer-Verlag, Berlin, Heidelberg, 421–434. https://doi.org/10.1007/978-3-642-36973-5_36

[5] Malek Hajjami and Chiraz Latiri. 2017. Combining IR and LDA Topic Modeling for Filtering Microblogs. Procedia Comput. Sci. 112, C (Sept. 2017), 761–770.

[6] Xiaotian Hao, Ji Cheng, Jun Vosecky, and Wilfred Ng. 2017. Towards a Query-Less News Search Framework on Twitter. In Database Systems for Advanced Applications, Selçuk Candan, Lei Chen, Torben Bach Pedersen, Lijun Chang, and Wen Hua (Eds.). Springer International Publishing, Cham, 137–152.

[7] Dorit S. Hochbaum. 1997. Approximation Algorithms for NP-Hard Problems. PWS Publishing Co., Boston, MA, USA, Chapter Approximating Covering and Packing Problems: Set Cover, Vertex Cover, Independent Set, and Related Problems, 94–143.

[8] Zahra Iman, Scott Sanner, Mohamed Reda Bouajjane, and Lexing Xie. 2017. A Longitudinal Study of Topic Classification on Twitter. In Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM-17). Montreal, Canada.

[9] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. 1999. The budgeted maximum coverage problem. Inform. Process. Lett. 70, 1 (1999), 39–45.

[10] Yong Sung Kim, Eunjun Hwang, and Seungmin Rho. 2016. Twitter news-in-education platform for social, collaborative, and flipped learning. The Journal of Supercomputing (2016), 1–19.

[11] Eugene L. Lawler and David E. Wood. 1966. Branch-and-bound methods: A survey. Operations research 14, 4 (1966), 699–719.

[12] Cheng Li, Yue Wang, Paul Resnick, and Qiaozhu Mei. 2014. ReQ-ReC: High Recall Retrieval with Query Pooling and Interactive Classification. In Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’14). ACM, New York, NY, USA, 163–172.

[13] R. Li, K. H. Lei, R. Khadivala, and K. C. C. Chang. 2012. TEDAS: A Twitter-based Event Detection and Analysis System. In 2012 IEEE 26th International Conference on Data Engineering, 1279–1276.

[14] Jimmy Lin, Rion Snow, and William Morgan. 2011. Smoothing Techniques for Adaptive Online Language Models: Topic Tracking in Tweet Streams. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’11). ACM, New York, NY, USA, 422–429.

[15] Walid Megdy and Tamer Elsayed. 2014. Adaptive Method for Following Dynamic Topics on Twitter. In ICWSM.

[16] Walid Megdy and Tamer Elsayed. 2016. Unsupervised Adaptive Microblog Filtering for Broad Dynamic Topics. Inf. Process. Manage. 52, 4 (July 2016), 513–528.

[17] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie. 2013. Improving LDA Topic Models for Microblogs via Automatic Tweet Labeling and Pooling. In Proceedings of the 36th Annual ACM SIGIR Information Retrieval Conference (SIGIR-13). Dublin, Ireland.

[18] Bernhard Meindl and Matthias Tempel. 2012. Analysis of commercial and free and open source solvers for linear optimization problems. Eurostat and Statistics Netherlands within the project ESNest on common tools and harmonised methodology for SDC in the ESS 20 (2012).

[19] Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen M. Carley. 2013. Is the sample good enough? Comparing data from twitter’s streaming API with Twitter’s firehose. In Proceedings of the 7th International Conference on Weblogs and Social Media, ICWSM 2013. AAAI press, 400–408.

[20] Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing Sporting Events Using Twitter. In Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces (IUI ’12). ACM, New York, NY, USA, 189–198.

[21] Swit Phuvipadawat and Tsuyoshi Murata. 2010. Breaking News Detection and Tracking in Twitter. 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology 3 (2010), 120–123.

[22] Barna Saha and Lise Getoor. 2009. On maximum coverage in the streaming model & application to multi-topic blog-watch. In Proceedings of the 2009 siam international conference on data mining. SIAM, 697–708.

[23] Gerard Salton and Chris Buckley. 1997. Readings in Information Retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter Improving Retrieval Performance by Relevance Feedback, 355–364. http://dl.acm.org/citation.cfm?id=275537.275712

[24] Kate Starbird and Leyia Palen. 2012. (How) Will the Revolution Be Retweeted?: Information Diffusion and the 2011 Egyptian Uprising. In Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW ’12). ACM, New York, NY, USA, 7–16.

[25] Wanhu Su, Yan Yuan, and Ma Zhu. 2015. A Relationship Between the Average Precision and the Area Under the ROC Curve. In Proceedings of the 2015 International Conference on The Theory of Information Retrieval (ICTIR ’15). ACM, New York, NY, USA, 349–352.

[26] Xinyue Wang, Laurissa Tokarchuk, Félix Cuadrado, and Stefan Poslad. 2013. Exceptional Event Detection and Analysis System. In Proceedings of the 17th ACM SigKDD International Conference on Knowledge Discovery and Data Mining (KDD ’11). ACM, New York, NY, USA, 1907–1916.

[27] Xin Zheng, Aixin Sun, Sibo Wang, and Jialong Han. 2017. Semi-Supervised Event-related Tweet Identification with Dynamic Keyword Generation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM ’17). ACM, New York, NY, USA, 1619–1628.