Collecting Representative Social Media Samples from a Search Engine by Adaptive Query Generation

Virgile Landeiro
Department of Computer Science
Illinois Institute of Technology
Chicago, IL USA

Aron Culotta
Department of Computer Science
Illinois Institute of Technology
Chicago, IL USA

***Abstract*—Studies in computational social science often require collecting data about users via a search engine interface: a list of keywords is provided as a query to the interface and documents matching this query are returned. The validity of a study will hence critically depend on the representativeness of the data returned by the search engine. In this paper, we develop a multi-objective approach to build queries yielding documents that are both relevant to the study and representative of the larger population of documents. We then specify measures to evaluate the relevance and the representativeness of documents retrieved by a query system. Using these measures, we experiment on three real-world datasets and show that our method outperforms baselines commonly used to solve this data collection problem. The resulting methods may be used by computational social scientists to aid data collection efforts.**

***Index Terms*—classification, data collection, sampling bias**

I. Introduction

Scientific studies that use online data commonly require interaction with a search engine of some kind. For example, Twitter's API may be used to identify tweets matching certain keywords, or Google's API may be used to identify news stories on a given topic. The validity of the study will hence critically depend on the representativeness of the data returned by the search engine.

As a running example, consider a study to characterize the prevalence of hostile/abusive messages on Twitter. The researcher may wish to estimate the frequency of such messages, perhaps grouped by characteristics of the recipient (e.g., do women receive more hostile messages than men?). To collect data for such a study, one must have a mechanism to identify hostile tweets. A simple method is to first manually identify keywords that likely indicate hostility, and then query the Twitter API to find matching tweets. However, there are at least two key threats to the validity of this methodology[1] :

- **Coverage Error:** The keywords are unlikely to span all types of hostile messages, so we may miss a potentially significant portion of relevant data.
- **Sampling Bias:** If the keywords happen to over-represent messages sent to male recipients, then the study may erroneously conclude that men receive more hostile messages than women.

These problems are exacerbated when there is high class imbalance — when only a small fraction of documents are relevant to the study, uniform document sampling may have low sampling bias but high coverage error.

While similar issues have been raised in prior work [1], we still lack a complete understanding of when these issues are most problematic and how we can overcome them. To advance progress in this area, this paper offers the following contributions:

1) We provide several empirical measures to quantify the amount of coverage error and sampling bias in a text dataset.
2) Using these measures, we compare several common querying methods on three real-world datasets to identify the strengths and weaknesses of each approach.
3) We introduce a new querying algorithm that directly aims to reduce coverage error and sampling bias, finding that it outperforms traditional methods across all three datasets. The algorithm assumes we have access to a small set of documents annotated by relevance. We use these documents to generate search queries that are likely to return documents that are (a) relevant and (b) representative of the larger universe of queryable documents.

The remainder of this paper is organized as follows. In Section II, we present the work related to this paper across multiple research areas. In Section III, we define the general problem addressed in this paper and introduce two measures used to evaluate the results. Section IV describes the different

[1]These threats also exist even with more sophisticated sampling methodologies, discussed in more detail below.

ASONAM '19, August 27-30, 2019, Vancouver, Canada

ACM ISBN 978-1-4503-6868-1/19/08?/$15.00
https://doi.org/10.1145/3341161.3342924

query methods, and Sections V and VI present the datasets, experimental details, and empirical results.

II. Related Work

A. Computational social science

Exploiting online social media data to build observational studies is becoming increasingly common due to the diversity and quantity of such data. Olteanu et al. [2] applied a propensity-scored analysis on 3 months of Twitter data to study the self-reported situations of individuals. For instance, they identified and quantified the outcomes of these situations and analyzed the vocabulary used in these situations depending on the outcomes. Cunha et al. [3] used Reddit data and a matching method to show that social feedback from an online community has an impact on online engagement with the community as well as on the probability of achieving an offline goal (weight loss in this case). These examples all used some strategy to control for possible confounders (matching, propensity score) and thus are more robust to possible bias caused by these variables. However, concerns have been raised that the population on social media is not representative of the larger offline population leading to biased social media studies [4, 1]. Because these studies are used to infer real world outcomes, it is crucial that the bias induced by the data collection be minimal.

B. Active learning

Several active learning algorithms relate to our proposed method. For example, the problem of class imbalance pushed researchers in active learning to develop strategies to find documents belonging to rare classes [5, 6]. Another common problem in the active learning field is how to choose the best next instance to label. Recent strategies combined informative and representative samples in order to achieve the best post-labeling performance [7]. We can draw the parallel between the mixed strategy developed by Huang et al. [7] and the approach proposed in this paper. Here, we focus on how to pick the best next word to put in the current query by combining components of coverage/relevance and representativeness. Finally, another branch of active learning [8, 9] encourages taking advantage the annotator's knowledge to manually search documents for a specific class in a dataset, either before or during the more common task of labeling instances.

There is an orthogonal line of work in "deep web crawling" [10] that also aims to increase the coverage (but not representativeness) of a document collection; however, the goal is generally to increase the coverage of all documents, rather than those relevant to a specific topic.

Although the approach proposed in this paper takes inspiration from multiple techniques developed in active learning, it does not require human interaction and therefore has different aims from active learning. Instead, we wish to develop a more scalable, low-cost method that does not require additional human interaction to obtain high quality samples.

C. Convert a classifier into search query

Finally, researchers wishing to collect more relevant data studied how to convert a classifier trained on labeled data into search queries [11, 12, 13, 14]. The differences between these techniques and the proposed approach in this paper is that these techniques mostly use information gain to pick top terms and they do not account for representativeness. Thus, these strategies might not be appropriate to collect data for observational studies. In particular, they might under perform in the presence of class imbalance (supported by our experiments in Section VI).

III. Problem Definition

In this section, we first define the general problem we wish to solve and the evaluation measures of interest. We then introduce a framework to build a query and retrieve one or more documents relevant to the query.

A. General Problem

The problem setting we are interested in is as follows: Suppose a researcher wishes to collect documents possessing a certain attribute; for example, Instagram comments containing hostility or tweets expressing mental distress. The researcher has access to some third-party API to search the universe of documents by keyword. The problem is to determine which queries to submit and in what order so that the researcher may obtain a large set of relevant documents that are representative of the population of all relevant documents.

A key assumption is that the researcher does not have access to the entire universe of documents directly, which is most often the case for researchers working with online social media. This limitation presents the challenge of "unknown unknowns" [15] – the researcher does not know if the collected sample is representative of all relevant documents because there may be types of relevant documents that the researcher does not know about. Indeed, often the point of such a study is to discover the diversity of ways that a concept is expressed. There are also often additional constraints that limit the number of queries one can submit, rate limits, as well as the number of documents one can retrieve for each query. Rather than tie our approach to one specific API, our approach and experiments below investigate the problem from a generic search interface, investigating the overall effectiveness of various query generation methods.

More formally, let $\textsc{Search}(q)$ refer to the API search function that takes as input a query q and returns a document r that matches q. After each search, assume we store the query and the returned document in the sets Q and R, respectively. Thus, after submitting N queries, all retrieved documents will be stored in R. There are a few things to note about this procedure. First, two queries may return the same document, so $|R| \leq |Q|$. Second, while here we have restricted the output of $\textsc{Search}(q)$ to be a single document, we will extend this to multiple documents below.

B. Quality measures

The quality of a querying strategy depends on the quality of the retrieved documents R. Motivated by the common types of studies performed by this data, we consider two measures of quality below: **coverage** and **representativeness**. These measures are idealized in the sense that they require full knowledge of the document universe to be computed. While we of course will not have this knowledge in practice, in order to compare different strategies, we will conduct experiments with document collections where we do have such knowledge and thus can compute these metrics. Below, we will let $\mathcal{D} = \{d_1 \dots d_{|\mathcal{D}|}\}$ represent the full, unobservable population of documents.

1) Coverage: For most use cases, relevant documents are rare. Thus, it is critical to the power of the study that we are able to collect as many of the relevant documents as possible. The goal of the coverage measure is to evaluate how well a querying algorithm performs at retrieving documents from the rare class. Therefore, we define our coverage measure as the number of unique relevant documents retrieved divided by the total number of relevant instances in $\mathcal{D}$. This measure is also known as recall or sensitivity; we use the term coverage due to its usage in survey methodology [16].

Formally, let binary variable $y \in \{0, 1\}$ indicate the relevance label for a document. Thus, each retrieved document becomes a document/label tuple $(r_i, y_i) \in R$ and likewise for elements of the document population $(d_i, y_i) \in \mathcal{D}$. Coverage is then:

$$\text{covg}_R = \frac{\sum_{(r_i,y_i)\in R} y_i}{\sum_{(d_j,y_j)\in \mathcal{D}} y_j}$$

where $\text{covg}_R \in [0, 1]$ and higher values are better.

2) Representativeness: The second measure we develop focuses on how reflective the words used in the retrieved documents are of the population of words used in all relevant documents. As mentioned in the introduction, one common way to collect documents is to manually define keywords that are expected to occur in relevant documents. However, the data collection resulting from using these keywords will be biased by the researcher's knowledge of the rare events, and therefore it might not cover the space of all relevant documents. For example, querying common profane words will retrieve many positive examples of online hostility, but will ignore any hostile messages that do not contain profanity.

While one may be interested in representativeness with respect to certain known variables (e.g., demographics), it is difficult in the general case to know *a priori* which variables are important; moreover, identifying such variables in noisy web data is error-prone. Instead, we consider measures of representativeness that do not require additional domain knowledge, using word distributions to measure how similar the retrieved documents are to the document population.

We build on standard language modeling techniques to compute this measure. Let $\vec{x} = \{x_1 \dots x_k\} \in \mathbb{R}^k$ be a vector of word counts for a document, where there are k unique words in the vocabulary. From the document population $\mathcal{D}$, we can compute the multinomial distribution $P_{\mathcal{D}}(x_i \mid y = 1)$ for $i \in \{1 \dots k\}$. That is, for each word x_i in the vocabulary, we can compute the conditional probability of x_i appearing in a relevant document using the maximum likelihood estimate:

$$P_{\mathcal{D}}(x_i \mid y = 1) = \frac{n_{i1}}{\sum_j^k n_{j1}}$$

where n_{i1} is the number of times term i appears in a relevant document. We can estimate the analogous multinomial $P_R(x_i \mid y = 1)$ using only the word occurrences in the retrieved set of documents R.

We now have word multinomials from the document population $\mathcal{D}$ and the retrieved sample R. A natural way to measure the representativeness of the words in R with respect to D is to quantify the discrepancy between the two word distributions $P_{\mathcal{D}}(x_i \mid y = 1)$ and $P_R(x_i \mid y = 1)$. While there are many such measures (e.g., KL-divergence), we select the Hellinger distance [17], as it is symmetric, bounded, obeys triangle inequality, and has been used previously to compare multinomials. The Hellinger distance $H(P, Q)$ between two probability distributions $P = (p_1, ..., p_j)$ and $Q = (q_1, ..., q_k)$ is defined as:

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{1}$$

which is closely related to the Euclidean distance between two vectors.

We define our representativeness measure as the Hellinger similarity between two multinomial distribution as:

$$\text{repr}(P, Q) = 1 - H\left(P, Q\right) \tag{2}$$

In this case, we define the representativeness between the two multinomial distributions described above:

$$\text{repr}_R = \text{repr}\left(P_{\mathcal{D}}(x|y = 1), P_R(x|y = 1)\right) \tag{3}$$

Thus, if there is a word that is very probable in $\mathcal{D}$ but not R, the difference will be large in the Hellinger computation, and the representativeness value will thus be small. As the Hellinger distance is bounded between 0 and 1, so is repr_R. Higher values of repr_R are better.

IV. Methods

Now that we have defined the problem and described the properties of good solutions, we next turn to different querying strategies. Each method implements a function CreateQuery$(\mathcal{U}, \mathcal{L}, R, Q)$, where

- $\mathcal{U} \subset \mathcal{D}$ is a small, unlabeled dataset sampled from the document population $\mathcal{D}$
- $\mathcal{L} \subset \mathcal{D}$ is a small, labeled dataset containing tuples $(\vec{x}, y)$. For example, a researcher may manually query a web interface to find a number of possibly relevant documents, then upon inspection determine their true relevance label.
- Q is the set of previously issued queries.

- R is the set of previously retrieved document.

We begin by describing two baseline approaches, then proceed to our proposed approach.

A. Baselines

To evaluate the performance of our approach, we developed two natural baselines optimized for one of representativeness or coverage.

1) Baseline 1 - Random sampling: The first baseline only uses information from the unlabeled data $\mathcal{U}$. The goal is to retrieve documents at random with respect to the frequency of each term in the vocabulary. To do so, we estimate a multinomial $P_{\mathcal{U}}(x)$, which is simply the probability of each word x_i appearing in dataset $\mathcal{U}$.

Each call to CREATEQUERY samples keywords from $x_i \sim P_{\mathcal{U}}(x)$. Thus, terms that are more common overall in $\mathcal{U}$ will be queried more often than terms that are rare.

This baseline method uses no information at all about relevance, but instead simply tries to generate a sample R that is representative of the word distribution over all documents. Thus, we expect this baseline to have very poor coverage, particularly when relevant documents are rare. In the experiments below, we refer to this baseline as B_1.

2) Baseline 2 - Most predictive words: A second strategy, designed to maximize coverage, is to somehow create a list of keywords that are likely to indicate a relevant document [11, 12, 13]. This baseline only uses information from the labeled data $\mathcal{L}$, using standard text classification methods to identify words that are highly correlated with the relevant class. Specifically, we fit a logistic regression classifier $p(y \mid \vec{x})$ on $\mathcal{L}$ to predict document relevance given the word counts it contains. We then examine the model coefficients and select words that have the highest coefficients for the positive class. In the experiments below, we consider two variants: $B_2(1\%)$ selects the top 1% of words, while $B_2(10\%)$ selects the top 10%.

The result of this procedure is a list of words, weighted by their corresponding model coefficient. Each call to CREATEQUERY samples keywords proportionally to their weights. Thus, terms that are more correlated with relevant documents in $\mathcal{L}$ will be queried more often than terms that are less correlated.

While this approach is expected to greatly increase coverage, by myopically focusing on a small list of relevant terms, we expect this approach to have low representativeness.

B. Proposed Approach: Multi-objective query construction

The baselines described above present two extreme strategies — the first favors representativeness, while the second favors coverage. This is analogous to the "exploitation versus exploration" problem that arises in many artificial intelligence settings [18]. On the one hand, we want to maximize the chance of finding a relevant document; on the other hand, we want to explore the document space to discover the "unknown unknowns". Our proposed approach aims to improve over the baselines by combining ideas from each baseline into a single objective. Additionally, the above baselines are all *static*; the ith query generated does not depend on any of the previous queries or the documents returned. Our proposed approach instead updates its model after each query to refine its strategy.

At the core of our approach is a function $f(x)$ that scores the importance of including term x in the query under construction. To construct a one word query, we select the word with maximum value of $f(x)$. To construct a multi-word query $q = \{x_1 \dots x_n\}$, we add one word at a time to the query, at each iteration updating components of $f(x)$ to re-rank the remaining words in the vocabulary.

The function $f(x)$ is composed of three functional components: two for representativeness and one for coverage. We then combine these criteria using the geometric mean to get a unique score per word. The three components of the scoring function are:

1) f_c, which improves coverage by favoring words that correlate with relevant documents
2) $f_{r(\vec{x})}$, which favors words that improve marginal word representativeness
3) $f_{r(\vec{x}|y)}$, which favors words that improve the class conditional representativeness

Below, we will describe each of these components in turn, then explain how we combine them into a single objective.

1) Targeting high coverage: f_c: The first component f_c focuses on selecting words that will increase covg_R. To do so, we let $y = 1$ be the relevant class, q the query created, and (r, y_r) the document retrieved and its class. Thus, coverage will increase if $y_r = 1$. However, because the document r returned by SEARCH is not known at query creation time, we use q as a surrogate for r. Therefore, we define $f_{c,i}$ for every word x_i as:

$$f_{c,i} = p_{\mathcal{L}}(y = 1|x_i) \tag{4}$$

where $p_{\mathcal{L}}(y = 1|x_i)$ is a classifier trained on $\mathcal{L}$. This component can be seen as a version of the baseline using the words most predictive of the relevant class considering all the words in the vocabulary. Therefore, this component will put more importance on words that are highly predictive of the relevant class.

2) Targeting high marginal representativeness: $f_{r(\vec{x})}$: The second component $f_{r(\vec{x})}$ puts more weight on words that will raise the marginal representativeness. Using the representativeness definition from Equation 2, we define the marginal representativeness as:

$$\text{repr}_{\vec{x}}(Q) = \text{repr}\left(p_{\mathcal{L}\cup\mathcal{U}}(\vec{x}), p_{R\cup Q}(\vec{x})\right)$$

where $p_{\mathcal{L}\cup\mathcal{U}}(x_i)$ is the unigram probability for word x_i in either the labeled or unlabeled datasets, and $p_{R\cup Q}(x_i)$ is the unigram probability of word x_i to appear in one of the queries created by CREATEQUERY or one of the documents retrieved by SEARCH. We use word counts with add-one smoothing to compute these probabilities. Then, given the current state of sets Q and R, we can estimate by how much $\text{repr}_{\vec{x}}$ would

increase if we were to add query q_i containing only the word x_i to Q:

$$\delta_i \text{repr}_{\vec{x}} = repr_{\vec{x}}(Q \cup q_i) - repr_{\vec{x}}(Q)$$

$\delta_i \text{repr}_{\vec{x}}$ will be the largest for words that are least accurately represented by $R \cup Q$ compared to $\mathcal{L} \cup \mathcal{U}$. Therefore, for every word x_i in the vocabulary, we define the second component:

$$f_{r(\vec{x}),i} = \delta_i \text{repr}_{\vec{x}} \tag{5}$$

If the query creation process was defined as always choosing the word that maximizes this component, then it would end up being a greedy approach to achieve high marginal representativeness.

3) Targeting high class conditional representativeness: $f_{r(\vec{x}|y)}$: In this section, we propose a third factor $f_{r(\vec{x}|y),j}$ that focuses on high class conditional representativeness. First, let us define Q_y and R_y, the sets of queries and documents that are associated with the target class $y = 1$. Similar to the previous component, we would like to use the word counts over Q_y and R_y to compute $p(x_i|y = 1)$ for every word x_i. However, since we do not know the true class labels of the retrieved documents in R, we use soft counts based on the classifier $p_{\mathcal{L}}(y = 1|\vec{x})$. For a retrieved document r, the word count over R_y is updated proportionally to $p(y = 1|r)$ for each possible class. In other words, the word count for the retrieved documents for class $y = 1$ is weighted by the confidence of our predictive model for class $y = 1$. Then, using the same method as in the previous section, we compute the increase in class conditional representativeness caused by word x_i as:

$$\delta_i \text{repr}_{\vec{x}|y} = repr_{\vec{x}|y}(Q_y \cup q_i) - repr_{\vec{x}|y}(Q_y)$$
$$\text{with } \text{repr}_{\vec{x}|y}(Q_y) = repr(p_{\mathcal{L}}(\vec{x} \mid y = 1), p_{R_y \cup Q_y}(\vec{x} \mid y = 1))$$

where q_i is the query that only contains the word x_i. Finally, we define the third component $f_{r(\vec{x}|y),i}$ for every word x_i as:

$$f_{r(\vec{x}|y),i} = \delta_i \text{repr}_{\vec{x}|y} \tag{6}$$

4) Combining the three components to select the best next keyword: In the previous sections, we described three factors that can be computed for each word x_i in the vocabulary and that we wish to maximize when creating a query. In this section, we take advantage of the geometric mean attributes to simply combine these three factors in one final importance weight w_i for each word:

$$w_i = \sqrt[3]{f_{c,i} \times f_{r(\vec{x}),i} \times f_{r(\vec{x}|y),i}} \tag{7}$$

Because the geometric mean is independent to the scales that each factor is expressed in as long as they stay the same for each word, this allows for limited preprocessing of each factor. The only necessary step is to make sure that all values are greater than 0. Additionally, the geometric mean tends to penalize more low values compared to the arithmetic mean, leading to weights that will be larger when all factors are relatively large than when two factors are very large and one factor is very small. This characteristic of the geometric mean reduces the chance of w_i being dominated by one factor and allows for a fairer combination of the three factors.

Finally, once w_i is computed for every word in the vocabulary, we use a greedy strategy by adding word x_i with the maximum associated weight w_i to the current query. We can then update the two factors accounting for representativeness before restarting the process to select the next word.

V. Data

In this section, we describe the three datasets collected to run the experiments of this paper. We report the size of each dataset and the F1 score of a standard logistic regression model for each dataset using 10-fold cross-validation on $\mathcal{L}$ in Table I. We encode each document as a vector of word occurrences using unigram features only.

a) Online harassment [19]: This dataset contains 1,134 Instagram posts and 30,987 comments that were manually labeled by Amazon Mechanical Turk workers as being hostile comments (positive class) or not (negative class). Around 4,000 comments are labeled as positive; for the purpose of our experiments, we consider each comment as an individual document, ignoring the thread structure.

b) Twitter smoking cessation: This is a newly collected dataset on smoking cessation, containing 12K smoking-related tweets. A tweet was manually labeled as positive if the author displays some intent to quit smoking; otherwise it is labeled as negative. As shown in Table I, around 2,000 tweets are labeled as positive, making this dataset the one with the highest proportion of positive documents.

c) 20 newsgroups: The 20 newsgroups dataset[2] is commonly used to evaluate natural language processing methods. It consists of approximately 20,000 documents covering 20 different topics from computer graphics to atheism. Each document in this dataset is annotated with the topic it belongs to. For the purpose of our experiments, we binarize the topics and create a new annotation such that each document is annotated with a positive label if it belongs to the `misc.forsale` topic and with a negative label otherwise. The `misc.forsale` topic contains online advertisements from people looking to sell personal items (e.g. games, computers, furniture, etc). For each ad, we remove the headers, footers, and the quotes.

To create the three disjoints datasets required by our query method, we shuffle each dataset and then split it in shares of 10, 30, and 60% of the original dataset. These subsets are then assigned to $\mathcal{L}$, $\mathcal{U}$, $\mathcal{D}$, respectively. Table I shows details of the dataset, as well as the cross-validation accuracy of a logistic regression classifier on $\mathcal{L}$, to provide an estimate of the difficulty of each classification task.

VI. Experiments and Results

In Section III-B, we proposed to evaluate the quality of the documents R retrieved by a query method using a coverage measure covg_R and a representativeness measure repr_R. In this section, we use these two measures to compare the

[2]http://qwone.com/ jason/20Newsgroups/

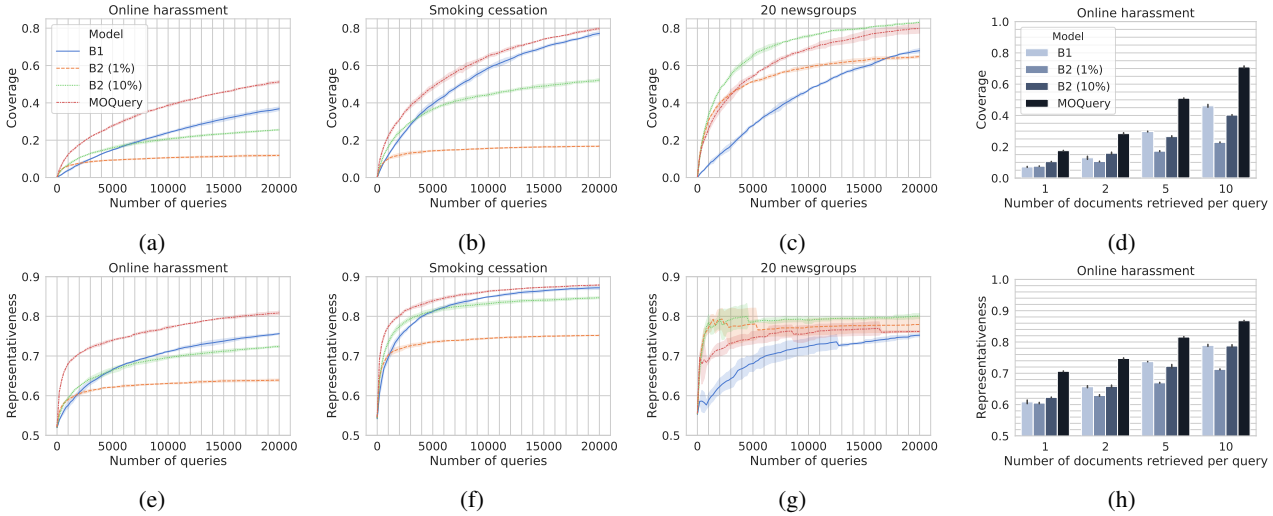

Fig. 1: Comparison of the coverage (a-c) and representativeness (e-g) of the four query generation methods; panels (d) and (h) additionally show how performance improves with the number of documents retrieved per query.

	Size	%rare class	10-fold $F1_{\mathcal{L}}$
Online harassment	30,987	13.2	0.677±0.048
Smoking cessation	12,000	18.3	0.717±0.062
20 newsgroups	20,000	5.2	0.692±0.131

TABLE I: Dataset statistics.

performance of our proposed approach in Section IV-B with two baselines defined in Section IV-A. For the retrieval model, we use the Okapi BM25 scoring function [20], a common baseline information retrieval system. Additionally, the queries we create are limited to a maximum length of 10 words, without replacement. To generate each query, we first uniformly sample a number from 1 to 10, then generate a query of the corresponding length.

A. Comparison to baselines

In our first experiment, we consider four CREATEQUERY methods: (1) B1 is the baseline simulating random sampling described in Section IV-A1, (2) B2 (1%) is the baseline using the top 1% of the words highly correlated with the relevant class to create a query, (3) B2 (10%) is the same method but using the top 10%, (4) MOQuery is the multi-objective query method we propose. We then run COLLECTDATA for each query method and over each dataset with $N = 20,000$ queries and we retrieve one document per query. The first three columns of Figure 1 show the evolution of the coverage measure on the first row and the representativeness measure on the second row for each dataset.

On the first two datasets, the proposed approach successfully outperforms the best baseline after any number of queries. After 20,000 queries, MOQuery provides 10% more coverage and around 5% more representativeness than B1 on the online harassment dataset (Figs. 1a and 1e). Similarly (Figs. 1b and 1f), it outperforms B1 by a few points on the smoking cessation dataset. However, coverage and representativeness on the 20 newsgroup dataset are both dominated by B2. As a reminder, the relevant class in this dataset contains advertisements for items on sale whereas the negative class contains documents on all kinds of topics (e.g. science, religion). Figure 1c indicates that a large portion of the positive instances in the 20 newsgroups dataset can be described using only the 10% most predictive words in the vocabulary. However, when this is not true (see the other datasets), then this method dramatically under performs compared to B1 and MOQuery. This is expected as B2 does not account at all for representativeness and therefore is limited when exploring the keyword space. Thus, when the class concept is very simple, the B2 baseline performs well; when the class concept is more complex, MOQuery can greatly improve both coverage and representativeness.

Another interesting behavior appears when evaluating representativeness for the 20 newsgroups dataset (Figure 1g): all the methods except for B2 (10%) display a sudden drop in representativeness at some point after a document is retrieved. After analysis, this appears to be caused by the same document across all methods albeit at different times in the query process. This document contains a large number of very rare terms – error messages in this case – leading to a drop in the representativeness of R.

Finally, we note that B1 achieves relatively higher representativeness and coverage on the smoking cessation dataset than on the other datasets. This is due to the small size of this dataset (12K instances) and the relatively large ratio of positive instances (approx. 18%). After 20,000 queries, B1 is able to retrieve more than 85% of the 2,153 positive instances. This behavior indicates that in case of a less imbalanced dataset, B1 might be sufficient to collect enough instances of the rare class. However, when the dataset is larger and the ratio of rare

instances is smaller, then B1 performs poorly compared to our approach MOQuery.

Overall, the proposed approach tops the baselines in relevance and coverage measures for all but one dataset in which its performance is only slightly lower than B2.

B. Effect of returning multiple documents per query

From the results of the previous section, we note that retrieving one document at a time is expensive. For example, in the case of the online harassment dataset, the best approach only retrieved approximately 50% of the positive instances accounting for around 2,000 instances after 20K queries. In this section, we reduce the number of queries from 20,000 to 2,000 but we increase the number of documents retrieved by SEARCH. We then compare the same query methods and investigate whether coverage and representativeness are impacted by this. For brevity, we include results for the online harassment dataset only. Similar results for the two remaining datasets can be found in the extended version of this paper.

Figures 1d and 1h display the coverage and the representativeness measures after 2,000 queries given that we retrieve either 1, 2, 5, or 10 documents per query. First of all, the results show that increasing the number of retrieved documents does not affect the rank of the methods. Second, although the performance increase is sublinear, retrieving more documents per query greatly improves the coverage score. For instance, MOQuery jumps from 17.6% of all positive documents retrieved to 71.3% when raising the number of documents per query from 1 to 10 in the online harassment dataset. The effect on representativeness is smaller but still consistent.

These experiments indicate that in real life situation, it is definitely an advantage to retrieve multiple documents for a unique query in order to speed up the collection of relevant documents. The results also suggest that our approach is able to adapt to a system where several documents are retrieved by correctly updating the factors associated with each word.

C. Additional Results

1) Impact of each factor: In this section, we focus on understanding what is the contribution of each factor in the proposed approach. To do so, we create three new query methods based on our approach with the slight difference that each method only contains two of the three components $\{f_c, f_{r(\vec{x})}, f_{r(\vec{x}|y)}\}$. We run 2,000 queries and retrieve 10 documents per query. Then, we report the difference of coverage between MOQuery (column 2) and each one of these variants (columns 3 to 5) in Table II. The first column shows the coverage score of the full proposed approach for every dataset. The other columns show by how much the coverage changes when a factor is removed. We do not display the representativeness results as the variations between each model is minimal.

Interestingly, f_c – targeting high coverage – is not always the component increasing coverage the most. Moreover, when considering the results averaged across all datasets, $f_{r(\vec{x})}$ seems to be the most important factor for coverage even though its goal is to maximize representativeness. This supports the assumption that more exploration of the keyword space is vital to retrieve a larger variety of documents.

2) Effect of updating the target distributions: In the previous section, we studied the impact that removing one component has on coverage in the proposed approach. Here, we propose a simple extension of the proposed approach to tackle the problems of covariate shift (word distribution is changing from $\mathcal{L}$ to $\mathcal{D}$) and word discovery (words correlated with the relevant class are in $\mathcal{D}$ but not in $\mathcal{L}$). To do so, we use documents in R and their pseudo labels to augment $\mathcal{L}$ and $\mathcal{U}$ while maintaining $p_{\mathcal{L}}(y)$ constant. This causes the distributions of words over $\mathcal{L}$ and $\mathcal{U}$ to change; thus making the representativeness components $f_{r(\vec{x})}$ and $f_{r(\vec{x}|y)}$ change for each word. The coverage component does not change as we do not retrain the predictive model with the augmented data. This approach is inspired by *self-training* [21], a semi-supervised learning approach. We implement this technique and show the coverage increase in the last two columns of Table II. The first column shows the improvement when we know the true label of every document in R using an oracle. Finally, the last column displays the coverage improvement when we augment the datasets with $R_t \subseteq R$ such that R_t only contains documents that our classifier predicted with confidence of 50% or more. We can see that even without using any active learning (i.e., no additional annotated examples are provided), we can improve the quality of the retrieved results using a classifier trained on the small training data $\mathcal{L}$.

VII. CONCLUSION

In this paper, we investigated the overall question: how should we query a search engine in order to gather a large set R of relevant documents given user-specified attributes? We then proposed a measure of **coverage** and a measure of **representativeness** to evaluate the quality of the set of documents R retrieved by a query method. We introduced two baseline methods: one targeting coverage and the other targeting representativeness. Then we proposed a multi-objective approach combining three components to build a query that improves both coverage and representativeness. Finally, we compared this approach to the baselines on three real-world datasets displaying class imbalance, and we showed that the proposed approach outperforms the baselines on both coverage and representativeness.

REFERENCES

[1] A. Olteanu, C. Castillo, F. Diaz, and E. Kiciman, "Social data: Biases, methodological pitfalls, and ethical boundaries," SSRN Pre-print, 2016.

[2] A. Olteanu, O. Varol, and E. Kıcıman, "Distilling the outcomes of personal experiences: A propensity-scored analysis of social media," in *Proc. of The 20th ACM Conference on Computer-Supported Cooperative Work and Social Computing*, 2017.

[3] T. Cunha, I. Weber, and G. Pappa, "A warm welcome matters!: The link between social feedback and weight

	MOQuery	w/o f_c	w/o $f_{r(\vec{x})}$	w/o $f_{r(\vec{x}\|y)}$	Oracle	Predicted
Online harassment	70.0%	-5.0%	-2.4%	-5.2%	+2.4%	+1.4%
Smoking cessation	85.8%	-11.2%	-5.0%	-5.5%	+5.9%	+5.3%
20 newsgroups	82.0%	-3.5%	-6.3%	-2.1%	+5.2%	+4.2%
Average	79.3%	-6.6%	-4.6%	-4.3%	+4.5%	+3.6%

TABLE II: Coverage changes caused by removing a component from MOQuery (columns 3 to 5) and by updating the target distributions (columns 6 and 7).

loss in/r/loseit," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 1063–1072.

[4] F. Diaz, M. Gamon, J. M. Hofman, E. Kıcıman, and D. Rothschild, "Online and social media data as an imperfect continuous panel survey," *PloS one*, vol. 11, no. 1, p. e0145406, 2016.

[5] T. M. Hospedales, S. Gong, and T. Xiang, "Finding rare classes: Active learning with generative and discriminative models," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 2, pp. 374–386, 2013.

[6] S. Li, S. Ju, G. Zhou, and X. Li, "Active learning for imbalanced sentiment classification," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 139–148.

[7] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," in *Advances in neural information processing systems*, 2010, pp. 892–900.

[8] J. Attenberg and F. Provost, "Why label when you can search?: alternatives to active learning for applying human resources to build classification models under extreme class imbalance," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 423–432.

[9] A. Beygelzimer, D. J. Hsu, J. Langford, and C. Zhang, "Search improves label for active learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3342–3350.

[10] I. Hernández, C. R. Rivero, and D. Ruiz, "Deep web crawling: a survey," *World Wide Web*, pp. 1–34, 2018.

[11] S. Zelikovitz and M. Kogan, "Using web searches on important words to create background sets for lsi classification." in *FLAIRS Conference*, vol. 1, 2006, pp. 298–603.

[12] A. Anagnostopoulos, A. Z. Broder, and K. Punera, "Effective and efficient classification on a search-engine model," in *Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, 2006, pp. 208–217.

[13] Z. Xu, R. Jin, K. Huang, M. R. Lyu, and I. King, "Semi-supervised text categorization by active search," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 1517–1518.

[14] R. Guzmán, M. Montes, P. Rosso, and L. Villasenor, "Improving text classification by web corpora," in *Advances in Intelligent Web Mastering*. Springer, 2007, pp. 154–159.

[15] H. Lakkaraju, E. Kamar, R. Caruana, and E. Horvitz, "Identifying unknown unknowns in the open world: Representations and policies for guided exploration." in *AAAI*, vol. 1, 2017, p. 2.

[16] P. P. Biemer and L. E. Lyberg, *Introduction to survey quality*. John Wiley & Sons, 2003, vol. 335.

[17] M. Nikulin, "Hellinger distance. hazewinkel, michiel, encyclopedia of mathematics," *Springer, Berlin. doi*, vol. 10, pp. 1 361 684–1 361 686, 2001.

[18] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[19] P. Liu, J. Guberman, L. Hemphill, and A. Culotta, "Forecasting the presence and intensity of hostility on instagram using linguistic and social features," *Ann Arbor*, vol. 1001, p. 48109, 2018.

[20] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford *et al.*, "Okapi at trec-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[21] M. Chen, K. Q. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *Advances in neural information processing systems*, 2011, pp. 2456–2464.

APPENDIX

A. Additional Results

1) Insights into query building: In this section, we display some of the queries generated by our approach when one document is returned per query. Table III shows the i-th query generated for the online harassment dataset with i spanning from 1 to 10,000.

We see that for the 100 first queries, most of the words are strongly associated with the positive class (hostile comment). This is normal as, at this point, the most predictive words for the positive class are also the ones that are increasing representativeness the most. Additionally, we notice that the term "specialmentioned" – encoding for the mention of another user in a comment and most frequent term in this dataset – appears in a large number of queries. In the early queries, it has a very large weight towards maximizing the marginal representativeness. In other words, our sample of retrieved documents only have a few occurrences of "specialmentioned" and it is not representative of how frequent this term appears in the overall dataset. Therefore, the query system puts it in the query in the hope to find a document that contains it and to close the gap between the frequency of this document in R and its "true" frequency in U.

After approximately 500 queries, the vocabulary used is more diverse although terms associated with hostile comments still appear in the queries. At this point, the small number of words providing the largest boost through f_c and $f_{r(\vec{x}|y)}$ in the early queries have been sampled in a reasonable amount such that the rest of the vocabulary is now more important in order to improve representativeness.

B. Further Discussion

In this section, we briefly discuss two decisions we made when building our approach.

a) Choice of the scoring function: For our approach, the coverage and representativeness performances of a querying method are highly dependent on the scoring function. Here, we decided to use Okapi BM25 as it is a scoring function commonly used (e.g. standard in Apache Lucene) but it is possible that our approach would require slight modifications when transferring from our experimental settings using Okapi BM25 to an application using other scoring functions.

b) Choice of the divergence measure: We also decided to use the Hellinger distance to measure the difference between two probability distributions in Section IV-B. However, many other methods exist to measure the distance between two probability distributions (KL-divergence, Jensen-Shannon divergence). It would be interesting to study if changing the distance measure used to weight a word for representativeness would have a significant impact on the coverage performance of our approach.

C. Limitations and Details

1) Limitations: In this section, we explain the limitations of the proposed approached and how we could extend it in the future.

The limitations of our method are mostly tied to the quality of the datasets used to initialize the approach. To use our query method, we need access to a small labeled dataset $\mathcal{L}$ and a larger unlabeled dataset $\mathcal{U}$. Because these datasets guide query generation, the more representative these data are of the unobservable data $\mathcal{Q}$, the better we expect the method to perform. Empirically, we find that even very small sample sizes for $\mathcal{L}$ and $\mathcal{U}$ are sufficient to generate effective queries.

Additionally, our queries are currently limited to words present in $\mathcal{L}$ or $\mathcal{U}$, since these are the words for which we can compute scores for the objective. In future work, we will consider semi-supervised approaches to term discovery.

Finally, our method is also sensitive to the annotation quality in $\mathcal{L}$, as the coverage factor f_c is directly based on them. Noisy annotations will create false associations between words and the relevant class and will reduce the association weight of words truly correlated with the relevant class. This will in turn reduce the power of the coverage component f_c.

a) Details about the proposed approach: When developing the second and third factors in the proposed approach targeting high representativeness, we compute a probability $p_{Q \cup R}$ for each word by counting occurrences of this word over the union of all the queries Q and all the documents retrieved R. However the representativeness measure repr_R used for evaluating query methods only uses R. Therefore, one could wonder why we use $Q \cup R$ in our model. We do so in order to avoid the situations in which a word x_i is added to a query because it improves the representativeness but it does not appear in the document returned by SEARCH. If we were to compute the probability of a word to occur using only R in the components, then this term x_i would repeatedly be picked in the next queries until a document is matched that contains it. In the rare case where no document in $\mathcal{D}$ contains this term, then it would be added to every remaining queries, which is not optimal for exploring the space of words. By considering the set of all queries when computing the probability of x_i to appear, we allow for a quicker exploration of the word space. The intuition behind this modeling choice is that we wish not to dwell on words that are added to queries but not retrieved in documents.

Number	Content
1	specialmentioned bitch ass fuck shit emoji_face_with_tears_of_joy nigga
25	specialmentioned dick emoji_face_with_tears_of_joy ass bitch
50	specialmentioned don lmao real dumb
100	don fag nigga probably specialmentioned hoe ass
250	emoji_face_with_tears_of_joy man shit ll post bunch
500	like believe got bro page
1,000	uses got fuck hood specialmentioned don
2,500	brought fucked turn specialmentioned rocks cup
5,000	#marvelcomics towers slave threats leaf closely #grandtheftauto
10,000	wether including bgc bestfriend make

TABLE III: Example of queries generated for the online harassment datasets at different times.

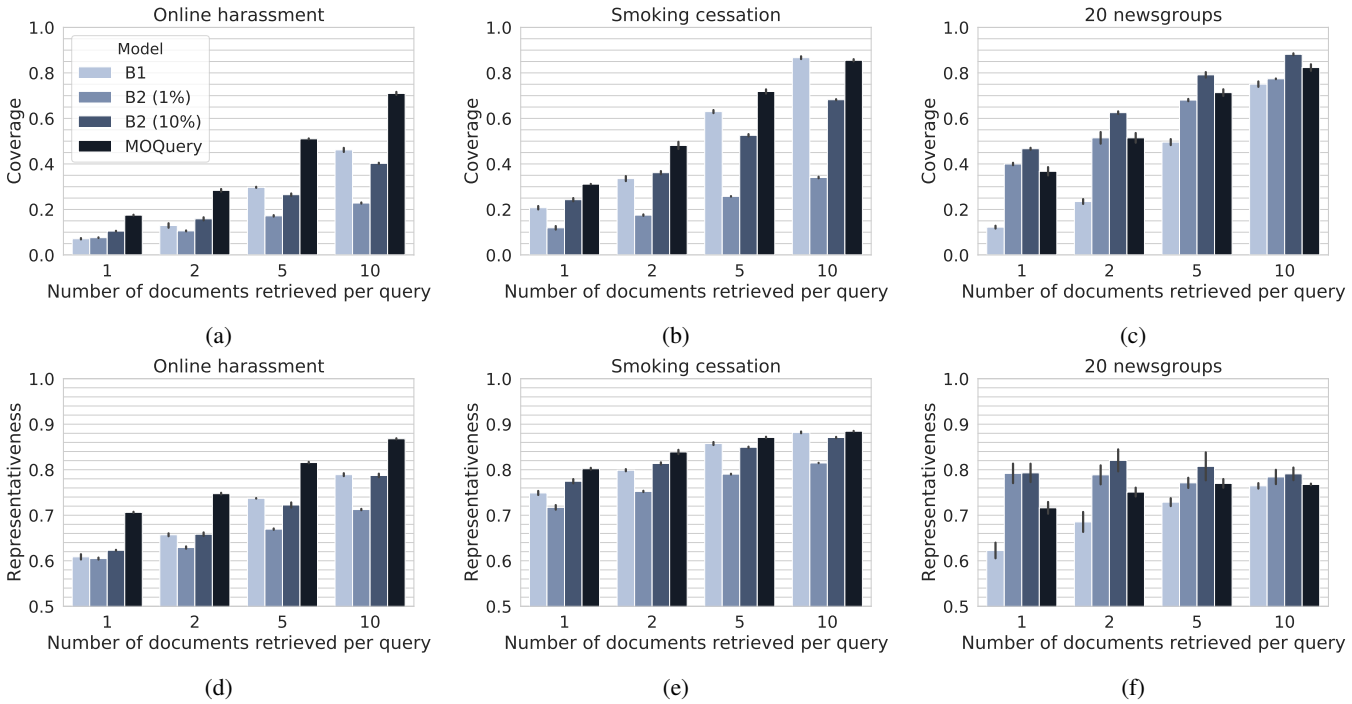

Fig. 2: Additional results: effect of returning multiple documents per query