@twitterdata

The Effective Filtering of Twitter Data

By Jeff Kolb

About the author

Jeff Kolb

Technical Lead | @kolb

Jeff is the Technical Lead for the Gnip Data Science team at Twitter. He guides customers through the process of deriving genuinely useful conclusions from the large-scale analysis of Twitter data. In a previous career, he studied the origin of mass at CERN.

1 Introduction

Twitter is an enormous repository of human thought and speech. Users representing vast demographic swathes continually grow a rich network of connections, relationships, and affinities. Because of the size and breadth of this source of human data, an analysis of Twitter content and connections can provide reliable answers to a wide variety of business questions. However, we must start by asking good questions of the data. What are we doing with this data? Why are we analyzing it? What question are we trying to answer? What expectations do we have of an answer? What is “good enough” and what qualities must the result absolutely have?

Given a suitable question to be answered, the task is to acquire and analyze the data in a way that is respectful of user privacy and, furthermore, meets the requirements of the business question. The latter aspect of the task consists of mapping the problem to be solved onto a data selection technique, as well as choosing, implementing, and refining an analysis technique. The focus of this paper is the Tweet selection technique.

Data customers interact with Twitter’s Gnip APIs via the PowerTrack query language. This interface allows a user to specify which Tweets are delivered, based on keyword matching and a variety of other filters on Tweet metadata. A set of PowerTrack rules represents an attempt to collect all Tweets relevant to some concept. What choices can we make to affect the number of desired and undesired Tweets received and not received? We propose a series of choices and evaluations to iteratively arrive at a state of data quality that meets the needs of the question we’re asking.

1.1 Tweet selection basics

Tweet selection is determined by two components: PowerTrack rules supplied to Gnip APIs –which determine which Tweets are delivered– and downstream filters, which determine how the Tweets are classified and accepted or rejected for further analysis.

PowerTrack rules operate by simple pattern matching, but it's easy to assume we are specifying and achieving semantic or meaning matches. One of the goals of this paper is to provide techniques by which you can narrow the gap between semantic matching and pattern matching, and to explain the remaining difference. In the case of downstream filters, we have the opportunity to use more robust classification techniques such that we can get closer to genuine semantic filtering.

A common analysis technique is to perform natural language processing on a language-filtered stream of Tweets. In this case, the rate of language misidentification would have a direct effect of downstream error rates. In another example, imagine a downstream application that must not ingest any Tweets with explicit adult content. In this case, the accuracy of the classification must be very high, even at the expense of throwing away a significant number of innocent Tweets. In contrast, an application that reconstructs conversations with a customer- service Twitter account can afford to misidentify some Tweets as relevant, because a human agent is applying human reason to the delivered conversations. However, misconstructed or missing conversations have a potentially high cost to the brand's perception. These data quality evaluations will be discussed in more detail in Section 3.

Although the needs and constraints from the business problem to be solved determine the necessary values for the data quality metrics, it typically takes a series of adjustments to the PowerTrack rules and downstream filters to meet these requirements.

In the following sections, we will outline this iterative process and give many examples. In Section 2, we review the PowerTrack query language and discuss common misunderstandings and suggest best practices. In Section 3, we present some broad categories of filtering, segmentation, and selection techniques. We also discuss techniques for Tweet corpus evaluation and description. In Section 4, we describe the cycles of iteration over the core analysis workflow, and we present a set of detailed examples. In Section 5, we review the effects of filtering on downstream analysis applications, and suggest some key maintenance tasks.

2 PowerTrack filtering

PowerTrack rules determine which Tweets a customer receives. While this interface is ultimately only one of many “knobs” that the user can “turn” to adjust the quality of the data set, it is particularly important because (1) it determines which Tweets will never be seen for downstream filtering and analysis, and (2) its Boolean filters and robust API provide a particularly simple interface to the data source.

PowerTrack rules can be thought of as binary classifiers: they label every Tweet as “in” or “out”, and the system delivers only the Tweets with the “in” label (or, equivalently, filters out only the Tweets with the “out” label). The goal of a PowerTrack rule, then, is to map characteristics of the desired Tweets onto operators acting on the data in the Tweet payload. These operators fall into categories such as: keyword-matching, user attributes, geo-location, language, attached media, and Tweet status. Details can be found at the Gnip support site. These operators provide the vocabulary of the PowerTrack query language.

The goal behind the use of PowerTrack is to acquire Tweets that represent the presence of some topic on Twitter. A brand may wish to know about how their new packaging is perceived, a researcher may wish to know about users who discuss a particular sports team, or a news organization may wish to know what news is breaking within some demographic segment. We represent an attempt to extract these concepts with a logical combination of PowerTrack operators.

Content expertise is often important at this step, to better understand the atypical vocabulary and behavior in the relevant social conversation. We must also take care to account for approximately duplicate representations of the same thing. For example, a keyword could also be searched for as a hashtag; a keyword could be searched for in the Tweet body text or in the user’s biography text; or a geographic location could be searched for in the user’s profile-based location as well as their geo-tagged location.

Most topics of interest are initially difficult to represent as PowerTrack rules; this is why we emphasize an iterative approach: evaluate the Tweets from one query, determine sources of noise and signal, and update the rules. This cycle can be enhanced by including classification and filtering functions downstream from the PowerTrack interface. In the following sections, we will describe the key elements in this, and present some example workflows.

3 Downstream Filtering

3.1 Downstream classification and selection

When we clearly state the problem to be solved with Twitter data analysis, our statement almost always implies some requirements for the analysis. Downstream filters, combined with a good set of PowerTrack rules, allow us to more fully meet those requirements. If PowerTrack rules are the first way of tuning the analysis, then the choices for classifiers, filters, and evaluative metrics comprise a second interface for adjusting which Tweets are received for analysis. Like PowerTrack rules, these downstream functions classify Tweets. Unlike the PowerTrack interface, they don't necessarily classify to exactly two classes, and there's no requirement of filtering out specific class. Any selection of classes of Tweets may be retained for analysis.

Downstream filters and classifiers achieve two ends:

1. By filtering out more noise, they provide higher quality datasets for the final analysis
2. By making finer distinction between classes of Tweets and by providing more nuanced filtering of noise, they provide a good infrastructure for creating better PowerTrack rules.

In the discussion, we divide downstream filters into two basic types: simple filters acting on the Tweet payload, and trained, classifier-based filters.

Simple filters are constructed from Boolean-valued functions of the Tweet payload. Like PowerTrack rules, they are restricted to act on inputs in the Tweet payload. Unlike PowerTrack rules, they can be constructed from any custom function of the payload elements. Additionally, we are not required to throw away Tweets that are rejected by the filter. Such filters provide a way to create subsets of the original dataset defined by a set of PowerTrack rules. In a common example, simple downstream filters can be used to create individual datasets for each PowerTrack rule, rather than having all Tweets matching any PowerTrack rule in one data set. Such rule-specific data sets may represent different products, different locations, or any other distinction that was specified by a single PowerTrack rule.

It is also common to use downstream filters to identify classes of Tweets that are not separable by PowerTrack. For example, a simple function of the Tweet payload can easily identify Tweets that have received more than a certain number of “Likes”, or Tweets produced during a certain time window.

Finally, simple downstream filters allow us to create custom groups of Tweets or Twitter users. For example, we can do a simple audience segmentation by filtering on the presence of key terms in the user biographies.

Simple downstream filters are constructed from simple functions of the Tweet payload. Classifier-based filters, in contrast, depend on externally-optimized metadata or model parameters. In the simple case, the filtering function is still Boolean valued, but it depends on a model for classification. The most general case is when we classify all Tweets, producing new data sets for each class, and apply filtering and further analysis on a class-by-class basis. Some sample workflows with classifiers will be discussed in Section 4. For now, let’s review the different types of classifiers and how they are created.

A downstream, supervised classifier uses elements of the Tweet payload as a feature set and builds a model for separating Tweets or users into a predefined set of classes. We build the model by hand-labeling a set of Tweets with the true class label, and then training the model to assign those labels to an arbitrary Tweet. The predefined labels often indicate how the Tweets in a given class are to be handled downstream. For example, we could segment a set of users mentioning an event into Highly-Engaged, Engaged, and Lightly-Engaged classes. We might then discard Tweets from users in the Lightly-Engaged class, save data from the Highly-Engaged class for downstream analysis, and feed the properties of the Engaged class users back into a refined PowerTrack Rule.

A downstream, unsupervised classifier uses elements of the Tweet payload as a feature set and segments the Tweets or users by whatever separation can be algorithmically discovered in the data. These techniques require no hand- labeling of desired or undesired Tweets. We do typically still have to choose some model parameters, such as the number of clusters into which the Tweets are grouped. For each output class, we can evaluate it and, based on that evaluation, choose a label and decide whether to discard it, study it further, use it as an auxiliary sample, or any other choice appropriate to our understanding of its properties.

There are many choices to make in the use, selection, and training of any sort of downstream filter. For simple filters, we need to decide the criteria for accepting or rejecting a Tweet. In addition to this, classifier-based filters can require many input choices: model choice, feature selection, and class labels, just to name a few. To make any of these choices, we need to refer to the requirements of the original problem. This, in turn, usually requires that we evaluate various Tweet sets for quality metrics and other figures-of-merit.

3.2 Tweet evaluation

Tweet evaluation is a necessary step for understanding whether a set of Tweets meets the needs of our analysis. Speaking broadly, it tells us how much of what we're getting or not getting is what we want, and how much of what we're getting or not getting is what we don't want. More specifically, for a set of Tweets derived from any combination of filters, evaluation is the process of describing the data and relating that description to a set of desired qualities derived from the primary business question.

We often start the evaluation from a descriptive perspective. This step can include the calculation of the most frequently observed words, sequences of words, hashtags, links, and users. These lists allow us to identify common themes of conversation, areas of interest, and patterns of speech found in the text of Tweets. This set of results, along with other properties of the human interactions in the Tweet data, forms a descriptive conversation analysis. We can similarly list common terms and interests in users' bios, geographic and gender distributions, and other user properties to form a descriptive audience analysis. Both types of analysis illuminate types of noise (data we have but don't want) and missing classes of signal (data we want but don't have).

Descriptive analyses of audience and conversation allow us to make qualitative judgments of the quality of our data. In addition to surfacing noise and missing classes of signal, these descriptions can also allow us to name groups of Tweets. For example, the unsupervised classification mentioned earlier allows us to algorithmically group similar Tweets, but it doesn't provide a description of –or name for– a group of Tweets, or even guarantee that a helpful description exists. Lists like the most frequent terms in Tweet bodies or user bios sometimes allow for the naming of groups of Tweets, which in turn makes is easier to assess the suitability of a group for the analysis.

A quantitative evaluation of a group of Tweets require an additional ingredient: labels which define the class to which a Tweet belongs. We sometimes call these “truth labels” or the “ground truth”, and they are distinct from the class labels discussed above. In the simplest case, the truth label could be just “signal” or “noise”, or “in” or “out”, or any appropriate set of quantitative signals that maps to these concepts. For a set of Tweets, the fraction that are labeled “signal” is called the precision, and this quality metric is one of the most important measures of data quality. Precision can be calculated for any set of Tweets for which we can obtain truth labels. This could be all the Tweets delivered from a Gnip API, all the Tweets passing some simple downstream filter, or all Tweets that end up with a particular class label.

In contrast to the precision metric, recall is the fraction of the desired (“signal”) Tweets that pass a filter. Here, the filter could be PowerTrack or any downstream filter. Recall can be difficult to calculate when, as with PowerTrack filters, we don’t have access to the body of desired Tweets before the filter. We discuss techniques for estimating recall in such cases in Section 4.

The calculation of precision and recall is complicated. First, there is the simple fact that the concepts of signal and noise are contextual: one person’s signal is another person’s noise. For this reason, subject-matter expertise is essential for at least some of the labeling. Secondly, truth labels are not binary for many important questions. For example, an evaluation of Twitter user bios rarely segments users neatly into and out of an audience. Rather, users often fall into more than one class or category of relevant audiences. While a trained classifier may ultimately be capable of reproducing this separation, a subject matter expert is still required to identify the relevant audience segments and to apply labels for training. Finally, it is impractical to hand-label the ground truth for all Tweets. Typically, labeling a small subsample is sufficient, but the size of the training sample contributes directly the misclassification rate. A classifier trained on expertly-labeled data may be able to automate large-scale labeling, but this must be cross-checked carefully and regularly.

We have discussed PowerTrack filters, as well as a variety of downstream classifiers and filtering techniques. We have also discussed conversation and audience analysis, and defined some quantitative measures of data quality that allow us to measure our filter performance against the analysis needs. Now, we will give examples of how all these elements are combined in a cycle of data quality improvement.

4 Example Workflows

The tools and techniques discussed thus far form a linear pipeline of data flow and analysis: we receive Tweets that match PowerTrack rules, we apply downstream filters, and we analyze the resulting groups of Tweets. In reality, a single pass through this chain rarely results in acceptable data for downstream analysis. Instead, we iterate on some or all of the links in this chain of tools, and update the state of the system based on an analysis of the results of the previous version.

When planning an analysis project based on Twitter data, there are some general considerations that apply to almost any project. As early as possible, identify the key metrics to be calculated, and make an estimate of acceptable values for those metrics. Stop the iteration when those values are achieved. The project may entail many cycles of evaluation and adjustment, but its arc often can look like:

1. An initial guess for PowerTrack rules
2. Manual evaluation of the resulting Tweets, applying domain expertise
3. Improved PowerTrack rules, based on:
4. increasing topic recall by creating more inclusive rules
5. removing noise by adding negations to rules
6. More sophisticated downstream filtering, possibly feeding back into PowerTrack rules
7. Rule rationalization for volume and cost

With this arc in mind, we next present examples of specific analysis projects and workflows. There are common operations in all these projects:

1. Delivery of Tweets from Gnip APIs, according to a set of PowerTrack rules (described in Section 2)
2. Tweet classification and filtering (described in Section 3)
3. Evaluation of a set of Tweets, with or without ground-truth labels (described in Section 3)
4. Collation of evaluation results

Each example includes a description of the process and its goal, a diagram of a software architecture that could power the analysis, and a discussion of the strengths and weaknesses of the procedure. In each diagram, we denote the fixed interface points, such as the Gnip APIs, with green rectangles; we denote the user actions and associated tools, such as a Tweet evaluation function, with blue rectangles with rounded corners; and we denote the results of an iteration of the system, such as identified noise sources, with red ovals.

4.1 Simple, manual noise reduction

We often have the simple goal of removing noise from our data set. The problem here is poor precision, and we start with a solution based on refined PowerTrack rules.

The process of cleaning the data may look something like the following:

1. We start by specifying the time period, and we make an initial guess at PowerTrack rules.
2. From the resulting Tweet data, we calculate the total volume of Tweets, and the Tweet volume per PowerTrack rule.
3. For each rule, we also identify the most frequent terms and hashtags found in the Tweet bodies and the users' bios.
4. Based on the sources of noise found in the Tweet evaluation, we completely drop some rules, and we add negation clauses to other rules
5. Finally, we repeat the whole process with new PowerTrack rules, and continue to iterate until the Tweet evaluation reveals a suitable lack of noise.

Figure 1 shows a simple architecture for performing this simple manual noise reduction.

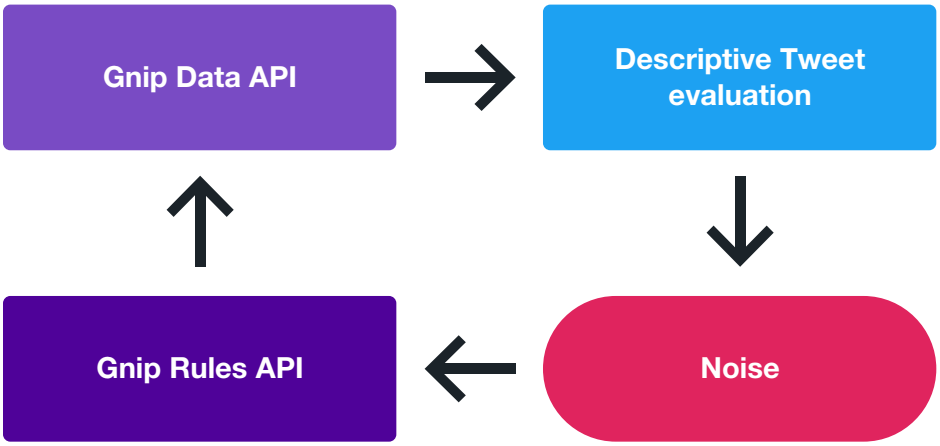

Figure 1: A simple architecture for performing this simple manual noise reduction.

The strengths of this technique are: simplicity, no need for per-Tweet ground truth labels, and the direct use of content expertise to identify noise. Its weaknesses include the inability to address or improve recall, that iterations require repeated calls to Gnip APIs, and the lack of sophistication around the definitions and identification of signal and noise. This technique provides a good place to start improving your data quality.

4.2 Data segmentation with unsupervised clustering

For some analyses, we need to segment a heterogeneous set of Tweets and Twitter users to achieve better precision and make cleaner labels of data. This process may look something like the following:

1. As usual, we start by specifying the time period, and we make an initial guess at PowerTrack rules.
2. With the Tweets received from the Gnip APIs, we use an unsupervised classification process [1] to algorithmically discover groupings within the data. This can be performed on all Tweets, or –if the PowerTrack rules identify substantially different topics– on each rule output individually. We typically perform grouping by the text in the body of the Tweet or by the text in the user's Twitter biography.
3. As with PowerTrack rules in the previous example, we calculate the volume of Tweets in each output group, or class. We also calculate the suite of descriptive statistics, including top terms and hashtags. The descriptive analysis allows us to identify interesting, missing, irrelevant, or noisy classes, and in many cases to name them.

4. Then, we update the PowerTrack rule set by:
    - Removing rules that contribute significantly to noisy classes
    - Adding negation clauses to remove noise from rules with significant signal contribution
    - Adding rules to increase the recall for relevant and missing signal classes
5. Finally, we repeat the process until the needs of the analysis are met. This may involve rerunning the segmentation process multiple time.

Figure 2 shows an analysis architecture for performing the filtering process.

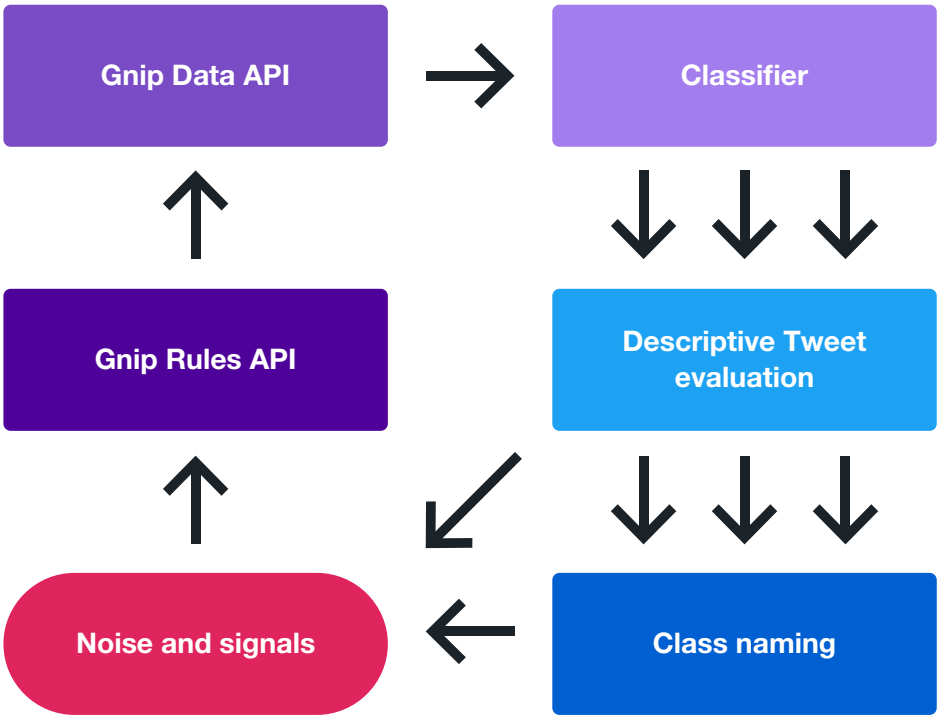

Figure 2: An architecture diagram for using unsupervised Tweet clustering to improve filter precision.

This technique naturally identifies structure in the data that goes beyond the positive/negative binary split, and requires no labeled training data. But choosing the number of groups is hard [2], and the technique can be over-complicated if simple labels are desired. We recommend this technique when the Tweet data matching your PowerTrack rules is likely to contain more than one relevant topic of conversation or more that one distinctive group of users.

4.3 Precision improvement with a trained classifier

When we need higher precision, but can't construct an adequate filter from only PowerTrack rules, we can use a trained Tweet classifier. This is a two-level technique, with iteration on a inner, classifier-based cycle, as well as on an outer, PowerTrack-based cycle. A typical use may have the following steps:

1. We start by specifying the time period, and we make an initial guess at PowerTrack rules, aiming for fairly broader rules with high estimated recall.
2. A subject matter expert or other hired human computation labels a subset of the resulting Tweets.
3. We train a classifier with a training subset of the labeled Tweets, and verify the model with a test subset of the labeled Tweets.
4. We label the remaining Tweets and assess the model performance by do- ing a panel of descriptive statistics, spot-checking individual Tweets, or performing further labeling. These results can be used to retrain the classifier. If the classifier performance peaks at an unacceptable level after a few iterations, we then move our workflow out to the level of PowerTrack rule refinement.
5. As in previous examples, we reduce persistence sources of noise by modifying or eliminating PowerTrack rules, and we include missing classes of signal by broadening and adding PowerTrack rules.
6. With new data from Gnip APIs, we re-classify the Tweets, which may require relabeling a training data set and retraining the classifier.
7. The cycles of iteration stop when precision calculated on a test sample is sufficient, or when the descriptive statistics indicate a qualitatively clean sample.

Figure 3 shows an analysis architecture for performing this filtering process.

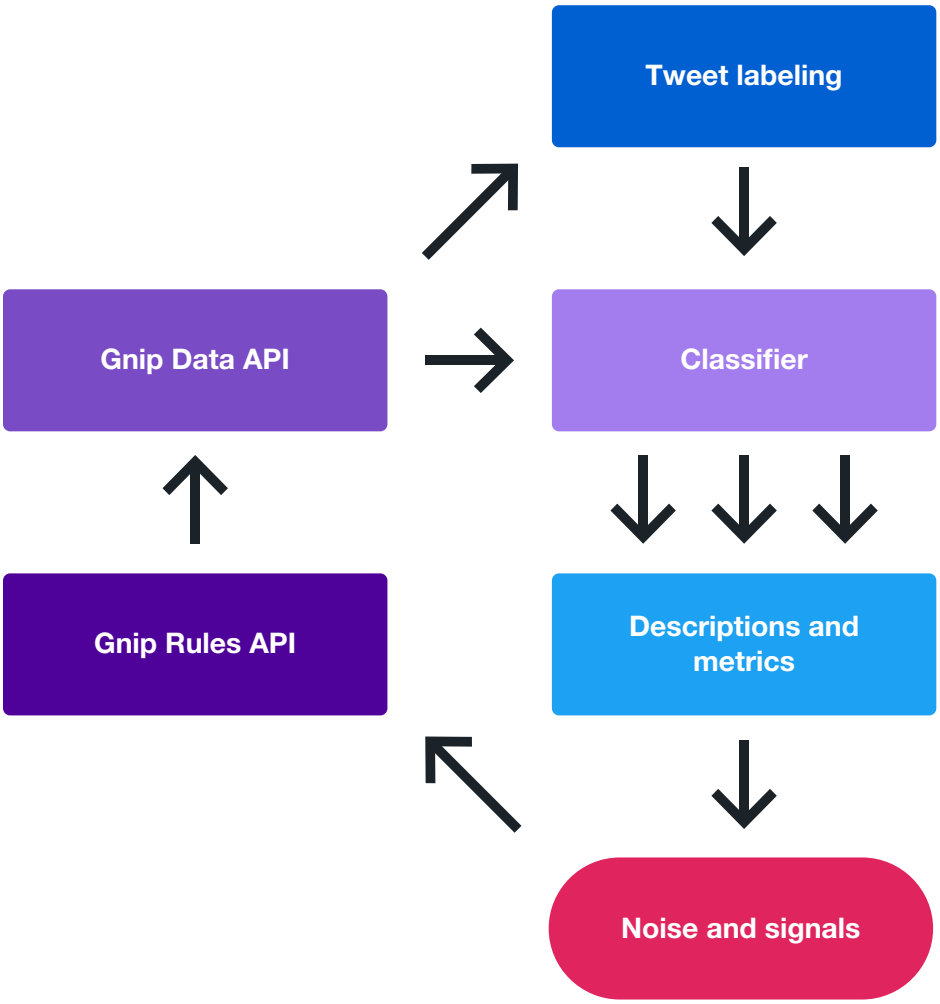

Figure 3: An architecture diagram for using a trained classifier to improve filter precision.

This two-level technique is a powerful system for identifying many classes of signal and noise, and increasing the precision of those signals through iteration of PowerTrack rules and trained classifiers. Its complicated nature means that it is a poor choice for an immature analysis. Additionally, it depends on hand- labeling in each iteration, which requires significant investment.

4.4 Improving recall

Achieving higher recall is an important but challenging goal. In effect, we're ask- ing the question: what is preventing me from receiving and/or correctly labeling desired data? Better recall reduces bias by incorporating a broader notion of the signal, and it more reliably reproduces conversations and other linked groups of Tweets. This task is made difficult by the fact that we don't have access to the full signal before PowerTrack rules are applied. We suggest two techniques for making this improvement.

For increasing the recall of PowerTrack filters, we recommend that a subject matter expert review the received Tweets and use their expertise to identify missing classes of Tweets. For example, a search for Tweets relating to the Leicester City football club might start with a PowerTrack rule like "Leicester football". This rule contains an implicit AND operator, meaning that it will match any Tweet containing both "Leicester" and "football". An analysis of the most frequent hashtags in the resulting Tweets indicates that users often tag a Tweet about Leicester City with the hashtag "#LCFC". An updated PowerTrack rule like "(Leicester football) OR #LCFC" would return additional Tweets about Leicester City that did not contain the keywords "football" or "Leicester". Recall has been increased, despite not truly knowing the full set of Tweets about Leicester City. This technique can be implemented with the architecture shown in Figure 1.

Once the recall of the PowerTrack filter has been increased to meet the needs of the end analysis, we can use downstream filters and classifiers to optimize the final precision. Because we can label Tweets received from Gnip APIs before further filtering, we can calculate both precision and recall for any class of Tweets passing a downstream filter. With these metrics in hand, we can optimize for precision, distinction between signal classes, or any other feature that is required by the analysis. This technique can be implemented with architecture shown in Figure 3.

For all examples just shown, there is a basic pattern: evaluation of data from PowerTrack leads to improved rules, as well as downstream filter parameters that produce better performance metrics.

5 Maintenance and Downstream Applications

Building a system that does the right thing once or twice is almost always easier than build a system that consistently does the right thing many times over changing conditions. In this section, we briefly discuss constructing a system that repeatedly supplies optimal results to downstream analysis applications. We then review how some the performance of common downstream applications is affected by changes in data quality.

We start by discussing the management of PowerTrack rules. Breaking a long rule into multiple rules can help clarify or simplify the intent of each rule. And because each Tweet is tagged with its matching rule, this provides easy rule management downstream. Early on in the exploratory phase of a data collection project, PowerTrack rules change often. While flexibility and fast iteration is important, we recommend maintaining a careful record of exactly what rules were used to pull each data set. A rules database is a good way of maintaining not only data set labels and exact rule values, but also metrics like data set sizes and rule production rates. Such a database allows us to look at differential rule production and estimate the importance of specific rules.

The base of users interacting on Twitter is constantly changing, so any system optimized for Twitter data requires periodic maintenance. We recommend an evaluation scheme that periodically confirms that the analysis goals for data quality are still being met. This could be an online system continually or periodically running on streaming data, or it could run offline on batches of stored Tweets. We could check for stability of performance metrics within a single data set, or we could measure stability over time. We want to demonstrate stability over changes in overall Tweet volume, as well as changes in the Twitter user base. Continual evaluation of data quality ensures that your data-based answers to important questions remain reliable.

In general, classifiers and filters acting on streams of PowerTrack data are not sufficient to answer high-level business questions. Downstream analysis applications are often needed, and their reliability and effectiveness is determined by the input data quality. For example, conversation reconstruction and natural language processing are common and valuable techniques for understanding user sentiment, yet their effectiveness is a strong function of the Tweet quality. A sentiment score can be highly skewed by noisy Tweets that have little to do with the topic of interest. Community

detection and influence measures depend on full fidelity description of the relationships between Twitter users. As connections are missed, the quality of these techniques drops.

To answer almost any realistic question with Twitter data, we need high quality sets of Tweets on which to perform an analysis. This quality is determined by filter performance metrics such as precision and recall, and by their mapping onto the requirements of the business question. With a clear business question and the examples of data quality refinement shown here, you are well on your way to precise, effective answers.

References

[1] a. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” ACM Computing Surveys, vol. 31, no. 3, pp. 264–323, sep 1999. [Online]. Available: http://portal.acm.org/citation.cfm?doid=331499.331504

[2] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data via the gap statistic,” J. R. Statist. Soc. B, vol. 63, pp. 411–423, 2001.

303.997.7488 | info@gnip.com | gnip.com | @gnip