Unique and Newsworthy Tweets

An ETL Approach to Identifying Niche Content

John Beckfield[*]
Dexcom
6340 Sequence Drive
San Diego, CA 92121
jbeckfie@eng.ucsd.edu

Martin Malasky[†]
Advantage Solutions
18100 Von Karman Ave #1000
Irvine, CA 92612
mmalasky@eng.ucsd.edu

Daniel Mariano[‡]
Cubic Transportation
5650 Kearny Mesa Rd
San Diego, CA 92111
dmariano@eng.ucsd.edu

Cheng-Hao Tai[§]
Evisort Inc.
1411 Industrial Rd
San Carlos, CA 94070
c2tai@eng.ucsd.edu

ABSTRACT

In the modern world, microblogs like Reddit or Twitter have become the primary source of information and news for many people. With a currently-sizable and increasingly-growing population of users, microblogs and the entities that populate them with information have the potential to easily sway popular opinion regarding just about any topic. More than ever before, it is important that individuals are capable of distinguishing important information. This paper discusses an attempt to implement an algorithm that can automatically identify both unique and newsworthy messages on Twitter.

A report component for DSE203's final project, this paper will provide an in-depth account of our team's effort to recreate a portion of a pre-existing investigation into applying ETL methods to discover insights between Tweets and mainstream news articles. The goal of our adaptation was to propose an efficient structuring of data sources and data pipelines such that tweets for any given day can be easily grouped by topics, novelty, and information-quality.

In achieving this goal, we utilized two separate database technologies (Postgres and MongoDB), cloud services, cluster analysis and topic modeling techniques, and several natural language processing strategies. We were able to accomplish our goal of implementing a metric for evaluating the *uniqueness* and *newsworthiness* of Twitter messages and successfully incorporated this algorithm into a query-friendly configuration.

Categories and Subject Descriptors

H.2 [**Database Management**]: General; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

Keywords

LDA, topic modeling, Postgres, MongoDB, dataflow specification

1. INTRODUCTION

At the time of this writing, Reddit and Twitter are (approximately) ranked 6th and 13th amongst the world's most popular websites[1]. With such a significant and pervasive readership, information has the potential of reaching a vast audience with unprecedented speed. The quality of information, however, is often questionable. One of microblogs' greatest appeals is the ability for just about anyone to contribute in an international forum. Unfortunately, this point is also one of the largest drawbacks to the microblogging platform. Unrestricted and (to some extent) unregulated participation in information exchange has created a free-for-all landscape of competing biases, facts, and speculations.

By itself, the internet community's prolific usage of microblogs isn't much of an issue. However, when compounded with the fact that a significant proportion of microblog users rely upon these platforms for news, several problems begin to arise. Without a proper vetting strategy to reliably distinguish newsworthy information from untrustworthy noise, it is easy for the general population to be misled by bad actors. Hence, as microblogging becomes increasingly integrated into peoples' lives, efforts must be made to develop robust frameworks with which to filter for truly newsworthy information.

Of course, that's not to say that microblogging doesn't offer revolutionary benefits that could potentially transform

[*]Responsible for News Branch and Tweet Branch

[†]Responsible for Collections Branch

[‡]Responsible for Newsworthy Branch

[§]Responsible for Clustering Analysis and Project Documentation

DSE203 '18 San Diego, California USA

[1]Statistic taken from Alexa top 50 global sites

the way that news is delivered. By allowing free and open participation, platforms like Twitter and Reddit are uniquely positioned to host newsworthy information faster and with a more diverse range of perspectives than modern mainstream news outlets. Even if these modern news outlets were to also participate on such microblogs (which they do), the myriad of individual users enables newsworthy and unique information to be shared that might otherwise be too troublesome for large news outlets to report upon. As such, it would behoove any potential vetting metric to also incorporate a filter for selecting such *niche* content as well.

The following sections will detail relevant sections of the research paper that we drew inspiration from and then offer a step-by-step walk-through of our overall implementation strategy.

2. THE QUERIES

Before launching into any in-depth discussions, we'd like to offer perspective into the ultimate purpose of this ETL exercise by enumerating the tasks / questions that our final queries will seek to answer:

1. How many twitter messages exist for a specific news topic for a specific day?
2. What are the top-10 most newsworthy tweets for a specific news topic on a specific day?
3. What is the percentage of newsworthy and unique tweets for a specific news topic on a specific day?

Note that each one of our queries aim to uncover insights relating to specific 1-day intervals. This was a conscious design choice that reflects the ever-changing nature of microblogs.[2]

The sections below will delve into the details of the theory, technologies, and methods that were utilized to satisfy these queries.

3. DATA AND DATABASES

3.1 Description of Data

In this ETL exercise, we had access to two sources of data: tweets (from Twitter) and news articles (from both Reuters and modern-day mainstream outlets).

3.2 Description of Databases

We used two types of databases: Postgres and MongoDB. All tweet data was stored and queried from inside MongoDB while news articles was kept inside Postgres. Furthermore, all intermediate results along our ETL pipeline are also stored inside Postgres.

4. THE PAPER

The paper that we referenced for the purposes of this ETL exercise was *Through the Grapevine: A Comparison of News in Microblogs and Traditional Media*[3]. In this paper, the authors set out to establish (amongst other things) a filtering algorithm capable of locating *unique* and *newsworthy* tweets in a large corpus of Twitter messages. The full breadth of topics that this paper covered is substantial and beyond the scope of the DSE203 course so as such, we only adapted a portion of the original research (to be described further below). This section will outline the overarching concepts behind the vetting metric and exactly how we chose to modify it for our purposes.

4.1 The Vetting Metric

The original vetting metric used in *Through the Grapevine* is composed of two parameters described in Table 1 and visualized in Figure 1. Recall that the main purpose of our ETL exercise was to find both *unique* and *newsworthy* tweets. In Table 1, the *Score* parameter takes care of *uniqueness* while the $News_{Term}$ parameter encapsulates the notion of *newsworthiness*.

Both of the Table 1 parameters measure the similarity of any single tweet to either a traditional news source (Reuters) or modern-day news articles of a specific topic. The degree of similarity of any tweet is measured using a set intersection of that tweet's words with a pre-assembled lexicon. The intersection value is subsequently normalized by the length of a tweet to constrain similarity values to between $[0 - 1]$.

The exact lexicons with which these set intersections are carried out can be described as follows:

- Reuters: The entire Reuters corpus (taken from the NLTK library) that is preprocessed[4] and reduced to a lexicon of unique words
- News Topic: A collection of modern-day news articles **of the same topic** that is preprocessed and reduced to a lexicon of unique words

Reuters is one of the most authoritative and well-established sources of newsworthy information and as such, the expectation is that the more similar a tweet is to language that is present in the Reuters corpus, the more newsworthy that it can be assumed to be.

Set comparisons against collections of modern-day news articles (grouped by news topic) serve a different purpose, however. Whereas we desired high-similarity against the Reuters corpus (signaling high-quality content), we actually wanted *low* similarity with modern-day news. Intuitively, this would suggest that tweets meeting these two criteria would be at once newsworthy and also *distinct from* content published by mainstream news outlets. To sum it up, this strategy aims to uncover tweets that are both informative and niche (which is identified as the upper-left region of Figure 1).

4.2 A Case for Consistent Preprocessing

One of the significant challenges that the authors of *Through the Grapevine* encountered was the choice of preprocessing to be applied on their text data. Since the foundational algorithm behind their vetting metric is set intersection, seemingly trivial design choices in preliminary text transformation can have a big impact.[5]

[2]For example, the hashtag #Boston could be relevant to any number of topics including sports, education, or technology

[3]Authors (listed in the order that they were published): Byungkyu Kang, Haleigh Wright, Tobias Höllerer, Ambuj K. Singh, and John O?Donovan; published in 2017

[4]All text data used throughout this ETL exercise is subjected to the same preprocessing strategy, to be detailed in a subsequent section

[5]Take for example, the difference between lowercasing *president* versus *President*. Though both words are clearly referring to the same subject, a design choice of keeping a

Table 1: Vetting Metric Parameters

Metrics	Nomenclature	Description
Modern News Similarity	*Score*	X-Axis in Figure 1
Reuters Articles Similarity	$News_{Term}$	Y-Axis in Figure 1

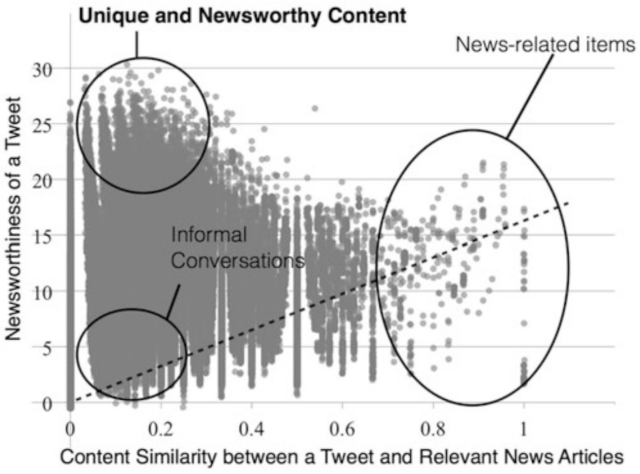

Figure 1: The tweet vetting metric presented in *Through the Grapevine*, visualized

In addressing this challenge, the authors of *Grapevine* implemented a customized optimization objective that incorporated both parameters summarized in Table 1. This objective is represented by an $InvDist$ metric that is described as follows:

$$argmax(InvDist) = \frac{1}{\mid News_{Term} - Score \mid +1} \quad (1)$$

Equation 1 was evaluated for every tweet across the authors' Twitter corpus and averaged to obtain a single value for each preprocessing strategy. Since each preprocessing approach (i.e. the choice of lowercasing, removing stop-words, n-gram tokenization, etc...) will have a direct impact on the Table 1 parameters, Equation 1 seeks to find a particular strategy that maximizes $InvDist$ by minimizing the difference between $News_{Term}$ and $Score$.[6]

Intuitively, this might not make much sense because our ultimate objective is to find tweets that are both very similar to Reuters and very different from modern-day mainstream news (which, at first glance, would benefit from maximizing the differences between the two Table 1 parameters). However, as Figure 2 demonstrates, excessive separation between the Reuters corpus and the modern-day news article corpus is actually undesirable.

While it is immediately obvious that the (overly simplified) distribution in Figure 2 is easily separable, such a distribution of tweets is both utterly nonsensical and unrealistic.[7] Since our goal is to adapt the *Grapevine* methodology to real-world tweets, it makes sense to create a model that accurately reflects the relationship between the Reuters and modern news corpuses. By choosing a preprocessing strategy that maximizes the $InvDist$ objective, we are keeping true to the inherent relationship between the two news sources: that while there is a difference from Reuters to modern news, this difference isn't *that* significant.

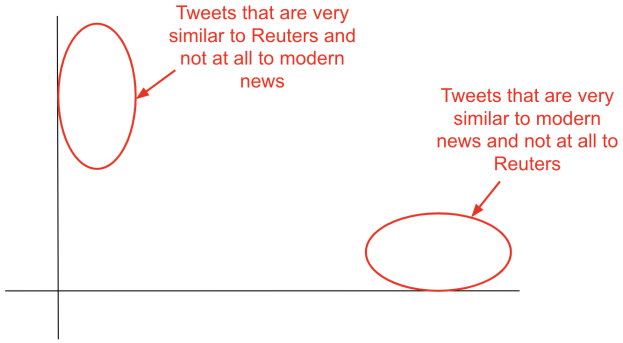

Figure 2: Figure 1 distribution in case of extreme separation between Reuters and modern-day news topic lexicons

The authors of *Grapevine* found that the following preprocessing minimizes the difference between Reuters and modern news articles:

- Stemming
- 1-Gram Tokenization

We decided to augment their vanilla strategy by adding the following steps:

- Lowercasing
- Stopword Removal

A key motivation behind adding additional preprocessing steps (at the risk of increasing the differences between Reuters and modern news) is that the two extra steps are expected to help with a critical element of our ETL pipeline: topic modeling.

4.3 The Need for Topic Modeling

Topic modeling wasn't a part of the original *Grapevine* paper since in that study, the authors conveniently had access to labeled tweets and modern-day news articles. In this way, they were able to create news topic groups simply through grouping by tagged keywords and in a similar fashion, assign tweets to particular news topic groups for set intersection calculations.

text's original casing could potentially allow such discrepancies to persist – thereby resulting in potentially lower-quality matches.

[6]Note that $InvDist$ has a maximum value of 1 which occurs in the scenario when $News_{Term}$ and $Score$ are the same (i.e. the Reuters lexicon and the modern-day news topic lexicon are identical)

[7]It will never be the case that modern-day news articles are astronomically different from traditional Reuters content such that one tweet can be identical to modern-day articles while not at all to Reuters and vice-versa.

However, the datasets that we worked with did not have such labels − hence, the for topic modeling. Specifically, we utilized Latent Dirichlet Allocation (LDA) through a Python library called `gensim`. The exact process flow for how this LDA model was implemented will be elaborated upon in a following section.

5. CLUSTERING ANALYSIS USING NMF

LDA is a generative probabilistic model that assigns documents into particular bins based on the occurrences of particular words or phrases. After it has been trained on a corpus of documents, it will be able to categorize new documents based on their similarity to the members of the training corpus.

However, LDA by itself can't figure out exactly how many topics to split a corpus of documents into; it needs user input. However, what if the user himself doesn't know how many topics exist in an existing corpus of documents? In this scenario, clustering techniques can be used to approximate the number of topics.

5.1 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (or NMF, for short) is a matrix decomposition technique that factorizes a single matrix into two distinct subcomponents. The logic of NMF is summarized as follows:

$$X \approx WH \tag{2}$$

$$X, W, H \geq 0$$

$$\min_{W,H \geq 0} \|X - WH\|_F^2 \tag{3}$$

As equation (2) explains, the purpose of NMF is to find a decomposition of an original matrix X such that the two subcomponents − when multiplied together − yield a product that is roughly equivalent to the original X. This decomposition process occurs iteratively through satisfying the optimization objective stipulated in equation (3).

The clustering aspect of NMF comes into play when looking at the dimensions of the factorized components W and H. Suppose that W has the dimensions $(n \times k)$ while H has the dimensions $(k \times m)$.[8]

A few observations can be made about this paradigm:

1. From a factorization perspective, it wouldn't make any sense for k to be larger than n or m
2. The value of k that results in the best error (as defined in equation (3)) corresponds to the ideal number of clusters within a dataset

5.2 NMF Process Flow

NMF[9] was applied on our corpus of modern-day news articles to provide LDA a specific number of topics for the purpose of topic modeling. The process flow for this procedure is detailed as follows:

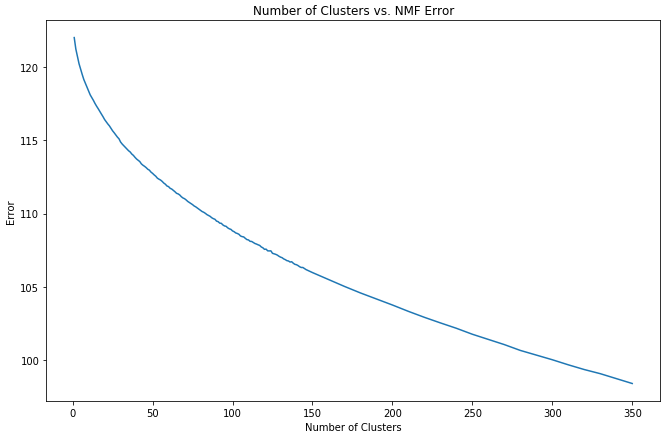

Figure 3: NMF errors with increasing cluster sizes

1. News articles over a 3-day period[10] was queried from the Postgres database
2. Article text for the specific 3-day period was preprocessed
3. A TFIDF Vectorizer[11] was trained on the preprocessed article text[12]
4. NMF models were trained with the follow cluster (k) sizes:
   - 1-150 (stride: 1)[13]
   - 160-350 (stride:10)
5. Errors for each cluster size was visualized (see Figure 3)

5.3 NMF Clustering Results

The results of our clustering analysis is summarized in Figure 3 which plots cluster sizes against NMF factorization errors. Figure 3 shows that as NMF factorization is performed with increasing cluster sizes (characterized by the k value), we obtain subcomponents that are increasingly-better representations of the original X matrix.

Ultimately, we decided to proceed with topic modeling using **205** clusters. However, it's not immediately obvious why this was the optimal choice since it can also be seen that the errors never actually converge along the spectrum of cluster sizes. To arrive at 205, we employed some qualitative measures. One of these heuristics was the training time of NMF models which − after reaching 250 clusters − took over an hour to train. The lengthy training times per cluster size combined with an aggressive project timeline resulted in us becoming unable to continue the clustering experiment to

[8]Which automatically implies that X has the dimensions $(n \times m)$

[9]Implemented using Python's `sklearn` library

[10]Despite our queries all working with 1-day intervals, we conducted our clustering investigation using a 3-day interval to serve as a buffer against underestimations

[11]Also implemented using Python's `sklearn` library

[12]The choice of using TFIDF as an embedding method was motivated by the NMF constraint that X must be composed of non-negative values

[13]In this context, stride refers to the number of clusters trained from one iteration to the next (1, 2, 3, 4, ... , 100) versus (160, 170, 180, ..., 340, 350)

locate a plateau or local minimum. At around 200 clusters, training time was still reasonable (around 30-45 minutes per cluster size).

Moreover, we were able to visually verify the quality of topics outputted from the LDA model. At 205 topics, LDA creates keyword sets that − for the most part − make immediate sense without any leaps in logic.

6. DATAFLOW SPECIFICATION

The clustering analysis is a process that happens independently of the overall process flow (which is illustrated in Figure 4).[14] On a high level, the entire data flow can be interpreted as four branches that are described as follows:

1. **News Branch**: Segregating a corpus of news articles into 205 topic bins and assigning each individual document to a specific bin
2. **Tweet Branch**: Linking tweets to news article topic bins by hashtags
3. **Collection Branch**: Loading tweets from MongoDB relating to a specific news topic
4. **Newsworthy Branch**: Calculating the uniqueness and newsworthiness of each tweet collected from MongoDB relating to a specific news topic

The following sections will discuss the specifics of these four main branches along with performance metrics measured over a day's worth of tweets and news articles.

7. NEWS BRANCH

The purpose of the News Branch is to define the topics used to link news corpuses to tweet hashtags, and to group the articles by topic to form distinct news topic corpuses.

We considered different strategies for handling the amount of data we had to preprocess and run through LDA, from using keywords to only considering the first 500 words of an article, but ultimately we found using the entire article text was very manageable, so we went with the full article text. A full day's worth of news articles took roughly 90 seconds to preprocess, generate topics, and sort into topic corpuses.

Topics change quickly on Twitter, so we decided that smaller partitions would reduce the chance that the topic of the tweets would change in the middle of a partition. News moves slower than tweets, but we wanted to compare contemporaneous news and tweets, so that set the news partitions to be the same as the tweet partitions. The news articles gave a date, but not a time, so that established one day as the minimum possible time partition.

The news articles are stored in comma separated files. We loaded them into a Postgres table, then queried for 1 days worth of news to run through the News Branch in Python. The news was first preprocessed.[15] Next the text of all the articles were combined and fed into a `gensim` LDA model, set to create 205 topics[16]. The trained LDA model was saved so that it could be used by the Tweet branch. Each news article was categorized into a topic by the LDA model, and the text was appended on to the corpus for that topic. Once this process is completed, we are left with 205 corpuses each with the text of all the articles that belong to the corresponding topic. These texts are written to Postgres, along with the day and topic number, to be used by the Newsworthy Branch to calculate a *Uniqueness* score. This process is repeated for each day.

[14]Keep in mind that this process flow is relevant for 1-day intervals

[15]The text preprocessing we did here, and in every future mention of preprocessing is removing all punctuation, putting everything to lowercase, removing stop-words, and stemming the text using Porter Stemming.

[16]Obtained from the Clustering Analysis in Section 5

8. TWEET BRANCH

The Tweet Branch is used to link tweets to the topics defined by the News Branch. We classified hashtags by topic based on the text of all tweets with that hashtag.

If we classified individual tweets, we could sort the highly unique tweets we are looking for incorrectly, since they do not have very many topic words by their definition. There is also the concern that tweets tend to use different diction than news articles, so individual tweets could be classified randomly if they share no tokens with the news topics. However, if we group all the tweets with a hashtag into a single document and classify that hashtag document, then it is likely that there will be enough tweets written with a formal enough diction to make the classification meaningful even if the more casual tweets are ignored.

There are enough Tweets that there is a concern that loading all the tweets in memory and processing them would cause resource and time issues, so we did the grouping through MongoDB, as well as limiting the tweet document size to be at most 200 tweets.

We used MongoDB because the tweet data was structured in semi-structured JSON, which is MongoDB's specialty. MongoDB can handle the grouping by hashtag and limiting the size of the tweet documents, which let us push those operations down.

We first loaded all the tweets for the relevant days into MongoDB, and wrote a query to take all the tweets for one specified day, group by hashtag to create a document of the texts of all the tweets of the hashtag, and limit the size of each document. The query took about 27 minutes to run. The documents were loaded into python, where the texts were preprocessed. The GENSIM model for the specific day was loaded, and then each hashtag was classified by the LDA model into one of the 205 topics. A table linking hashtag with day and topic number was created in Postgres, which is used by the Collection Branch to query for tweets by topic. Another table is created to show the important terms in each topic for the day so that the topics had a human readable form. *All the Python processing and sorting took under a minute for a day of data.*

9. COLLECTION BRANCH

The collection branch consisted of 200GB of tweet data in JSON format stored in a single document collection hosted on an Amazon Web Services (AWS) EC2 instance. This choice to opt for a cloud provider was simply the large volume of data on a local machine. The added benefit of an EC2 instance would allow high reliability as well as a shared instance of a single instantiation of MongoDB. AWS allowed the shared L1 memory to compute database queries on a large collection improving our performance.

The architecting of the collection branch in a single Mon-

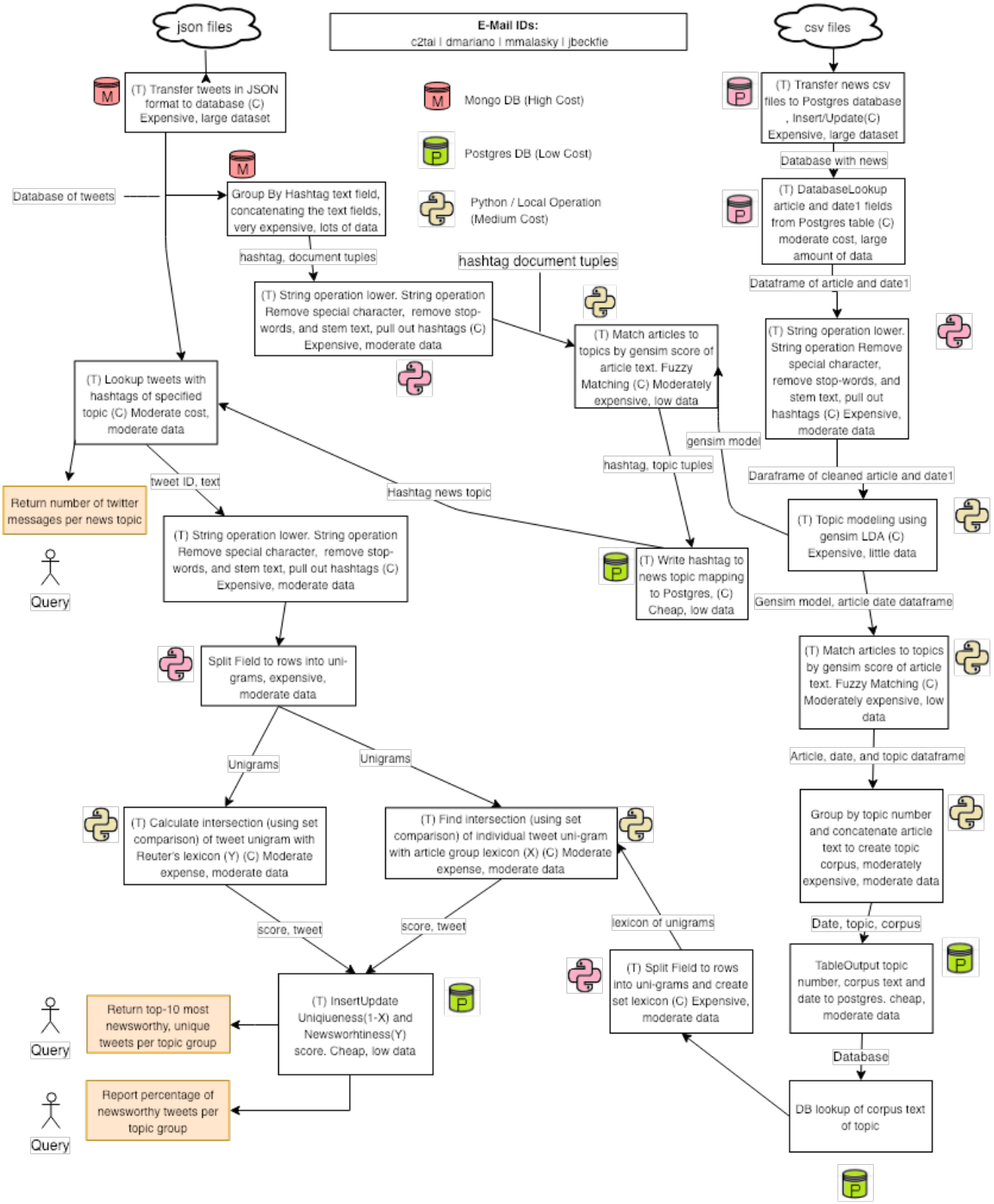

Figure 4: Dataflow specification of our ETL approach to finding unique and newsworthy tweets

goDB document was to allow dynamic queries to be executed between Postgres and MongoDB utilizing a synchronous Python script. The main concern with dynamically interacting between two separate database systems and performing calculations on the intersection of the two text fields was the memory allocation for both L1 and L2. Synchronous Python scripts allowed us to execute dynamic queries to iterate through our news corpus in Postgres, obtaining our partitions and related hashtag topics to be used as variables to execute and return associated tweet text from MongoDB. Computationally this is an expensive operation and therefore the decision was made that sequential execution of the queries would be returned into application memory on a local machine then sent to the newsworthy branch to perform the Reuters lexicon set intersection score and write back to Postgres. This synchronous approach allowed us to limit the volume and computational expense of processing numerous records without overloading our pipeline causing L1 or L2 to spill and crash our process. Ultimately there likely were better approaches which would allow a more parallelized or distributed approach to this problem, but for this exercise we were able to successfully iterate through entire dataset without issue.

A Python application was written to carry out the Collections Branch?s work. The process for this branch was to select a day and query all the relevant rows from the Hashtag-Topic table previously created in Postgres. Then for each LDA-defined topic for that same time period, we then queried the tweets collection in MongoDB. This query would search for tweets that include the corresponding hashtags to the current topic that is being iterated on. One item to consider is that as tweets can have multiple hashtags, those tweets will end up belonging to several topics. We chose to allow this in as tweets should be able to belong to two or more separate groups. For example, we would want to capture that a given tweet can be both about politics and sports in nature.

Once that query is returned, we then load the text of those tweets to pass along as input to the Newsworthy branch along with the appropriate day and topic number. This is done until all topics for all days are covered. A consideration we took when passing along the text of these tweets, is that we removed duplicates. When one considers the nature of twitter in social media, one can expect there to be a non- negligible amount of tweets with duplicate content due to items such as hashtag movements or spam-bots. As such, this step was done to lessen the amount of unnecessary calculations on repeated data as well as providing clearer results for our queries in the future. For instance, when executing Query 2 from Section 2, we wanted to showcase the top-10 most newsworthy unique tweets and eliminate the chance of having duplicates.

10. NEWSWORTHY BRANCH

This branch's goal is to score each tweet text passed as input in terms of *Newsworthiness* and *Uniqueness*. As mentioned earlier in section 4.1, *newsworthiness* is calculated by obtaining the set intersection with the Reuters corpus and *uniqueness* is calculated by obtaining the set intersection of a given news topic corpus. For the purposes of this paper, these two scores make up the metrics for how we evaluate tweets' contents.

Now prior to processing the input from the collections branch, we first imported the Reuters lexicon from NLTK and preprocess it. The preprocessing done here is the same as is outlined in section 4.2 (stemming, 1-gram tokenization, lowercasing, stop-word removal) and matches the same steps done in the other branches. The output was then stored in a pickle file so it could be easily loaded and used later. As a result, this Reuters segment of the branch only needed to be executed once.

The main part of this branch incorporates a Python module that will be called repeatedly by the collections branch to score all the tweets that we have stored. As input, this scoring Python module takes in a list of tweet texts where each element is the original unaltered text content of a tweet, a topic number representing one of the 205 LDA news topics, and a time partition representing the day for which these tweets occurred.

Once the proper parameters are passed in, the scoring module will first preprocess each of the elements in the tweet texts list in the exact same way as the news and the Reuters lexicon was done before it. After that finishes, we then pass along each tweet in the list through our calculation segment so that each one is scored in terms of newsworthiness and uniqueness. For newsworthiness in this segment, the Reuters lexicon pickle file that was created previously is first loaded. Then the set intersection between a given tweet and Reuters lexicon is calculated. This means obtaining the count of distinct elements that exist in both. The newsworthiness score for a given tweet is that count divided by the number of unigrams for that tweet. For the uniqueness score, we first obtain the news topic corpus from Postgres by querying from the *topic_corpus* table previously generated by the news branch. The query will filter out the given topic and partition module parameters. The corpus returned from that query was then split up by whitespace. Duplicate words in the corpus were removed as well. That corpus was then used in the same calculation as the newsworthiness to obtain the uniqueness score.

Upon conclusion of this recurring segment being called from the collections branch, all tweets will have received scores. Next, a dataframe was created containing the original texts, topic number, time partition, newsworthiness score, and uniqueness score. This dataframe was then uploaded to Postgres and appended to our *tweet_text_scores* table. From this table, we then can obtain the results for query 2 as well as query 3 from Section 2. The thresholds for what constituted as newsworthy or unique were user-defined. Tweaking of these thresholds were done by visual inspection of the results. In our experience with the data we worked with, we ended up having to keep our uniqueness threshold low. In our notebook code, one can see the output of we included of running a single day through the collection / newsworthy branches for 23 topics. 23 topics was the cutoff point due to the amount of time it took to process while also still having enough results to show off. It took around an hour and a half for one of our local machines to finish scoring that amount of tweets (4636).

11. FUTURE IMPROVEMENTS

There are many improvements to be made that would improve the results we found, let us run faster and easier, and have made the other improvements easier to find and implement.

The first category of improvements are improvements that

	text text
1	They can't the missile test or the carrier. Where is my money??? #india #trump
2	Trump to host Australian PM on February 23 - https://t.co/C6lbgrhgb8 #Diplomacy #India #International #Politics
3	Interesting review, I did not realise Colin Powell's involvement as a staff officer in #Vietnam . https://t.co/hj12G11Vsi
4	Supreme Court rejects petition seeking to make the crime of #rape #genderneutral
5	CBI Moves Supreme Court Against 12-Year-Old Order In Bofors Case https://t.co/zjfBMVxc85 #news #India https://t.co/BPFqJtXZ3z
6	Before it was Iraq, Syria, #Yemen, #Ukraine and Libya. Today it is #Venezuela. Tomorrow it will be #Iran. When... https://t.co/cnSRs2AVP7
7	RT @Paprika_im_Blut: I want all the troops home. Pull everyone out. Let other countries such as #Germany #France #Japan etc.. use their own...
8	speaking out: @THELITTLEIDIOT (February 2, 2018; 13:55 PST) #DONALDTRUMP #TRUMP #FAIL
9	I followed back all the #Resistance If I missed you, send me note. #Follow as many #Resistance as you can! Build t... https://t.co/7AQmgTh0UI
10	@CNN #Topic #Immigrants? So un America. Mr. #Penis #Pump! Answer to this: Here long, before #Columbus & you... https://t.co/ssqHaF0A1d

Figure 5: Example query 2 output

could improve the results of our queries. A problem with finding the top 10 newsworthy tweets that crossed a uniqueness threshold is that we had to keep the uniqueness threshold low, otherwise the only tweets we got were ones with no words at all. Our belief is that that is caused by the overlap of the topic corpus and the Reuters lexicon. Since the topic corpus is made up of news articles, eventually the topic corpus could entirely contain the Reuters lexicon, which would make Newsworthiness and Uniqueness completely inversely proportional. We propose that for the topic corpus we should instead take the set difference between the topic corpus and Reuters lexicon to capture the truly topic specific words.

Another issue we noticed is that for some topics on some days, there were only 100-200 tweets. This could lead to there just not being any unique newsworthy tweets. Since NMF never converged, using fewer topics isn't mathematically unjustifiable and using fewer topics would prevent topics with an insufficient number of tweets. A more drastic approach would be to eliminate topics altogether, and use the entirety of the news article texts as the corpus. If we are looking for niche topics in tweets, then it doesn't make a lot of sense to sort it into an unrelated topic, then determine that it doesn't relate to that topic rather than determine in one calculation that it is unrelated to all topics.

The second category is technical fixes to make the pain points in the flow easier. The MongoDB queries are the where the challenges are. The query making the hashtag documents should have saved off the results from when it had all the tweets of the day unwound by hashtag. Then the query to get all the tweets would have a lot less to do. Since we were only looking at hashtag and text, it might have even been worthwhile to load everything into Mongo, query to get text hashtag tuples, and load those into Postgres. More investigation into whether MongoDB was the right choice also could have surprised us.

What's more important than what specific technical fix would help us in this specific problem is the general case of what process would have let us find the above solutions earlier so that we could have implemented them. The optimistic answer is to have better communication. The query for the Tweet Branch and Collection Branch were done by different people, and they both came up with different solutions and neither checked in with the other on how their section could improve the others' section. The more pessimistic answer is have the same person do both Mongo queries. That way that person has the benefits of communication without having to be good at communication.

12. CONCLUSION

The ultimate results of our efforts was an ETL pipeline by which users can easily query for Tweets and related information regarding a notion of *newsworthiness*. Though our choice of topic modeling (LDA) resulted in some difficulties in obtaining consistently reliable topic groups, the final query results demonstrate that this approach is still capable of finding niche and high-quality Tweets. Though it is certainly true that more improvements can be made upon our framework, we are satisfied that our implementation proves that metrics for *newsworthiness* and *uniqueness* can be smoothly-integrated into an ETL structure that – without further fine-tuning – obtains reasonable performance.

13. ACKNOWLEDEGEMENTS

We'd like to thank Professor Amarnath Gupta and teaching assistants Ms. Jigyasa Grover and Mr. Mithun Dharmaraj for their steady guidance and invaluable advice throughout the various phases of this project. Without their efforts in providing us with the Tweet and modern-day news article, this ETL exercise wouldn't have been possible. Prof. Gupta was especially helpful in recommending NMF to us as a more robust and efficient replacement over K-Means.

In addition, we are also deeply indebted to the authors of *Through the Grapevine*. The metrics that they developed were the backbone behind our entire endeavor.