EECS 4415 - Big Data - Project Report
Computing TF-IDF Vectors for Subreddits

Ibrahim Suedan
York University
Toronto, Canada
isuedan@my.yorku.ca

Ken Tjhia
York University
Toronto, Canada
hexken@my.yorku.ca

Qijin Xu
York University
Toronto, Canada
jackxu@my.yorku.ca

ABSTRACT

Reddit is currently one of the most visited websites [6]. The large amount of traffic and user content on Reddit presents a great opportunity for advertisers. In order to provide advertisers with the information necessary to target, formulate, and evaluate their Reddit campaigns, we provide here a big data architecture capable of computing term frequency - inverse document frequency (TF-IDF) vectors for subreddits in a fully distributed manner. Moreover, for each subreddit we use these TF-IDF vectors to compute the top n most relevant terms and top m most similar subreddits, and show that these results make sense on an intuitive level. We store all data in a distributed file system and perform the computations in a distributed manner with PySpark, hence our architecture can scale to handle a large number of subreddits and comments; we tested with a small 1.2GB file of 1 million comments using a single node cluster (16GB RAM, i5-6500k), the computations completed in ~ 1 hour.

KEYWORDS

Reddit, TF-IDF, informtion-retrieval, Spark, PySpark

1 INTRODUCTION

Reddit is currently one of the most visited websites, averaging over 1.5 billion unique visitors monthly between February 2018 and August 2019. In July alone, Reddit saw 1.684 billion unique visitors post over 145 million comments [6]. This large amount of traffic and user content presents a great opportunity for advertisers; for 2021, Reddit projects a revenue of 267.1 million dollars from digital advertisement [1]. In order to provide advertisers with the information necessary to target, formulate, and evaluate their campaigns, we provide here a big data architecture capable of computing subreddit level term frequency - inverse document frequency vectors (TF-IDF) in a fully distributed manner.

TF-IDF is a popular term relevancy metric in the information retrieval community [4], and has previously been used to construct numerical vector representations of documents, which are especially useful for machine learning tasks. We create TF-IDF vectors for subreddits by forming a corpus where each document is the concatenation of all comments posted to a particular subreddit in a given month, then compute the TF-IDF vector for each document (subreddit). We use these vectors to compute the pairwise similarity between subreddits and identify, for each subreddit, the n most relevant terms and the m most similar subreddits. Because we have access to batch datasets containing all of the comments posted on Reddit in any given month, we can compute these quantities for any month.

Advertisers can then use this information to select which subreddits to focus their efforts on (for example target subreddits with relevant terms that are aligned with the advertising goals, or extend on-going campaigns to similar subreddits), to tailor campaigns to particular subreddits (for example use language in accordance with the subreddit relevant terms), and to evaluate the performance of ongoing campaigns (for example by monitoring the changes in term relevancy or subreddit similarities). More generally, the TF-IDF scores can be used as subreddit features for machine learning tasks such as subreddit classification or recommending subreddits to users. The results of our analysis can provide valuable information for businesses such as marketing or public relations firms, Reddit itself, and will also provide a dataset of subreddit representations/features for academics and other interested parties.

2 DATA AND DATA ANALYSIS

The Reddit website is organized into different subreddits, where each subreddit contains a number of posts, and each post contains a number of comments, each of which is a text string. We source our datasets in monthly batches; each dataset is a text file containing all comments posted to Reddit in a particular month, with each line containing one comment represented as a JSON object, in no particular order. Figure 1 presents an example comment; the


```
{
    "author": "ExampleCommenter",
    "author_flair_css_class": null,
    "author_flair_text": null,
    "body": "This is an example comment",
    "created_utc": 1270637661,
    "id": "c0nn9iq",
    "link_id": "t3_bne3u",
    "parent_id": "t1_c0nn5ux",
    "score": 2,
    "subreddit": "askscience",
    "subreddit_id": "t5_2qm4e"
}
```


Figure 1: Example of Reddit comment JSON object

relevant fields for our task are the *subreddit* and *body* fields. There are a number of optional fields which have been omitted for brevity.

2.1 Data Dimensions

Structure Semi-structured. Each comment follows a JSON schema, however there are a number of optional fields, and several fields containin unstructured data. In particular, the *body* field is a

text string of 10,000 characters maximum. We transform the data into a structured data set, however.

Size Medium. Each monthly batch dataset is about 180 GB.

Sink Rate Low. We download the datasets monthly.

Source Rate Low. We update our tables only once a month.

Quality Medium. How well our TF-IDF vectors represent a subreddit depends on how many words posted to this subreddit we can extract from the dataset. We require comments consisting of text that our pre-processing/cleaning stages will properly tokenize.

Completeness Medium. We only require the *subreddit* and *body* fields to be present. However, we require the dataset text file to be properly formatted, containing one JSON object per line.

2.2 Processing Dimensions

Query Selectivity High, Low. Once our processed data is in a HIVE table, both high and low selectivity queries can be handled efficiently.

Query Execution Time Short, Medium. Once our computations are done and stored in HIVE, most queries execute quickly, however it is possible to perform queries that may take longer.

Aggregation Advanced. Our processed data is stored in HIVE tables, which support advanced aggregation capabilities.

Processing Time Long. On our single node cluster (16GB RAM, i5-6500k) a 145, 965, 083 comment dataset will take over 24 hours to process. However, this time can be considerably reduced by adding more nodes to the cluster.

Join Advanced. Our processed data is stored in HIVE tables, which support advanced join capabilities.

Precision Exact. Though the cleaning and stop words removal steps can potentially remove or change words, we will always get the same results on a given dataset for a fixed configuration of the cleaning/pre-processing stages.

Our pre-processing and analysis is performed in a PySpark Pipeline [7]; we now briefly describe the stages.

2.3 Pre-processing

Each monthly batch dataset is a text file where each row is a JSON object of one Reddit comment, containing a number of fields (some optional) including populated *subreddit* and *body* fields, which we use as our data. Our first pre-processing step is to create a document for each subreddit by concatenating together all of the *body* strings corresponding to the subreddit. Next, we clean each document by converting them to all lowercase and removing all non-white space/non-alphabetical characters. Next, we filter out subreddits whose documents are less than some configurable number of characters (parameter *minlength*, set to 50,000 in our experiments). Finally, we tokenize the documents using PySpark's Tokenizer [7], then remove stop words using PySpark's StopWordsRemover [7].

2.4 Analysis

After the pre-processing and cleaning, we use PySpark's CountVectorizer [7] to compute a vocabulary over the dataset with two configurable parameters: *mindf* which controls the minimum number of documents a word must appear in to be included in the vocabulary, and *vocabsize* which controls the maximum number of words to include in the vocabulary. Computing this vocabulary requires one pass over the documents. Next, we use PySpark's IDF [7] to compute the TF-IDF scores for each document (subreddit), requiring another pass over the documents. Next, we find the *nwords* most relevant terms for each subreddit by identifying the *nwords* words with greatest TF-IDF scores. Next, we compute a matrix where the (i, j)th entry is the cosine similarity between the TF-IDF vectors of subreddits i and j. Finally, we use this matrix to identify the *nsubreddits* most similar subreddits for each subreddit.

As a result we have, for each subreddit: the *nwords* most relevant words (from the vocabulary) and their TF-IDF scores, the *nsubreddits* most similar subreddits (by cosine similarity of TF-IDF vectors) and their cosine similarities, the TF-IDF vector, and the vector of TF-IDF cosine similarities.

For a particular word and document, the TF-IDF score tells us how relevant that word is to the document. The TF-IDF vector for a document then tells us how relevant each term is to the document; the entries with greatest value correspond to the words (in the vocabulary) which are most relevant to the document (by the TF-IDF metric). We use CountVectorizer [7], rather than HashingTF [7], to preserve a one-to-one mapping between vector entries and words in the vocabulary. Then, because entries of each TF-IDF vector (for each subreddit) correspond to the same words, the cosine similarity between two document TF-IDF vectors tells us how similar the two subreddits are in terms of the relevance for words in the vocabulary. This analysis is done monthly, hence we can track over time how word relevance and subreddit similarities change, adding another dimension to the data. As mentioned, this information can be used by advertisers to monitor the effectiveness of ongoing campaigns, determine what language/themes are appropriate for targeting particular subreddits, or to determine which subreddits to target initially or extend an ongoing campaign to

3 ARCHITECTURE OF PROPOSED SOLUTION

3.1 Architecture

Ingestion We download our monthly datasets of Reddit comments from Pushshift.io [3].

Storage We store the raw dataset temporarily on a HDFS. After processing, our results are stored back into the HDFS on a HIVE table, and the raw data is deleted.

Processing We use Apache Spark [7] with the PySpark API to do all of our processing. We have written all of the computations to work on a cluster, the only 'large' data structure we keep in the Spark driver's local memory is a dictionary mapping subreddit names to rows in a distributed Spark DataFrame. This data structure is small, however, as there are currently only about 1.8 Million subreddits [2], many of which are inactive.

Serving After the processing, our results are stored back to the HDFS in a HIVE table. This table contains, for each month and subreddit: the TF-IDF vector, the *nwords* most relevant words and their TF-IDF scores, the *nsubreddits* most similar subreddits and their cosine similarities, and a vector of cosine similarities (ith entry is the cosine similarity between this subreddit and the subreddit with row index i). This table may be queried directly, but also an (unimplemented) Tableau dashboard connected to

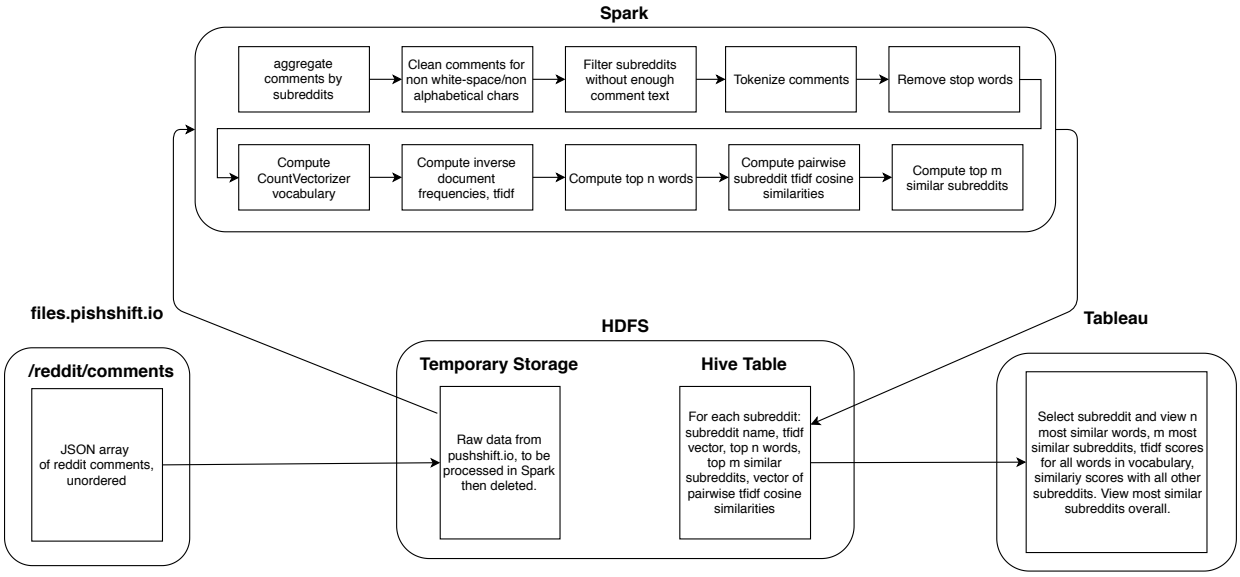


Figure 2: Architecture and Data Flow

the HIVE table can be used to select a month/subreddit and view the relevant quantities.

Visualization We would like to learn a 2d embedding function for each monthly dataset, so that within Tableau a user may select subreddits and have them plotted in a 2d Cartesian plane, giving a visual indivation of how similar the two subreddits are. Ideally, for this we would use principal component analysis (PCA) on the TF-IDF vectors to reduce them each down to 50 dimensions, then use t-distributed stochastic neighborhood embedding (T-SNE) to further reduce each vector to two dimensions. This remains unimplemented because PySpark's 'distributed' PCA computation requires the data covariance matrix to be computed and stored in the memory of a single Spark executor (i.e. not really distributed); time constraints did not allow us to write a truly distributed computation of PCA. Figure 3 depicts an examples of such a visualization, produced on a small dataset with Scikit-Learn [5].

Figure 2 provides an overview of this architecture including the pre-processing and cleaning steps done within Spark.

3.2 Limitations/Problems

Vocabulary CountVectorizer uses a fixed vocabulary, so we have to set *vocabsize* sufficiently large to ensure that all 'relevant' words are included. Furthermore, if we want word relevance and subreddit similarities to be comparable across different months, we have to use the same vocabulary for each month.

Text Cleaning Our vocabulary, and the resulting TF-IDF vectors, are dependent on the sets of words that we extract from the dataset for each subreddit. The words we extract are dependant on our choice of pre-processing an cleaning (text cleaning, stop words removal, filtering, etc.), and optimal configurations of these steps can not be determined beforehand (they depend on the particular dataset).

PCA We would like to include the PCA and T-SNE steps of our analysis in the PySpark Pipeline, however the PySpark implementation of PCA is not truly distributed as it computes the covariance matrix in the local memory of a single worker. As an extension, we could write a truly distributed PCA algorithm.

4 EVALUATION/RESULTS

We perform a proof of concept analysis on a 1.2 GB dataset consisting of a sample of 1M comments from July, 2019. We set *nwords* and *nsubreddits* both to 10, *minlength* to 50, 000, and *vocabsize* to 65, 536. After pre-processing, cleaning, and filtering, only 458 subreddits remain. The mean number of words per subreddit was 16, 000 and standard deviation was 5, 000, hence there was quite a bit of variance between the number of words in each subreddit. We perform the analysis using a single node Spark cluster with 16GB RAM and a 4-core i5-6500K processor. The analysis completed in $\sim$ 1 hour.

To evaluate how well our top *nwords* words represent subreddits, we first looked at the 10 subreddits with greatest mean TF-IDF scores for their set of relevant words. Unfortunately, the subreddits with greatest mean scores were not the best representations, they generally included some words that only resulted from our pre-processing/cleaning steps (see the *cars* subreddit in Figure 3). In general, however, we have found that the relevant words *are* representative of their corresponding subreddit. Figure 3 presents some examples.

subreddit	top_similar_words	top_similar_subreddits
nfl	[nfl, qb, brady, nba, chiefs, tyreek, brees, players, rodgers, patriots]	[CFB, nba, barstoolsports, NYKnicks, rockets, baseball, soccer, hockey, GoNets, changemyview]
boxoffice	[endgame, avatar, ffh, rerelease, disney, aladdin, movie, marvel, maleficent, theaters]	[movies, marvelstudios, horror, saltierthancrait, comicbooks, JusticeServed, mildlyinteresting, AskReddit, television, NoStupidQuestions]
investing	[boeing, fidelity, pnc, stocks, bitcoin, fund, dividends, blockchain, cma, engineers]	[technology, personalfinance, Bitcoin, CryptoCurrency, financialindependence, PersonalFinanceCanada, wallstreetbets, cscareerquestions, AusFinance, RealEstate]
pokemon	[pokemon, pokmon, gamefreak, dex, swsh, gen, graphics, animations, games, oras]	[gaming, pokemongo, pokemontrades, Games, patientgamers, NintendoSwitch, PS4, nintendo, fireemblem, TwoBestFriendsPlay]
CryptoCurrency	[bitcoin, crypto, vidt, btc, nano, rcryptocurrency, vechain, blockchain, rcryptocurrencymemes, subreddithttpswwwredditcommessagecomposetofrf]	[Bitcoin, investing, personalfinance, wallstreetbets, technology, singapore, AskReddit, worldnews, Piracy, learnprogramming]
Bitcoin	[bitcoin, btc, eps, electrum, wallet, node, coinbase, hodl, fiat, ip]	[CryptoCurrency, investing, techsupport, personalfinance, learnprogramming, worldnews, singapore, PersonalFinanceCanada, sysadmin, technology]
gaming	[pokemon, games, gaming, kinect, fps, wii, mario, dlc, game, steam]	[Games, pcmasterrace, patientgamers, pokemon, NintendoSwitch, PS4, xboxone, pcgaming, Amd, buildapcsales]
worldnews	[solar, trump, nuclear, sanctions, whaling, nk, coal, india, korea, kim]	[politics, Conservative, news, PoliticalHumor, The_Donald, ABoringDystopia, worldpolitics, collapse, PoliticalDiscussion, Libertarian]
CFB	[fuente, yards, qb, interception, vt, msu, auburn, osu, lsu, clemson]	[nfl, ApplyingToCollege, barstoolsports, soccer, baseball, nba, hockey, rockets, AskReddit, MLS]
cars	[subredditmessagecomposetorcars, cars, car, herehttpwwwredditcomrcarswikirules, manual, honda, coupe, rwd, acura, engine]	[MechanicAdvice, IdiotsInCars, teslamotors, motorcycles, vancouver, formula1, PersonalFinanceCanada, funny, PublicFreakout, AskReddit]

Figure 3: Sample of similar words and subreddits

To evaluate the subreddit similarities, we looked at the 100 subreddit pairs with greatest TF-IDF cosine similarity. Subjectively, the pairs were subreddits which we expect to have similar words. We omit the results, however, as many pairs contained the same subreddits and we feel the sample provided in Figure 3 is more interesting.

While the PCA and T-SNE stages are not implemented in PySpark, we perform the analysis on our small sample of 1M comments using ScikitLearn [5] to get an idea of how the results would look. In particular, we use PCA to reduce the TF-IDF vectors from 65536 dimensions to 50, then T-SNE to further reduce the dimensions down to 2. Figure 4 shows plots of the T-SNE transformed vectors for varying number of subreddits. It is apparent from the plots that the distribution of T-SNE vectors are concentrated in a region of 2-space, with a couple of outliers being situated far from this cluster. These outliers have significantly different TF-IDF vectors from those closer to the cluster center, indicating that these subreddits use significantly different language compared to subreddits closer to the cluster center.

5 CONCLUSION

Our analysis answers the questions of what are the *nwords* most relevant words and *nsubreddit* most similar subreddits for each subreddit. Our measure of relevance and similarity are based on the TF-IDF and cosine similarity metrics, which have previously been shown to provide satisfactory results for document term relevance and document similarity [4]. While a human investigator could glean this information for a single subreddit, our architecture allows for the processing of an arbitrary number of comments spread over an arbitrary number of subreddits, we are only limited by the hardware of our cluster, which easily scales horizontally due to our use of Spark. We store all of our data back into the HDFS in a HIVE table which allow for fast and easy querying. Because we store the monthly TF-IDF and cosine similarity vectors, we can even perform queries such as finding any number of most relevant words or most similar subreddits, or the top k most similar subreddit pairs for any k; we only provide the *nwords* words and *nsubreddit* subreddits in particular to give quick access to information relevant to an advertiser's usage as discussed in the introduction.

For future work, we would like to implement a distributed PCA computation in PySpark and add this stage to our Pipeline, along with the T-SNE embeddings. It may also prove useful to compute a data cube with our HIVE table to allow for online analytical processing. Lastly, it is clear that our pre-processing/cleaning steps have room for improvement, we noticed in particular that certain subreddits have high term relevance for URLs (see *cars* subreddit in Figure 3).

REFERENCES

[1] EMarketer. 2019. *Reddit to cross 100 million in ad revenues in 2019.* Retrieved Dec 24, 2019 from https://www.emarketer.com/content/reddit-to-cross-100-million-in-ad-revenues-in-2019

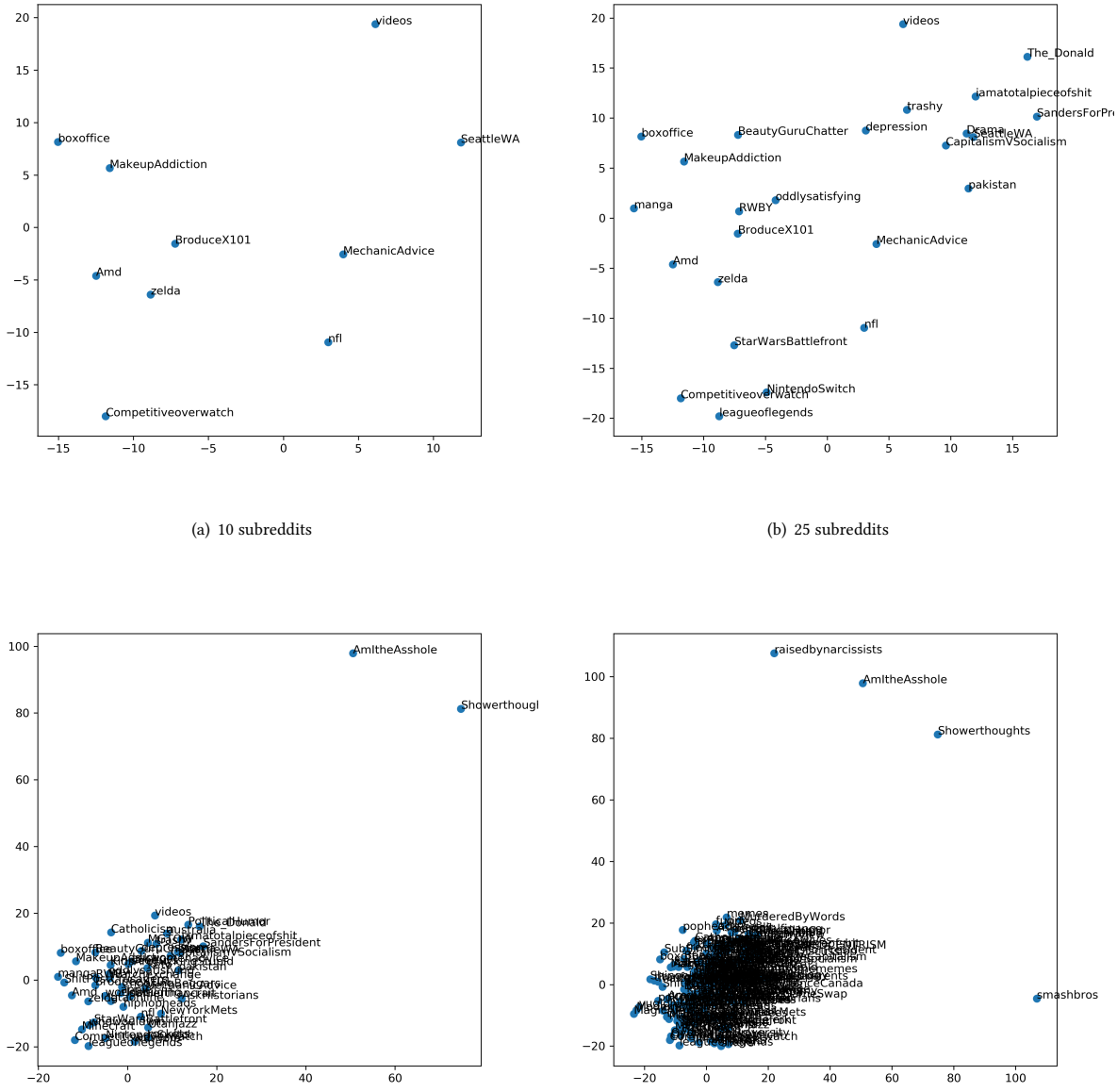
[2] Reddit Metrics. 2019. *New subreddits by date.* Retrieved Dec 24, 2019 from https://redditmetrics.com/history

[3] PushShift. 2019. *PushShift Reddit API.* Retrieved Nov 11, 2019 from https://files.pushshift.io/reddit/comments/

[4] Shahzad Qaiser and Ramsha Ali. 2018. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International Journal of Computer Applications* 181 (07 2018). https://doi.org/10.5120/ijca2018917395

[5] Scikit-Learn. 2019. *Scikit-Learn: Machine Learning in Python.* Retrieved Dec 24, 2019 from https://scikit-learn.org/stable/

[6] SimilarWeb. 2019. *Combined desktop and mobile visits to Reddit.com From February 2019 to July 2019.* Retrieved Nov 11, 2019 from https://www.statista.com/statistics/443332/reddit-monthly-visitors/



(a) 10 subreddits

(b) 25 subreddits

(c) 50 subreddits

(d) 200 subreddits

Figure 4: PCA to reduce to 50 dimensions, then T-SNE to reduce to 2

[7] Apache Spark. 2019. *PySpark 2.4.4 Documentation.* Retrieved Dec 24, 2019 from https://spark.apache.org/docs/latest/api/python/index.html