Twitter Topic Analysis using Apache Mahout

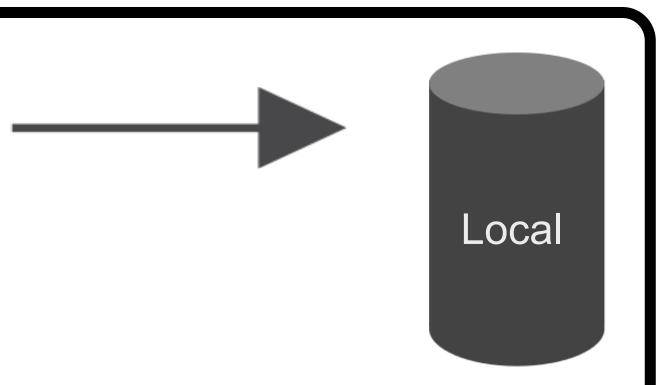
Eitan Romanoff
Thad Billing
Egezon Berisha

What's this all about?

As it becomes easier to share thoughts and sentiment online through social networking services, the amount of publicly available data grows with it. This information, in the form of natural text, can be acquired, processed, and analyzed to identify features such as trending topics across all users of these services. Companies can leverage this data to market intelligently, but as the amount of data increases, so does the need for scalable systems. In this project, we attempt to create a system to identify changes in the topical landscape over time.

Acquire.py



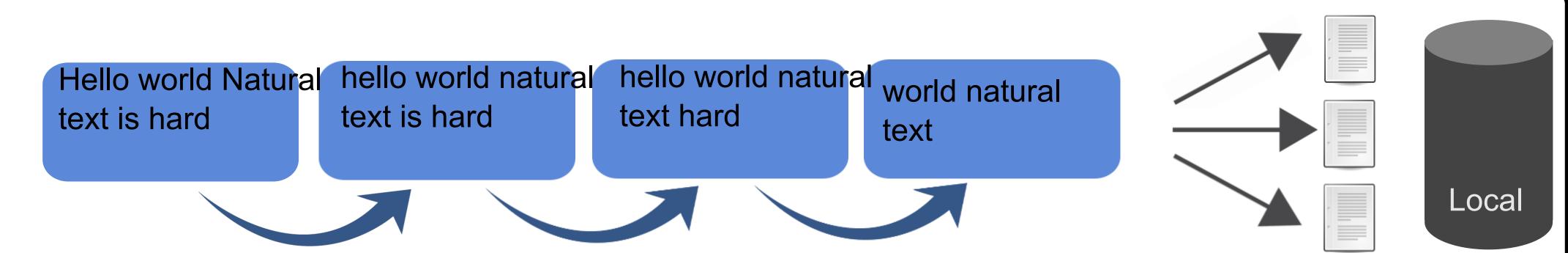


Data was acquired through usage of the Twitter Stream API, where we get a 1% sample of the overall twitter stream. Tweets come in as JSON, delimited by carriage returns, and we store these as a "dump" file on the local disk.

Fragment.py



For topic analysis on natural language text, only some of the fields are used. For our project, we use id_str for file fragmentation, the created_at timestamp for file binning, and the text and hashtags for analysis.



Each tweet is split into three fragment portions: JSON, Content, and Hashtags. The JSON portion is simply for the whole, pure storage of the tweet's metadata for potential future use. The Content fragment is the text of the tweet without any metadata. This portion of the tweet gets cleaned in a preprocessing stage prior to being stored as a file fragment, where the text is normalized, and stop words are removed. The Hashtag fragment contains only the hashtags of the tweet. These are stored on the local disk in a directory hierarchy, but ideally, this would be placed on the HDFS. Unfortunately, HDFS is not designed for many small data files, which remains an unsolved issue.

Each file was binned based on the timestamp of the tweet. Creating the file hierarchy on the disk in this way is necessary for running the Seq2sparse utility in Mahout. Furthermore, this allows us to create vector "snapshots" between time periods. In the future, we may want to make the binning even more granular for small time steps.

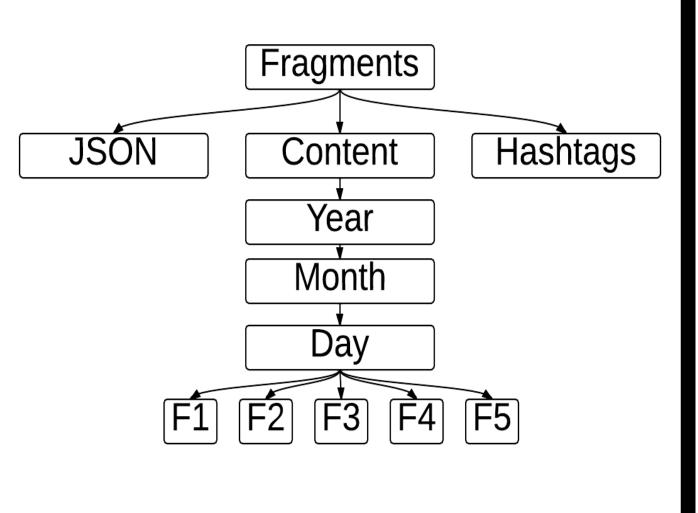
Sequence file containing the collected tweets for a time period into a large file to be used in processing.

Seq2sparse was executed to create TF vectors for utilization in LDA. Each term needed at least a frequency of 50 to be considered a topic for LDA. A dictionary of topics was generated for reference in retrieval of LDA Topics.

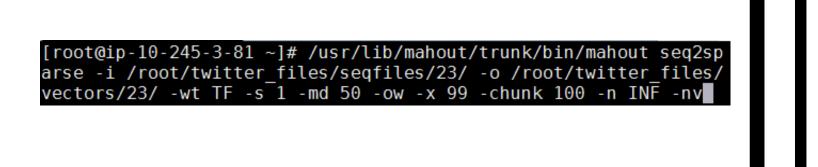
rowid converts text indexes to numerical values for input to LDA.

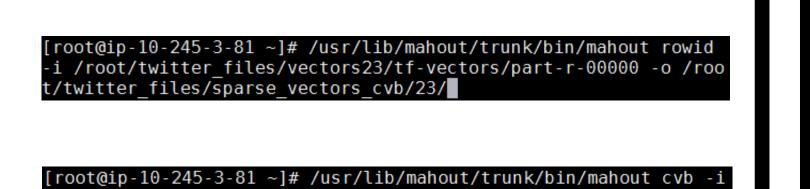
LDA is executed against the data set for 30 topics over 20 iterations.

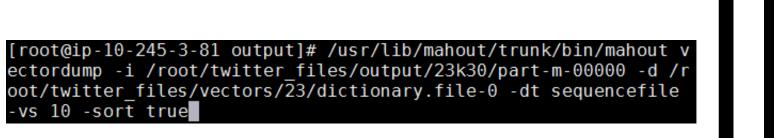
A vectordump output is generated for the resulting LDA topics.

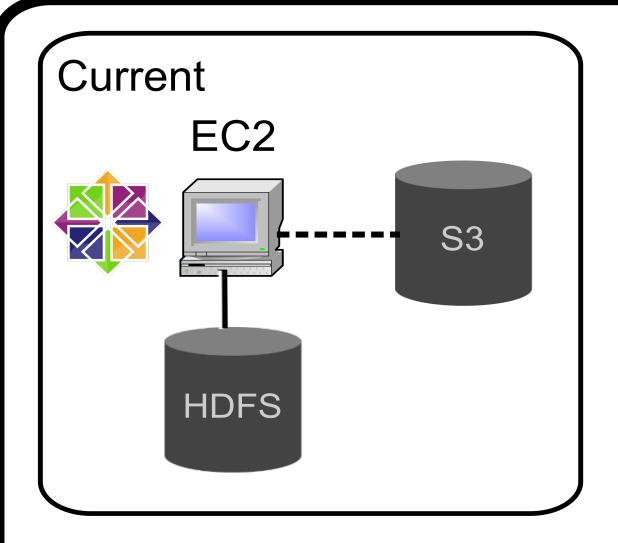


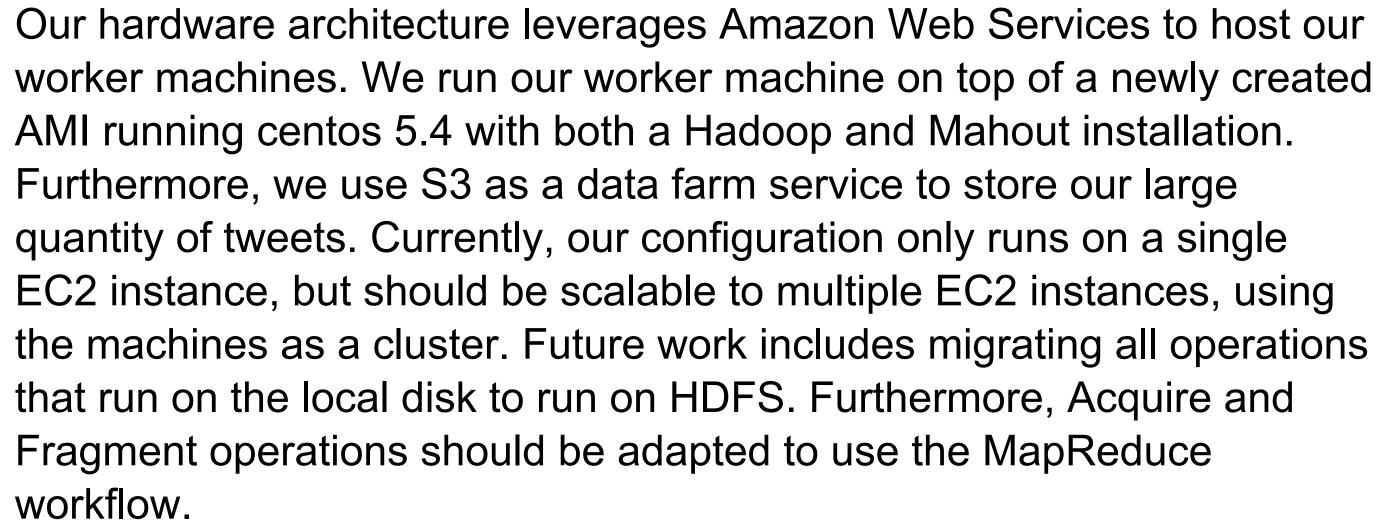












Ideal



Our project used a service called s3fs (S3 FileSystem) which allowed us to mount an S3 bucket and work with it as if it was on the local filesystem. But beware! The s3fs service did not scale well with many small files, and the overhead for each transaction rendered the service unusable at any non-trivial scale!

Results

In reviewing the results over the two timeslices we can see overlap in the hashtags and topics generated over an hour apart. It also served highlight the difficulties in foreign language detection and trivial content contained in tweets.

