W205.1 Group Project – Final Report
Data-Driven Linguistics

Brandon Shurick & Jared Maslin

Background & Use Case:

In considering a project topic, we wanted to hone in on a topic that no only had relevance to the team, but also to a broader audience of potential users in the future. Through conversation, one team member explained that he was currently taking lessons to learn Spanish, which is not his native language. This brought up a notion that many people experience in their lives, which is the difference between linguistics information and useful linguistics information.

This idea centers on the recurring theme in certain levels of education in which students constantly ask, "Why does this matter to us?" After all, learning can often be quite difficult, especially if the purpose of a specific topic is a mystery to students. The team also recalled one member's experience in taking foreign language courses in high school, in which a class took nearly two weeks' worth of class time in order to teach the sentence, "Donde esta la biblioteca", which in English translates to "Where is the library?" Now, for many students, learning the translation of the term that they've spent so much time on could make students wonder why it's worth their time to continue learning what they believe to be trivial terms or phrases. Right, wrong, or indifferent, this is a common reality amongst different levels of education. In order for teachers to get through to students about a given subject matter, there is almost a prerequisite notion of proving the information's worth prior to spending time on learning it.

With the above idea in mind, we set out to develop a tool for identifying current or relevant information in order to prioritize different points of focus in education. For example, a foreign languages instructor at a K-8 school in Missouri often has a borderline-herculean task: take a subject matter that can require decades to truly master, and condense that material into a single semester's worth of teaching. Toss in funding concerns and outdated textbooks, and we see a very common, yet troubling situation in language education today. What options are available for developing a language curriculum that is both effective and relevant to the students being taught? How can teachers prioritize and organize material in a manner that is more meaningful to their student base? These questions are drivers in our course project, which sets out to use public datasets to identify key topics and cultural norms through which to guide curriculum development in linguistics education today.

The solution detailed in this paper will include aspects of each of the "Three V's" of data: Volume, Variety, and Velocity. For volume, our choice of data sources will allow for the continued growth of our application. This is especially true for Twitter, being that it allows for a streaming application to ingest tweet activity at near real-time. For variety, we have chosen to use two distinct datasets, being the Twitter API and public Wikipedia traffic. Both datasets have unique aspects (i.e., JSON versus XML inputs, presence of Unicode characters, etc.) that will require special attention in order to properly clean and merge both datasets into a single one. And finally, we will handle high velocity in our dataset by keeping our streaming application running for the maximum time frame, as well as using Redis to execute real-time word counts. With this in mind, the team will confront the balancing of multiple datasets and multiple ingest and processing functions that can happen simultaneously. While ambitious, the team's solution reflects an appropriate level of attention to the needs of potential future users, as fitting their needs and allowing for subsequent use can be paramount in adding value and nurturing a sustainable solution long-term.

Scoping & Identifying Data Sources:

Determining the scope and the sources of data for this project, we must first recognize the shortage of resources across all levels of education in the United States. There can sometimes be a tendency to *throw money at a situation*, such as buying new computers in classrooms or purchasing software to enhance the interactive learning experience. Given that such a solution is likely outside the realm of possibility, we must design a solution that does not require significant funding or technological resources to use. This notion points to open source technology and public datasets, which fits both the spirit of the project, as well as the existing constraints in education.

In brainstorming available public datasets (or data that would contribute to a self-compiled set), one might consider the possibilities of utilizing web scraping to construct a dataset to fit our intended purpose. Unfortunately, this solution presented concerns not only in the formatting and consistency of the data, but also carries a wide array of legal and/or ethical conundrums in how the data is obtained and repurposed. So, with web scraping from independent sources or reference texts, what other options are available? How can we obtain or construct a large corpus of words or phrases to aid in linguistic education without imposing a great deal of ethical concerns?

Our proposed solution was to take advantage of publically available social media. Not only does the notion of publically available data (often through APIs or *Application Programming Interfaces*) remove a majority of ethical concerns, but it also removes any general costs associated with either purchasing the right to use private software or commercial datasets. Also, we note the difficulty in preparing a curriculum that is both current and relevant, from the perspective of the students. The more a student can relate to subject matter or can be enthusiastic about, the greater chance there will be for material to *click* in their minds.

Furthermore, with social media becoming a vital part of mainstream America and underlying cultural construct, social media data is as close to real-time as one could hope for in public data, and as the spoken word evolves over time, so does the dataset that is available for use. In order words, social media is often one of the first aspects of human life to show evidence of linguistic evolution or to recognize new social and cultural trends. A perfect example of this would be Twitter, whose public API allows for streaming access to “tweet” data and provides a lens into the way that speech is being used in real life scenarios. For these reasons, we felt that Twitter would be a perfect tool with which to compile a rich corpus of speech (via the written word).

With the choice of Twitter, the next step was to search for a complementary dataset, as we recognize the limitations of only selecting a single dataset for high-level analysis. One limitation of Twitter is that you are subject to erroneous claims, and often, poor grammar. On this premise, we searched for complementary datasets from those that were publically available and came across the Wikipedia traffic dataset, which is held in a public elastic block store (EBS) volume within Amazon Web Services. Like Twitter, Wikipedia updates and user traffic are often influenced by social or cultural norms, as well as the changing interest of users during that time, which provides a bit of a counter-balance to the higher-velocity Twitter data. So, with the public Twitter API and public Wikipedia data in mind, the team set out to develop an architecture fit to preprocess, store, and retrieve the combined dataset for analysis and use in furthering linguistic education.

Choice of Technology:

After defining the data sources of interest, the next step was to determine the proper tools for the mission. Given the sheer magnitude of the data in question, it was obvious to the team that locally storing or running any kind of analysis would have been unbearably slow, if even possible. Instead, the team decided that the most feasible option would be to utilize an AWS EC2 machine with large, magnetic disks and use Hadoop with Hive for batch processing, and Redis as a key-value store for storing simple word counts in real-time. After the project, the datasets will be archived in an object store like S3 for later use. The architecture is designed so that additional nodes could be added as needed when the data accumulates to a size that is no longer manageable within a single node.

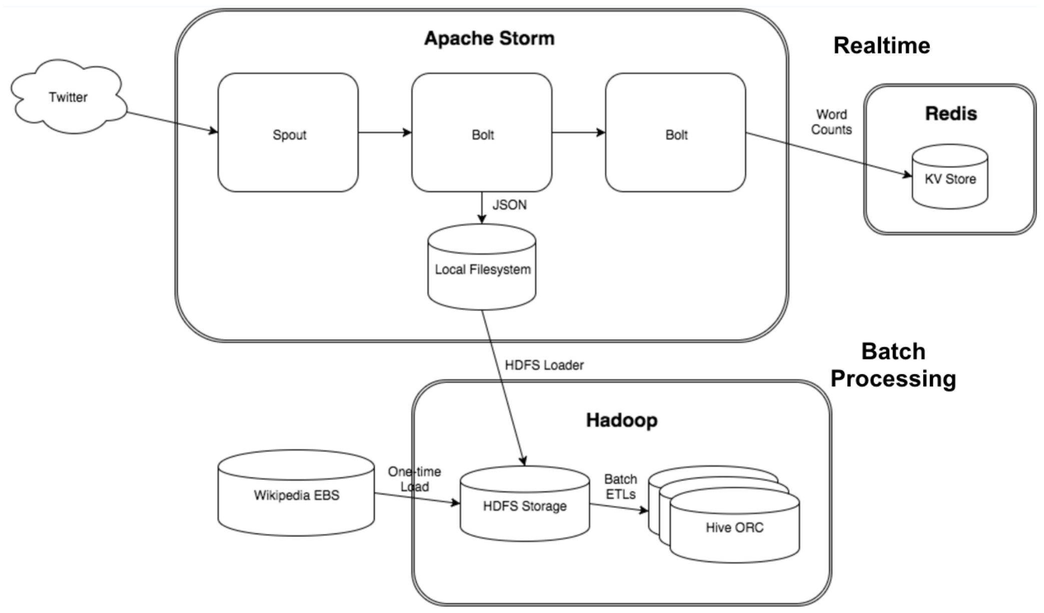
Design/Architecture:

Given the varying storage locations for the Twitter and Wikipedia datasets, we needed to take two distinct approaches to obtaining the data prior to combination. As mentioned above, Twitter has a public API that allows for broad access to a streaming application for capturing batches of tweets given user-defined criteria. To support this, the team developed scripts for a streaming Twitter application in order to capture a real-time stream of tweets. The primary focus, given the decision to focus on Spanish language education, was to set the "lang" attribute in our application to ingest only Spanish tweets.

Apache Storm is used to connect to twitter and filter tweets from the twitter "Firehose" API, and then the tweets are sent in JSON format to the local file system. The topology then parses out unneeded words, such as numbers or hashtags, and then sends word counts to a Redis server instance in order to enable real-time analysis of word choices as they occur.

We choose to run a bash script each hour that compiles all of the JSON files into a larger file, compresses it and then loads the compressed file into HDFS. This enables better compression and, since real-time processing is enabled via Redis, the batch processing jobs that run within Hadoop and Hive are not time sensitive. A number of ETLs were created that format the data into different segments, which are purposed for different tasks, such as user analysis or analysis of word choices within tweets.

Wikipedia data exists on a public Amazon EBS volume in XML format and is static, so a one-time connection and load from the EBS volume into HDFS was necessary. Data for both the compressed wikipedia XML and hourly compressed JSON files is then processed with Hive ETLs from the Hive external table abstraction into Hive ORC tables, each with it's own specific purpose, discussed further in the Entity-Relationship Organization section. ORC format is used because of the improved compression and data processing that the format provides. The Wikipedia data is processed using an XML serde found online (https://github.com/dvasilen/Hive-XML-SerDe/wiki/XML-data-sources) and data is loaded into ORC format also.



Entity-Relationship Organization:

Our Entity-Relationship Diagram was created with consideration for the how the solution could support expansion with both additional languages and additional datasets. The raw tweet data and raw Wikipedia data tables are created so that data for each language would be held in a distinct table, while the stg.tweets table and prd.wikipedia tables each have a language field which allows multiple languages to be compiled together into the same table and allows for filtering later on for a specific language. To add a new language, the developer simply needs to add another table for both Wikipedia and tweets and then write an ETL to extract the elements from raw data that fit into the staging and production datasets.

For Wikipedia data there is only one step required - load data from raw XML format into the production table called prd.wikipedia, along with the language of the data. One step is required because the data elements are extracted from the XML tags using an XML serde, which is publicly available. For twitter data, each tweet is stored in its original JSON format, and instead of modeling in Hive all of the elements within the complicated structure using a JSON serde, the choice was made to create a python mapper which would extract and parse only the necessary elements from each tweet and send them to the ORC-format stg.tweets table.

Next, data is loaded from stg.tweets into the prd.tweet_log, prd.tweet_users, and prd.tweet_words tables. The log table contains timestamp, retweet, and number of shares for each tweet along with a link to the users table. The users table contains the screen name, location (often empty or in bad format) and the user's inputted name. Since all of these fields are input by the user, they are quite limited in their usefulness. The tweet_words table is built to extract out the necessary elements for analysis that are scoped for the solution. Elements such as words and sentence structure are extracted in the same way for the Wikipedia data -- and any future data source as well -- so that the data can then be combined into a final table called prd.combined_words.

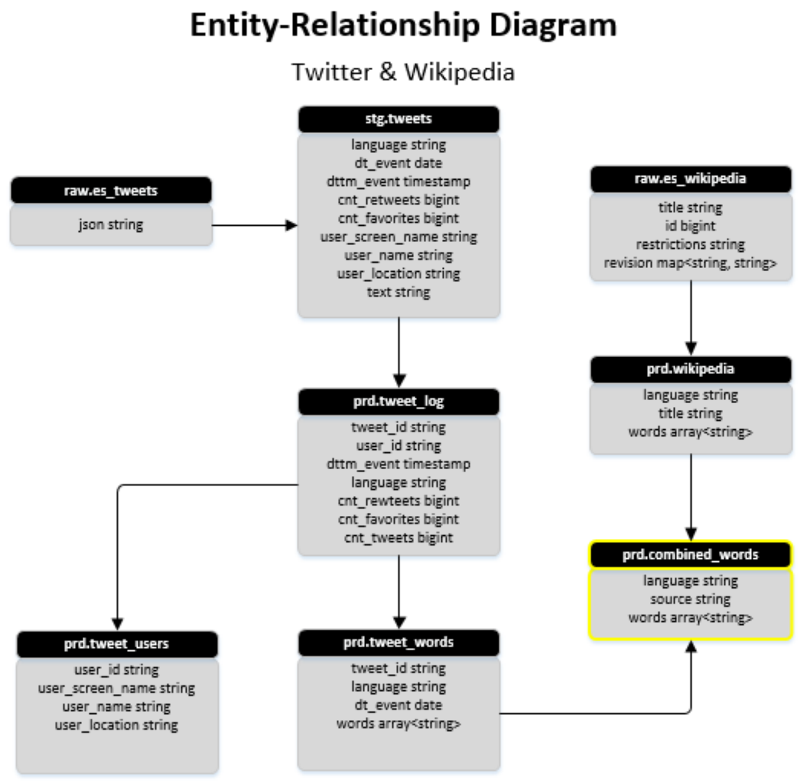


Figure B: Entity-Relationship Diagram showing steps to merging the final dataset

Obstacles and Successes:

Our team faced and overcame a few obstacles during our project. One of the biggest challenges was how to process a steady stream of Spanish-language tweets in an efficient fashion. We had a number of problems when we attempted to write each tweet to a single (growing) file. Because we chose to use magnetic disks in order to increase the storage capacity (and because Hadoop's design does not enable it to utilize the benefits of more-expensive SSD hardware), the write speed would often slow down and cause all tweet writing to halt. To mitigate these issues, we chose to write each tweet to a single file. Each file would have a name that contains the full timestamp down to milliseconds in order to prevent two tweet being written to the same file at the exact same time. This enabled us to scale up the Storm topology to have many processors writing at the same time, although in the end we chose to limit the number of bolt processes writing to JSON files to stay with only a m1.large instance.

Another big issue we had was parsing the file formats involved, namely XML and JSON. For JSON, a JSON-serde was attempted at first; however the tweet JSON format has nested elements, which aren't supported well by the serde, so instead of using the serde we created a custom python mapper which read in raw JSON and outputted the parsed columns we needed.

During the process of creating our custom mapper we ran into another, related issue, which was the processing of Unicode data. For the Spanish language, there are many unusual characters with accents and so data is usually stored in unicode, which is handled just fine by Hive and Hadoop; however, the data streamed to python needs to be handled appropriately, and this was not as straightforward as adding a normal 'unicode_literals' flag. Instead, the correct combination of string decode and encode (to UTF-8) functions were required, and this took some trial and error to finally resolve.

Parsing the Wikipedia XML was initially a problem; however, we found a XML processing serde online and once it was discovered that the XML document structure was fairly simple, the issue evolved into how to correctly parse out markdown elements from the Wikipedia structure. Our solution involved creating another custom python mapper job, which evaluated each word and throws out some that are bad and does some parsing of markdown formatting in order to let those words remain in the document.

Though our solutions to the above problems are imperfect, by using publicly-available serdes whenever possible and writing our mappers in python, we have enabled maximum flexibility for additional data sources and languages in the future.

Results and Future Enhancements:

The infrastructure, with all corresponding code/scripts to run the application, can be found on Github, in the following repository: https://github.com/bshur2008/w205project. You can also find copies of our presentation decks in the aforementioned repository, as references.

At conclusion, the application and system architecture (shown above) produced an incredibly rich dataset for future mining and analysis. In all, the team gathered for than 27,000,000 tweets from the Twitter API, as well as more than 787,000 Spanish Wikipedia documents. Again, our tool is intended to provide an approachable and cost-effective manner for educators to assess trends in speech and student interests in order to prioritize and organize their classroom curriculum in a manner that can be most easily received by students. For example, using word count functions (which Redis does in real-time, from our infrastructure) and looking at those counts over time can offer a great deal of insight into what is current and considered "relevant" to the student base. In many classrooms, an inability to relate to students with topics that interest them the most can make or break the success of the current educational paradigm.

There are many options through which to further develop our solutions. For example, in considering how to scale the application, the issue of storage efficiency and processing efficiency would be something to consider. As mentioned previously, using multiple nodes simultaneously can significantly escalate AWS expenses. For significant scaling through dataset expansion or more complex computation to occur, though, may be necessary costs depending upon the motivation and end goals in mind.

In considering potential enhancements, our application can very easily shift from one language to the next. While Spanish was the focus of our prototype, choosing to utilize our solution for a different language could be updated and run very minimal adjustment. On top of this, the model is also designed to be flexible in extending the model across other data sources. Since the application's primary function is to develop a robust corpus of current events and speech, one might add Facebook data, Google news data, or academic journals, depending upon the type of language being taught. Again, our solution is meant to be a teaching aid. Whether teaching an individual or an entire classroom, keeping an updated corpus of public speech and social media can pay big dividends in terms of compiling a relevant and engaging curriculum for the classroom.