# Introducing NoSQL Using Real-world Datasets from Twitter

Thyago Mota
motat@moravian.edu
Department of Mathematics and Computer Science
Moravian College
Bethlehem, PA, 18018, USA

## Abstract

*Due to the increasing popularity of NoSQL systems, there is an urgent need to introduce the topic in computer science and information systems undergraduate curriculum. These systems are characterized by a variety of database models and vendor implementations, each with its own specific query language, particularities, and application niches. For these reasons, it has become increasingly evident that teaching NoSQL requires a more creative, project-based learning approach; one that instigates students' curiosity and urges them to seek knowledge by themselves. This paper describes two NoSQL projects that use real-world datasets crafted from Twitter; they can be easily adapted and assigned to students taking an introductory course in database systems to help them learn NoSQL quickly with an engaging approach.*

**Keywords:** NoSQL, Twitter, Project-based Learning, and Self-directed Learning.

### 1. INTRODUCTION

It has been nearly a decade since NoSQL, as we know it today, began to take shape together with a whole generation of database systems. Driven by the necessity to be able to work with ever increasing amounts of unstructured data, NoSQL systems propose flexible, schema-less data models that apply abstractions different than the tabular relations used in traditional relational systems. Built with horizontal scalability in mind, NoSQL is better suited to support applications that deal with constantly changing and rapidly increasing data such as those found in healthcare and social media applications.

Due to the growing popularity of NoSQL systems seen in both industry and academia, there is an urgent need to introduce the topic in computer science and information systems undergraduate curriculum. As a liberal arts college, our computer science program has general education requirements, focusing on problem-solving skills, team work, and critical thinking. Similar to other programs across the nation, we regularly offer an elective course in database systems. Traditionally, this course emphasized the theoretical and practical concepts surrounding the relational model, SQL, and database programming. However, in efforts to adapt to the new challenges faced by database applications today, we made substantial changes to our database systems course over the past two years by reserving time at the end of the course to properly introduce students to NoSQL systems.

Different from SQL, which is backed by well-established relational model and industry standards, NoSQL brought a myriad of database models and vendor implementations, each with its own specific query language, particularities, and application niches. It soon became evident that teaching NoSQL would require a more creative, project-based learning approach that instigates students curiosity and urges them to seek knowledge by themselves.

To help achieve this new, project-based learning approach, we proposed assigning to students NoSQL projects using real-world datasets based on Twitter. Working with real-world datasets, as opposed to fabricated data, give students a sense that what they are doing in class is meaningful and has applications outside of the college setting. This paper describes two NoSQL projects that can be easily adapted and assigned to students taking an introductory course in database systems. Each of the projects is designed to use a different type of NoSQL flavor: one is based on a document-store, while the other uses a graph-based system. For these projects it is assumed that students demonstrate an intermediate (or higher) programming skill level.

As noted by [1], "... each NoSQL database has been developed in response to a specific problem and brings along with it, its own underlying concepts and language for interaction." Therefore, NoSQL has a steeper learning curve, taking considerably more time for students to understand how to apply the new model in real scenarios. We advocate that a project-based approach can help accelerate learning as it makes the material more relevant and engaging to the students.

Both projects presented in this paper use real-world datasets built from Twitter that were privately shared with our database system students. Open questions were left at the end of each assignment to encourage self-reflection and in-class discussion on how social media platforms like Twitter are being used today in different contexts.

## 2. METHODOLOGY

Education researchers soon realized that students are less likely to learn when bored or unengaged [2]. Project-based learning helps students achieve higher levels of understanding while working with their peers exploring real-world problems. To succeed it often requires multiple learning techniques such as research, trial-and-error, and logical reasoning. Students should be given the chance to experience what they learn in class by creating and testing new hypothesis on their own. With project-based learning students work on real-world activities that are similar to activities professionals engage in in the workplace.

Another key aspect of our methodology is the application of self-directed learning. We strongly believe that teaching and learning is a two-way street. A carefully designed course might fall short if the students are not able to make the right connections and learn. To achieve this end, students need to spend time studying on their own so they can reflect on their learning and create new connections in their brains.

Because of the fast pace in which new technologies are developed, hardware devices, programming tools, and paradigms used today will most likely be very different in a few short years; this is particularly true with database systems. Therefore, students are urged to learn how to self-educate and be prepared to face a future scenario where technologies will quickly be replaced or become obsolete. In particular, self-directed learning helps students to remain up-to-date with the new NoSQL developments (or any other related technology) as it continues to evolve.

Java was the programming language of choice to run the data collection and preprocessing for both projects. Even though it was discussed, students were not required to understand how the data was collected to complete the assignments. Twitter4J API (version 4.0) [3] was used to develop the data crawler programs that read and store real-time Twitter content.

Unfortunately, due to Twitter's Developer Agreement and Policy [4], datasets built using Twitter's API cannot be shared with the general public to respect user privacy. However, instructors interested in creating similar activities can generate their own datasets adapting our Twitter code [5].

## 3. 2015 SPANISH ELECTIONS PROJECT

For this project students were granted access to a MongoDB database containing tweets originated by four official Twitter accounts associated with major political parties and coalitions in Spain (as of 2015):

- `@PPopular` (*Partido Popular*),
- `@PSOE` (Partido Socialista Obrero Español),
- `@ahorapodemos` (*Podemos*), and
- `@CuidadanosCs` (*Ciudadanos*).

The database comprises tweets sent between October 2015 and January 2016 by the above accounts during a period that spans the Spanish general elections held on December 20, 2015. As a result of this election, the governing conservative *Partido Popular* failed to renew the absolute majority and lost almost one-fifth of its electoral support [6], triggering new elections the following year.

The goal of this project was to have students use MongoDB's query language[1] to investigate how the different political accounts used Twitter to influence Spanish voters for the 2015 general election. Before starting work on this project, students were introduced to the document data model, semi-structured representation of data using JSON, and MongoDB fundamentals which included how to remotely connect to a database and how to run simple queries through the shell.

Since most of the queries assigned for this project required students to perform grouping operations to summarize data, students were taught the fundamentals of MongoDB's aggregation framework using *pipeline stages* (essentially data transformation operations).

To encourage self-directed learning, students were left to figure out on their own which pipeline stages were best suited to answer a particular query. From our experience, NoSQL pipeline aggregation is much easier for students to grasp when compared to other programming models such as map-reduce. This is because the operations in a pipeline are broken into sequential steps that can be incrementally tested, which in turn facilitates a step-by-step approach to answer the queries. In addition, MongoDB's aggregation framework offers many ready-to-use pipeline operators which require less code to be written compared to map-reduce. Below is a list of all queries that the students had to answer for this project:

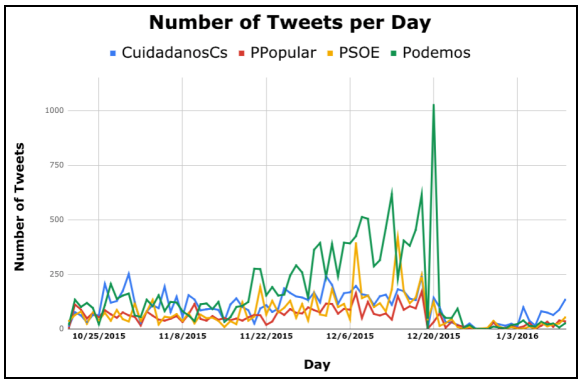
1. total number of tweets;
2. screen names of all user accounts;
3. text of all tweets sent by a given user;
4. date of the oldest and latest tweet;
5. number of tweets per user in descending order of number of tweets;
6. number of tweets per user each day in chronological order followed by the users names in alphabetical order;
7. frequency of words used in all tweets texts per user in alphabetical order of their names.

The first two queries are straightforward and can be answered using MongoDB collection methods `count` and `distinct`, respectively. Query #3 required students to use a variable to define the user's name and the `find` method with *query* and *projection* parameters. The remaining queries were written to be answered using the `aggregate` method combining different pipeline stages, such as `match`, `project`, `unwind`, `group`, `sort`, and `out`.

Queries #6 and #7 required students to save the query results in a temporary collection to facilitate exporting the results into a CSV file for further processing. For example, based on the results obtained from Query #6, students were asked to create a timeline chart of the number of tweets per day sent by each of the users. Figure 1 illustrates the expected chart.

[1] MongoDB's version 3.6 was used at the time.

Figure 1: Timeline of Number of Tweets per Day (best seen in color).




Query #7 is the most complex query in this project because it required students to learn how to extract the words from each tweet text. Although not performed perfectly, this task can be easily implemented using MongoDB's `split` operator which returns an array with all of the words from the tweet. The problem with this approach is that words near a punctuation mark (or any other special character) are mistakenly classified as distinct words (e.g., "*país*" and "*país.*"). To avoid this potential problem, we preprocessed the tweets by removing all punctuation and special characters prior sharing the database with the students.

Also for Query #7, the array of words returned by `split` had to be later deconstructed using the `unwind` pipeline stage. This step guaranteed that each array element is saved in its own document, preparing the data for the next phase of the pipeline. Continuing the sequence of data processing operations, a `group` stage had to be added to the pipeline so the number of occurrences for each word per user could be counted properly.

As their last task for this assignment, students had to further process the CSV file generated from Query #7 to remove unrecognized words, short words, prepositions, etc. The final goal was to have students generate a word cloud from the words used by each of the political accounts when posting on Twitter. To illustrate what we were expecting, Figures 2 and 3 show the word clouds for two of the accounts.

Figure 2: Word Cloud for `@PPopular` .


Figure 3: Word Cloud for `@ahorapodemos`.


At the end of the project, students were asked to write a few sentences describing how they think the different Spanish political parties tried to use Twitter to influence voters in the 2015 general elections. Some students pointed out how *Podemos* used Twitter considerably more than the other political parties and that after the December 20th election day there was a significant drop in the number of messages from all of the accounts. Other students noted that *Podemos* used the word "cambio" quite often as they advocated for political change in Spain. Finally, a student described how *Podemos* predominantly used the word "*país*" ("Country" in English) instead of "España" (Spain). One possible explanation was that *Podemos* was purposely trying to be more respectful to voters who may be more sympathetic to nationalistic movements in the country.

## 4. GUN CONTROL PROJECT

The debate over regulations on civilian ownership of firearms has motivated intense discussions in countries throughout the world. In particular, this debate has grown fiercer over the past several years to become one of the most divisive topics in America today. And just like many other controversial issues, discussions over firearm regulations have been taken to social media outlets, Twitter included.

Beginning in the summer of 2018 we have been collecting the public postings and profiles of Twitter users active in the US gun control debate. Based on data collected, we selected a representative sample of 1K user profiles (including their timeline) and shared it with our students taking database systems. At this time, students were given the data in raw JSON format. Using the data file, we asked our students to build a graph database that illustrates the "retweet relationship" in the gun control debate. In this graph nodes represent Twitter users and directed edges (the connections between the nodes) were created whenever one user retweeted another user. To build the graph and perform the subsequent analysis students were required to use Neo4J[2].

Neo4J is a graph database management system based on Java that was first made available in 2010. Data in Neo4J is represented as either nodes or relationships (connections between nodes). Named values (called properties in Neo4J) and labels can be associated with both nodes and relationships. Labels are used to define different types of nodes (or relationships) in a graph. Because Neo4J is schema-free, similar nodes (or relationships) can have different set of properties (or labels attached). Neo4J defines its own declarative query language called Cypher. Common graph algorithms are made available in Neo4J as library-exposed Cypher procedures.

Before working on this project students were taught the basic theory behind graphs, in addition to the specifics of Neo4J's implementation. For example, students learned how to create a database and how to run Cypher statements through Neo4J Browser. To stimulate self-directed learning, students were encouraged to read Neo4J's online documentation and to learn on their own how to express database operations in Cypher.

In the first part of their assignment, students wrote the code to import the shared JSON file that contained the sample of user profiles (and their respective timelines) into Neo4J. The basic data structure of the JSON input file was described to the students as shown on Table 1.

Table 1: Structure of JSON Input File.

```
<User> ::=
{
      screen_name: <String>,
      timeline: [ <Tweet>, <Tweet>, … ],
}

<Tweet> ::=
{
      entities: {
            hashtags: [ <Hashtag>, <Hashtag>, … ]
      },
      retweeted_status: {
            user: <User>
      }
}

<Hashtag> ::=
{
      text: <String>
}
```

Students were directed to Cypher's `LOAD CSV` statement to learn how to import data into the graph database. The `LOAD CSV` statement in Cypher triggers an iterative process that reads the lines of the input file. At each iteration the columns of each line, as they are read, can be individually referenced and used as parameters for other Cypher statements, such as `MERGE` or `CREATE`. For this particular example, the `MERGE` statement was the prefered method to create nodes because it avoids creating the same node more than once. On the other hand, the `CREATE` statement worked better for the "retweet relationship" because one user might retweet a particular user more than once.

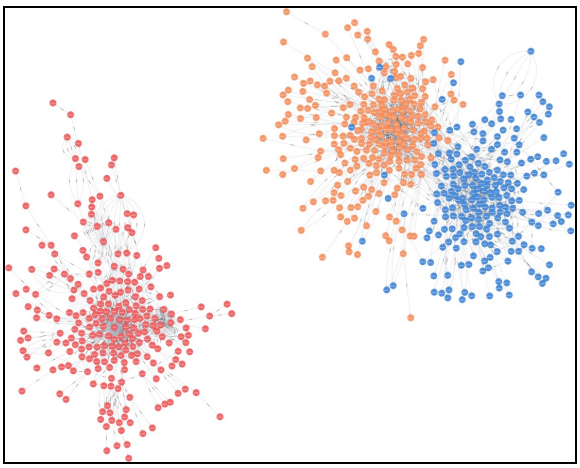
[2] Node4J version 3.2 was used at the time.

As previously mentioned, students had to write code to parse the input JSON file and generate a CSV file ready to be loaded into a new graph database representing the "retweet relationships." The expected CSV file only needed two columns: the first to represent the "retweeting user" (i.e., the user that retweeted), and the second to represent the "retweeted user." For example, if user `A` retweets a gun control tweet[3] originally sent by user `B` there should be a new relationship labeled "retweets" from `A` to `B` indicating that `A` retweeted `B`. To create this relationship a new line containing "`A,B`" should be added into the CSV file.

After importing the CSV file and generating the graph, students were asked to apply a community detection algorithm to identify groups of users in the graph that were highly interconnected. One way to detect communities in a graph is to look for partitions that have high modularity scores. Modularity is a scale value between -1 and 1 that measures the density of edges inside communities to edges outside communities. In other words, modularity is higher when the number of edges that fall within the partition is significantly higher than what would be expected if edges were distributed at random [7].

The Louvain algorithm for community detection [8] is based on the modularity concept that uses a two step, greedy optimization approach. First, it looks for small communities by optimizing modularity locally. Next, the algorithm repeats itself by aggregating small communities into larger ones by looking at each small community as a single node. Using Neo4J Louvain algorithm implementation, students were tasked to generate a visual representation of the "retweet relationship" graph using distinct colors to represent different group of users. The result of this graph visualization task is illustrated in Figure 4 and depicts the three largest communities[4] detected by the Louvain algorithm. The community in red (on the left-side of the figure) represents users that are in favor of gun rights, while the other two communities are related to users that are pro gun control.

Figure 4: Retweet Relationship Graph with Communities (best seen in color).



What is the most influential user in each of the three detected communities? To answer this question students were required to compute the degree (number of edges) of each node (using a Neo4J appropriate method) and display the user that had the highest degree in each community. The results are shown below in descending order by the number of users:

- Orange community (266 users): `@AMarch4OurLives` (retweeted 192x);
- Red community (244 users): `@NRA` (retweeted 357x); and
- Blue community (223 users): `@Everytown` (retweeted 38x).

Referring back to the initial questions left open at the end of class assignments concerning social media and its use as a tool for influencing political beliefs or supporting controversial issues, we asked our students to propose other ways to work with the data they had. This activity was conducted in a class setting where students initially chatted in small groups followed by a class discussion where they shared their opinions and ways of reasoning. Some students suggested plotting the location of the users on the US map to see how they are geographically dispersed. Other students

[3] Students were given a list of gun control hashtags to be used to identify whether a tweet was gun control related or not.
[4] Defined as groups with at least 100 users.

considered looking more closely at links embedded in messages to understand how users in the gun control debate are influenced by news sources.

## 5. RELATED WORK

Mohan [1] highlighted the importance of adding the NoSQL paradigm to undergraduate curriculum in computer science. As stated by the author, "... students need to be exposed to multiple NoSQL paradigms and be provided with the opportunity to practice and reflect on their work." The paper describes a four-tier learning model to teach a course on modern database paradigms. Fowler et al. [9] showcase a project-based, teaching case that introduces NoSQL using a document-based NoSQL system in a traditional database system course. In their project proposal, students were asked to design a database system for a private social media website. In another study, Lee and Balmuri [10] discuss the advantages and disadvantages of using and teaching NoSQL databases. They also compare the features of commercial and open source NoSQL databases. Taking a different approach, the authors in [11] propose in-class activities to engage students in hands-on experience with SQL in the context of big data systems. To illustrate, the paper describes the use of HiveQL and SlamData to run SQL-like statements on MongoDB and Apache Hive, respectively.

## 6. CONCLUSION

Incorporating NoSQL into the undergraduate curriculum is challenging due to the variety of database models that are available. Particularly so since each model is optimized for certain types of problems. This paper describes two projects that can be assigned to students in a typical database system course, adapted to cover not only the traditional curriculum found in most undergraduate programs, but also the basics of NoSQL systems. The projects are based on real-world datasets crafted using Twitter, a popular social media network. Using a project-based approach with open questions requiring research and afterthought, the proposed approach stimulates students engagement and interest while encouraging self-directed learning. The projects presented can be easily adapted (with slight modifications) allowing their continuous use by instructors interested in incorporating NoSQL in their database courses.

## 7. REFERENCES

[1] Mohan, S. *Teaching NoSQL Databases to Undergraduate Students: A Novel Approach*. Proceedings of the ACM Technical Symposium on CS Education (SIGCSE). Maryland, MD. 2018.
[2] Krajcik, J. and Blumenfeld, P. *The Cambridge Handbook of the Learning Sciences.* Chapter 19. Cambridge University Press. 2006.
[3] *Twitter4J API*. URL: http://twitter4j.org/en. Accessed on January 2019.
[4] *Twitter's Developer Agreement and Policy*. URL:https://developer.twitter.com/en/developer-terms/agreement-and-policy.html. Accessed on January 2019.
[5] Mota. T. *NoSQL Twitter Projects Repository*. URL:https://github.com/thyagomota/nosql-twitter-prjs. Accessed on January 2019.
[6] The Washington Post. *After Spain's startling election, here are the five ways it can form a government*. December 22, 2015.
[7] Newman, M. *Networks: An Introduction*. Oxford University Press. 2010.
[8] Blondel, V. et al. *Fast Unfolding of Communities in Large Networks*. Computing Research Repository (CoRR). 2008.
[9] Fowler, B., Godin, J., and Geddy, M. *Teaching Case: Introduction to NoSQL in a Traditional Database Course*. Journal of Information Systems Education. Vol. 27(2). Spring 2016.
[10] Lee, F. and Balmuri, S. *Teaching NoSQL Database*. In Proceedings of the Information Systems Education Conference. 2018.
[11] Silva, Y., Almeida, I., and Queiroz, M. *SQL: From Traditional Databases to Big Data*. In Proceedings of the ACM Technical Symposium on CS Education (SIGCSE). Memphis, TN. 2016.