Sentiment Analysis

MySQL vs mongoDB

mongoDB is a NoSQL document-oriented database, removing relational structure in favor of datasets with key-value store behavior.

mongo stores data in JSON-like objects called BSON which have fully dynamic schemas.

Why use mongoDB?

mongoDB allows the programmer:

- ➔ more flexibility (no rigid schemas)
- ➔ the ability to shard data for distributed systems with more ease
  - ◆ (Allows greater scalability and availability)
- ➔ to avoid using ORM layers, as JSON maps naturally as is

Sentiment Analysis: mongoDB

- ➔ Building the dataset completed as in the assignment instructions
  - ◆ no code required beyond the mongo interpreter
- ➔ Used Ruby's mongo gem to connect to the mongo instance
- ➔ Loaded the positive and negative words into in-memory Ruby hashes
- ➔ Determined sentiment by counting the words that appeared in each respective hash, and creating an output JSON file based on the id, review text and output sentiment
- ➔ **~37 lines of Ruby code**

Sentiment Analysis: MySQL

- ➔ As per instructions, only schema and query implemented
- ➔ Code would be required to build the database from the JSON and txt files provided
- ➔ Schema very simple:
  - ◆ id and review content for each review
  - ◆ id foreign key, the word, and its count for each word
  - ◆ sentiment value and word for each sentiment word (from * words.txt)
- ➔ After this, **all of the code can be performed in SQL**
- ➔ **~3 lines for the SQL query**

MySQL Query

```
SELECT r.review, r.id, CASE WHEN SUM(w.sentiment * s.`count`) >= 0
    THEN 'positive' ELSE 'negative' END AS sentiment
    FROM unlabel_review r, unlabel_review_after_splitting s, words w
    WHERE w.word = s.word AND r.id = s.id
```

- ➔ Join each table and grab words that have sentiment values
  - ◆ +1 for positive, -1 for negative
- ➔ Sum the products of the sentiment values by the counts to determine the sentiment value of the review
- ➔ Use CASE WHEN...THEN...ELSE...END to return the string directly, and avoid the need for more code

MySQL vs MongoDB

- ➔ Given the input reviews in .json format, building the database would be far simpler to build in mongo
- ➔ mongo does not need to join tables
  - ◆ for sufficiently large reviews, mongo would be *much* faster
  - ◆ sql schema includes an index on the foreign key to mitigate this
- ➔ Given an pre-built database, however, SQL is the clear winner
  - ◆ Fewer total lines needed for query; query handles everything
  - ◆ Data format lends itself easily to relational structure