Optimizing User-Posts on Stack Overflow

Gabriel Vigil, Kyle Lagerberg, Reece Sharp

CS 455: Introduction to Distributed Systems

Computer Science Department, Colorado State University

Background information

Stack Overflow is an important tool

- New and old programmers use, ask, and read questions in every day development
- Leverage against potential issues a programmer encounters.
- It’s Important to know how to use this tool effectively

Problem characterization

Using a dataset of 10% of posts skimmed from 2008 - 2016 to quantify a “good” post

3.5 GB total dataset of Questions, and Answers

2 million answers, 1.2 million questions

- Composed of primarily post identification values, text body, relevant timestamps, and scores

Methodology (1/3)

Programs start by pipelining raw information through a standardization process.

- Parses data from a messy multiline CSV
- Use of Regex and Splits to parse individual words or sentences from markdown paragraphs
- Turns each body into individual standardized words for later jobs

Methodology (2/3)

Using Map and Reduce on RDDs to do simple problem simplizations

- Use of SQL on datasets to select specific data
- Word counts with map and reduce
- Word Averages with RDDs using tuples and chaining MapReduce jobs
- Filter and orderby to rearrange data efficiently

Methodology (3/3)

Analyze the bodies of user posts using sentiment analysis

- Gives an overall rating of the post: “positive” vs. “negative”
- Utilizes a 3rd party library built on Spark ML, called Spark NLP (Natural Language Processing), using Python
- Pipelines raw information through a pre-trained model for output

Performance benchmarks (1/3)

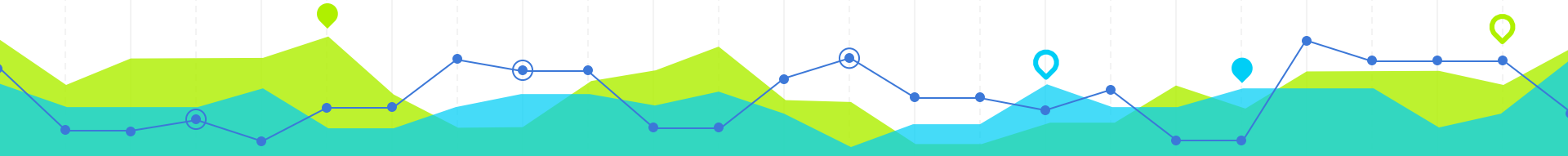
Computation done by spark clusters over hadoop file systems

- 12 to 15 nodes depending on user
- 4gb of ram on each machine to handle the full dataset if need be
- 2 workers on each node

Performance benchmarks (2/3)

Word counts and score analysis

- Created counts to find most common words for questions and answers
- Averaged scores by word to find the most liked words
- Paired words with score thresholds to count words by successful posts
- Started models for successful posts



Performance benchmarks (3/3)

ML and NLP implemented and working, feeding in the body data to result in a 1 dimensional list of either “positive” or “negative” relating to each post

- Spark NLP instead required Spark 2.4.x to be compiled
- However Spark 2.4.x no longer supports PySpark on Clusters, requiring Scala, leading to a complete rewrite

Insights and conclusion

We were able to implement machine learning and natural language processing to rate the text sentiment of each post

- Output from this allows for better analysis of dataset in the future

We will want to use the outputs of the NLP in Spark to acquire different statistics from the data

- i.e. key words, word counts, and scores of post with the best overall sentiment