Comparison between a sequential and a distributed version of a Twitter sentiment analysis tool

Emilio Cecchini

emilio.cecchini@stud.unifi.it

Abstract

In this paper I will analyze the advantages in speed and efficiency obtainable in implementing a Twitter sentiment analysis tool with Hadoop. The main goal of this study is not to build a perfect classifier, but it is instead to show the speedup that you can get using a distributed algorithm versus a sequential version. To build the sentiment classifier the library LingPipe is used, so the details of that classifier will be ignored. The distributed version is written in Hadoop with the MapReduce framework.

1. Introduction

Sentiment analysis involves classifying opinions in text into categories like *positive* or *negative* often with an implicit category of *neutral*. In this project, for simplicity, we will ignore the *neutral* category. All the entire part of building the classifier and getting results out of it is done with the LingPipe library, that is a Java library to perform machine learning tasks such as text classification and natural language processing.

First of all there is a *training* phase, where the model is trained to recognize a tweet as negative or positive. This phase is only implemented in a sequential version. Once the model is built and trained, there is the *classification* phase, where many unclassified tweets are passed as input to the model. This phase is implemented in two different versions: a sequential version and a distributed version in Hadoop with the MapReduce framework.

2. Training

The training process is done with a dataset containing tweets already classified. The model reads the tweet texts and their sentiment and try to learn how to perform the classification.

The process of training the model must be computed in a sequential way because every tweet and its associated sentiment must be processed by the same unique model. A parallel version could be hypothetical possible but in that case reading and processing the tweet must be encapsulated in a critical section and therefore all the advantages of parallelization would be lost.

Thanks to the LingPipe library, the algorithm of training the model is straightforward:

Algorithm 1 Training the model

---
m = initializeModel()
for each tweet t **do**
    handle(m, t)
end for
saveModel(m)

The function `saveModel()` save the model in a file so that it can be loaded later by a classifier to perform a classification.

3. Classification

The classification phase can be performed in a distributed way, so I have implemented a sequential version and a distributed version in Hadoop.

3.1. Sequential version

With the LingPipe library it is very easy to implement a sequential algorithm that classify tweets.

The first thing to do is to load the trained model. This process may take a long time if the model has been trained with a very large dataset. Loading the model can not be parallelized, so it is suggested to keep the file as small as possible to prevent it to become the bottleneck of the program.

After loading the model, the program iterates over the tweet texts and uses the classifier to predict their sentiment. During the iteration, the program keep tracks of the number of negative tweets as well as the number of positive tweets with two counters. The output of this algorithm are the two counters from which the general sentiment can be derived.

Algorithm 2 Sequential classification

$m = \text{loadModel}()$
$p = 0$
$n = 0$
for each tweet t **do**
    $s = \text{classify}(m, t)$
    if isPositiveSentiment(s) **then**
        $p = p + 1$
    else
        $n = n + 1$
    end if
end for
return p, n

3.2. Distributed version