Assignment 3: Sentiment Analysis on Amazon Reviews

Arash Vahdat

CMPT 733

Fall 2015

Readings The following readings are highly recommended before/while doing this assignment:

- Sentiment analysis survey:
  - *Opinion Mining and Sentiment Analysis*, Bo Pang and Lillian Lee, Foundations and trends in information retrieval 2008.
  - *Opinion Mining and Sentiment Analysis*, Bing Liu, Web Data Mining, 2011.
- Sentiment analysis tutorial at `https://www.kaggle.com/c/word2vec-nlp-tutorial`
- Spark JSON I/O: Karau et al. Ch. 5.
- Spark TF-IDF: Spark documentation and Ryza et al. Page 105.
- Spark word2vec: Google project page `https://code.google.com/p/word2vec/`
- Spark K-means: Spark documentation and Ryza et al. Page 82.
- Spark linear regression: Spark documentation.

Provided Data We will use 1.2 million reviews extracted from Amazon's website [1]. The JSON file containing the dataset is available at the following address:

`/cs/vml2/avahdat/CMPT733_Data_Sets/Assignment3`

1 Introduction

The goal of this assignment is to perform sentiment analysis on the Amazon reviews. According to Wikipedia, "Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document."

[1] http://snap.stanford.edu/data/web-Amazon.html

For sentiment analysis on Amazon reviews, we will examine two different text representations. First, we will consider the *Bag-of-Words representation* that describes a text (in our case a single review) using a histogram of word frequencies. There are different approaches for Bag-of-Words representations, we will consider the "term frequency-inverse document frequency" (TF-IDF). In TF-IDF, a feature vector is created for a given review by counting the frequency of words in the review. These word frequencies are normalized based on how common (or frequent) each word is. Second, we will examine the *distributed word vector representation* (word2vec) where a word is mapped to a feature vector such that words appearing in a similar context have a similar representation.

This assignment is inspired by the recent Kaggle challenge "Bag of Words Meets Bags of Popcorn" available at: `https://www.kaggle.com/c/word2vec-nlp-tutorial`. This Kaggle challenge was designed as a sentiment analysis tutorial and contains valuable implementation details. The challenge uses Python libraries such as Scikit-learn, however, for designing more scalable solutions, our implementation will use Spark framework. You are highly encouraged to read and understand the challenge tutorial as you are answering the assignment and designing your own solutions.

We will work with the Amazon reviews dataset available at `http://snap.stanford.edu/data/web-Amazon.html`. You are provided with a portion of this dataset which includes 1.2 million reviews on pet supplies. The data directory provided with this assignment contains two JSON files: 1) `reviews_Pet_Supplies_p1.json` contains exactly 120,000 reviews used for training and test; 2) `reviews_Pet_Supplies_p2.json` contains more than 1 million reviews that will be used for training word2vec feature representation. Each text review can be accessed under the "reviewText" field and is associated with a rating (ranging from 1 to 5) under the "overall" field of the JSON files.

Goal: Given a review, we are interested in predicting a user's attitude. A subset of the dataset will be used to train a "regression" model to predict user ratings. Note that a higher rating is typically associated with a positive sentiment, therefore, the same technique can be used to "classify" positive vs. negative sentiments.

Acknowledgement We would like to thank Julian McAuley and Jure Leskovec for gathering and sharing this valuable dataset with the research community. The data used in this assignment was originally collected in association with the following publication: *J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys, 2013.*. If you are interested in using this dataset in projects other than this assignment please email the authors.

2 TF-IDF Representation

Question 1: (10 marks) Write a Spark script that creates TF-IDF representation for each review in the `reviews_Pet_Supplies_p1.json` file. Your script should follow these steps:

1. Write a function that accepts a string review and:
    - replaces all punctuation with spaces,
    - converts the input string to lower case,
    - splits the string to words
    - and removes stop-words (see below).

2. Load the JSON file as a temporary Hive table.
3. Use the above function to convert each reviewText to a "cleaned" list of words.
4. Compute term frequency and inverse document frequency (TF-IDF) for each review to find its representation using the following word list: `https://spark.apache.org/docs/1.2.0/mllib-feature-extraction.html`
5. Split your data into train and test sets. Use reviews submitted before 2014 as training and reviews submitted in 2014 as test samples.
6. Store your train and test features as well as their corresponding rating scores.

Implementation Notes

- You can use either Spark 1.3 available on the rcg-hadoop server or Spark 1.5 installed on your machine. If you are using the lab machines, in order to add Spark binaries to your path, run the following command in terminal:

  ```
  setenv PATH "$PATH":/Users/avahdat/cmpt733/spark-1.5.0-bin-hadoop2.6/bin
  ```

  Please note that running this assignment on a local machine with multiple cores will be significantly faster than running it on the cluster.

- You may need to combine two RDDs containing review feature vectors and overall ratings. You can use zipWithIndex() to index elements in an RDD and join them on this index.

- To remove stop words, you can use the NLTK library English stop words:

  ```
  import nltk
  from nltk.corpus import stop-words
  # path to the nltk data directory.
  nltk.data.path.append("/cs/vml2/avahdat/CMPT733_Data_Sets/Assignment3/nltk_data")
  stop_words = set(stop-words.words("english"))
  ```

  If you are working on a lab machine or a personal computer, you can download the `nltk_data` directory to your machine. You will need to update the path to the library in the above code.

3 Train a Linear Regression Model in Spark

Question 2: (5 marks) Write a script that reads the training and test data created in question 1 to train and tests a linear regression model in Spark. You can use LinearRegressionWithSGD() in Spark without any regularization (default). Use the Root Mean Squared Error(RMSE) as your error measurement. Set the number of itterations to a large value and use cross-validation to choose a step-size. Report the best RMSE achieved and the corresponding step-size.

One approach to improve the accuracy of a histogram-based representation is to normalize the feature vectors. Review your script from the previous section and normalize your feature vector such that the L2 norm of each instance equals 1 using Spark's feature normalization methods `pyspark.mllib.featur.Normalizer`. Train and test your model again using the normalized features. Report your RMSE error. Are there any improvements?

Question 3: (5 marks) Think about the drawbacks of a bag-of-words representation. What are the main issues with this representation in your opinion? How can you improve your bag-of-words representation ?

4 Distributed Word Vector Representation (word2vec)

Paraphrasing Spark documentation, "word2vec computes a distributed vector representation of words. The main advantage of the distributed representations is that similar words are close in the vector space, which makes generalization to novel patterns easier and model estimation more robust. Distributed vector representation was shown to be useful in many natural language processing applications such as named entity recognition, disambiguation, parsing, tagging and machine translation."

A nice characteristic of the word vectors is that they capture many linguistic regularities. For example vector operations such as vector('Paris') - vector('France') + vector('Italy') result in a vector that is very close to vector('Rome'). Or vector('king') - vector('man') + vector('woman') is close to vector('queen').

One of the main advantages of word2vec representation is that it learns rich linguistic relations in an unsupervised framework. For example, consider training a sentiment model using regression models. We may need to have a large dataset of reviews and ratings to be able to train our regression model. But, for training word2vec representation, we can use any large available corpus. Internet is full of textual data. We can simply use all the data on the internet to train word2vec models.

Question 4: (5 marks) Write a Spark Python script that trains a word2vec model using the larger corpus available in the `reviews_Pet_Supplies_p2.json` file. Your script should store the trained word2vec model in the file system in order to re-use it later. Answer the following questions:

1. Saving the word2vec model trained in Spark's Python API is not trivial. Propose a simple trick to save this model. *Hint:* Remember word2vec basically trains a representation for each word and we have a limited number of unique words in our corpus.
2. Find similar words to some words of your choice. For example try words like 'dog' or 'happy'. List the similar words that you get using your word2vec model. Do the retrieved words make sense? Be creative and try other words.

Note: you can use the same function that you have used in Question 1 to clean up your reviews. However, when training the word2vec model it is better not to remove stop-words.

Question 5: (5 marks) As we discussed earlier, word2vec representation has the advantage of keeping words similarity in the feature space. One simple way to extract groups of similar words is to cluster them. Extend your script from question 4 such that at it performs the following:

1. Extract unique review words in the `reviews_Pet_Supplies_p2.json` dataset.
2. Use your word2vec model trained in the previous section to map these words to their corresponding feature vector.
3. Use Spark K-means clustering model to cluster word2vec representation of words. Set the number of clusters to 2,000.
4. Store you clustering model. Note that instead of storing the cluster centers you should store the cluster index for each unique word in your corpus. This will significantly improve your look-up speed when you are finding the cluster index for a large set of reviews.

After training your clustering model, explore some clusters and report the words that were assigned to them. Do these words look similar to each other?

5 Representing Reviews using Average word2vec Features

Question 6: (10 marks) Write a simple Spark script that extracts word2vec representations for each word of a Review. Use the word2vec you have trained in the previous section. Then, represent each review using the average vector of word features. Similar to question 1, compute the average word2vec features on all available reviews in the `reviews_Pet_Supplies_p1.json` file and split these reviews to a train and test set.

By default, Spark's implementation of word2vec represents each word using a 100-dimensional vector. Therefore, each review will be represented using a 100 dimensional vector. Use your linear regression training script from question 2 to train a regression model using the average word2vec features. Test the performance of your regression model on a test set. What is the best RMSE obtained using these features? Do you notice any improvements?

6 Representing Review Text using word2vec Bag-of-Words Features

In Sec.4, we created clusters of words in the word2vec feature space. A simple way of representing a review is to use a bag-of-words representation created on top of word2vec features. In this approach, instead of computing raw word frequencies, we will compute cluster frequencies, i.e. the number of times a cluster was assigned to the words in a review.

Question 7: (10 marks) Write a script in Spark that finds the closest cluster index for each word in a review using the look-up table created in the Sec.4. Then, represent each review using a histogram of cluster indices, i.e., the number of times a cluster was assigned to the words in a review. Your histogram should be (L1-norm) normalized meaning that sum of the values in your histogram should add up to one.

If you have 2,000 clusters, your histogram should be 2,000 dimensional. On average, the Amazon review dataset contains 70 words per review. So, the maximum number of non-zero bins in the bag-of-words histogram will be 70 on average. It is very important to store your word2vec bag-of-words in a sparse format. Learn how to create sparse vectors in Spark.

Use 2,000 clusters and form word2vec bag of words for the reviews available in the `reviews_Pet_Supplies_-p1.json` file. Follow the same partitioning rule to create train and test splits to train a linear regression model using word2vec features and report your RMSE error. How do your results change after creating word2vec bag-of-word features?

Question 8: (10 marks) Different approaches can be used to improve the sentiment analysis model you have created. The following is a short list of suggestions:

1. Different features: We could try different encoding techniques instead of bag-of-words. For example, VLAD [2] or Fisher encoding [3] or a concatenation of TF-IDF with word2vec features.
2. Different models: We could use different regression models sich as a random forest regression model.
3. Exploring model parameters: We saw that the size of word2vec features can affect its accuracy at the cost of a longer training time.

Implement one or a combination of the above approaches. Design experiments to validate approaches from your choice on the test dataset. Report how your performance changed. Feel free to propose approaches other than the ones listed above.

7 Submission

Your assignment should be submitted online at `https://courses.cs.sfu.ca/`. You should submit two files for this assignment :

1. **Report:** Create a PDF report including all your answers to the above questions.
2. **Code:** Create an archive file containing all scripts used for this assignment. Your scripts should be well documented and able to reproduce all the reported results. Please include a small readme.txt file in your archive to briefly explain which script is used for which question.

[2] Arandjelovic, Relja, and Andrew Zisserman. "All about VLAD." 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[3] Fisher Vectors: Beyond Bag-of-Visual-Words Image Representations (Computer Vision,Imaging and Computer Graphics) Part 1 `http://what-when-how.com`