Assignment 2: Text Processing and Classification using Apache Spark (100p)

Submission Deadline

May 13, 2025 (23:59)

Goal

In this assignment, you will implement a Spark program to process large text corpora. You will be working with the same group as in Assignment 1.

Environment

Again, you will be using your account on the 12-node Hadoop Cluster `lbd.tuwien.ac.at`. Please use your SSH client of choice to connect to the cluster.

You can develop and run your Spark programs either:

- directly on the cluster in Jupyter notebooks in the provided JupyterHub environment (**Preffered approach**)
- locally on a subset of the data
- using the interactive Spark shells on the cluster (`spark-shell`, `pyspark`) or
- submit your Spark jobs to Yarn in Cluster mode (using `spark-submit`, cf. slides)

For the second option, ie. local development, we do not offer an LBD-specific Docker image. However, for developing in Jupyter Notebooks with PySpark, the dataLAB refers to the Jupyter Spark Docker Container as a possible option. See also this tutorial to create your Spark Docker container if necessary.

Regarding running notebooks in the JupyterHub environment on the LBD cluster, note that there is a time limit of 48 hours for running Jupyter kernels (graciously extended by the dataLAB team from the cluster-wide two hour time limit). The reason for the time limit for running kernels is that as long as they are not shut down, they use resources. From experience users tend to forget to shut down kernels when they're finished, leading to resources being all used up rather quickly and rendering the cluster inoperable for all others. If the resources (memory) get scarce, time limit would be reduced.

As such, please make sure to stop your Jupyter kernels once you are done and do not use the interactive environment for long-running processing tasks.

> **Always shut down your kernels after usage, do not simply just close the tab.**

Again, do not "probe" the data with trial and error using non-Spark packages and data structures. Also here, it will slow (or eventually bring) down the cluster for everyone. Keep an eye on the resources your jobs are using and kill them if necessary.

Be aware that high cluster utilization is expected until the end of the submission, especially before the submission deadlines.

Therefore,

- always stop their Spark Contexts after finishing an analysis
- always shut down kernels when a Spark pipeline is complete
- always shut down Jupyter notebooks when not in use
- test jobs with a small sample of the data first
- plan ahead and do not wait until the last days before the deadline for your analyses

Dataset and Tasks

You will be reusing the *Amazon Review Dataset* from Assignment 1 and implement partial and similar functionality in parts 1 and 2, respectively, this time by making use of Spark. In Part 3, you will make use of the developed processing pipeline as input for a text classification task.

For submission (see below), produce the files requested using the development set to keep cluster usage low. Also make comparisons with Assignment 1 using only the development set.

The reduced data set for development can be found at

`hdfs:///user/dic25_shared/amazon-reviews/full/reviews_devset.json`

The full dataset can be found at

`hdfs:///user/dic25_shared/amazon-reviews/full/reviewscombined.json`

Part 1: RDDs

Repeat the steps of Assignment 1, i.e. calculation of chi-square values and output of the sorted top terms per category, as well as the joined dictionary, using RDDs and transformations.

Write the output to a file `output_rdd.txt`. Compare the generated `output_rdd.txt` with your generated `output.txt` from Assignment 1 and describe your observations briefly in the submission report.

Part 2: Datasets/DataFrames: Spark ML and Pipelines

Convert the review texts to a classic vector space representation with TFIDF-weighted features based on the Spark DataFrame/Dataset API by building a transformation pipeline. The primary goal of this part is the preparation of the pipeline for Part 3 (see below).

> Note: although parts of this pipeline will be very similar to Assignment 1 or Part 1 above, do not expect to obtain identical results or have access to all intermediate outputs to compare the individual steps.

Use built-in functions for tokenization to unigrams at whitespaces, tabs, digits, and the delimiter characters ()[]{}.!?,;:+=-_"'`~#@&*%€$§\/, casefolding, stopword removal, TF-IDF calculation, and chi square selection) (using 2000 top terms overall).

Write the terms selected this way to a file `output_ds.txt` and compare them with the terms selected in Assignment 1. Describe your observations briefly in the submission report).

Part 3: Text Classification

In this part, you will train a text classifier from the features extracted in Part 2. The goal is to train a model that can predict the product category from a review's text.

To this end, extend the pipeline from Part 2 such that a **Support Vector Machine** classifier is trained. Since we are dealing with multi-class problems, make sure to put a strategy in place that allows binary classifiers to be applicable. Apply vector length normalization before feeding the feature vectors into the classifier (use `Normalizer` with L2 norm).

Follow best practices for ML experiment design and investigate the effects of parameter settings using the functions provided by Spark:

- Split the review data into training, validation, and test set.
- Make experiments reproducible.
- Use a grid search for parameter optimization:
    - Compare chi square overall top 2000 filtered features with another, heavier filtering with much less dimensionality (see Spark ML documentation for options).
    - Compare different SVM settings by varying the regularization parameter (choose 3 different values), standardization of training features (2 values), and maximum number of iterations (2 values).
- Use the `MulticlassClassificationEvaluator` to estimate performance of your trained classifiers on the test set, using F1 measure as criterion.

> *Note:* Again, to ensure availability of the cluster to as many students as possible, resort to the development set to build, optimize, and evaluate the classifier. You may further downsample the development set to make model training easier.

Report

Produce a `report.pdf`, that contains detailed report inlcuding at least five sections:

1. Introduction
2. Problem Overview
3. Methodology and Approach
4. Results
5. Conclusions

The Methodology and Approach section should have a figure **illustrating your strategy and pipeline in one figure** (1 page maximum).

Include the performance indicators obtained over the different setting explored and interpret the results obtained.

The overall report should not exceed more than 8 pages (A4 size).

Important notes

- Efficiency of the implementation is a crucial aspect of this assignment. Consider carefully how you design your implementation in order to avoid unnecessary overheads and calculations while achieving the expected final results.
- For all parts, a Jupyter notebook is the preferred format. Writing Spark jobs is also acceptable, however, in this case pay close attention to **documenting all code** submitted as well as **intermediate outputs, graphs, etc.** to make your choices traceable.

Scoring

- Part 1: max. 30 points (correctness and resource/runtime efficiency)
- Part 2: max. 25 points (correctness and resource/runtime efficiency)
- Part 3: max. 25 points (correctness and resource/runtime efficiency)
- Code documentation: 10 points
- Report: 10 points

Maximum total score: **100 points**

Submission

Submission Files

Please submit a single file named `<GroupID>_DIC2025_Ex2.zip` that contains:

- `output_rdd.txt`, `output_ds.txt`: results obtained
- `report.pdf`: 8 page report (max), 11pt font size, one column format
- `src/`: subdirectory containing **all sources** and a **structured and documented Jupyter notebook** (preferred) and/or the **very well documented source code** (Spark jobs) of your implementation and experiments (Java, Scala, or PySpark) and all intermediate outputs, graphs, etc.

Submission procedure

Submit your solution via TUWEL before **13.05.2025 (23:59)**.