

Assignment Big Data MA120

Exploring BitCoin



General information

This assignment counts 75% towards the final grade for the course. The project can be completed individually by each student or in the group of up to two students. All the assignment materials must be delivered via It's Learning well before the deadline. For any delay, a valid absence documentation must be presented to the school administration.

The assignment consists of two parts:

1. An implementation reflected in a code base (delivered as a zip/tar.gz);
2. A report describing the implementation, structure and student reflection.

Both parts together affect the grading. This project delivery is not anonymized. Any assumptions made by students must be clearly stated in the report. You are free to use original Hadoop with Java or choose any other language that works with Hadoop streaming. Report should include clear descriptions of the code.

The dataset is small enough and does not require you to run on a cluster.

The Assignment: Exploring Bitcoin

Introduction

Bitcoin Stack Exchange (www.stackexchange.com) is a question and answer site for Bitcoin crypto-currency enthusiasts. Chances are high that you will find an answer to questions related becoming a bitcoin ledger.

Dataset

The dataset is an anonymized dump of all user-contributed content on the Bitcoin stack exchange network. Bitcoin stack exchange is a question and answer site for people interested in finding an answer to a question they might have. You can check it out at: bitcoin.stackexchange.com.

The dataset is formatted as a separate archive consisting of XML files. The archive includes:

- Badges.xml
- Comments.xml
- PostHistory.xml
- PostLinks.xml
- Posts.xml
- Tags.xml
- Users.xml
- Votes.xml

For complete schema information, see the included readme.txt in appendix A.

Obtaining the dataset

The dataset is available from:

archive.org/download/stackexchange/bitcoin.stackexchange.com.7z

Tasks

This section describes the actual tasks to be performed by each student/- group. The bolded word before the task text suggests the name of the task.

1. Warmup
 - a. **WordCount**. Count the words in the body of questions (note the PostTypeId). This is the classic WordCount. The resulting data should

include counts for each word, that is, how many times each word appears in the body of questions.

- b. **Unique words.** Write a Hadoop MapReduce job that outputs words in the question titles. The output should contain all words used in the title of questions, only once. No count, just the word. That will be the dictionary over titles of the questions.
- c. **MoreThan10.** How many questions are there which have more than ten words in their titles?
- d. **Stopwords.** Based on a), exclude the stopwords from titles (an example list can be obtained at: <http://j.mp/STOPWORDS>. We refer to the output of this task as to *popular words*).
- e. **Pig top 10.** Write a Pig script that selects the top 10 list words after you remove the stopwords in step d).
- f. **Tags.** Write a MapReduce job that creates a dictionary over the unique tags (that is unique tags in the whole dataset, identified by @Tags attribute).

2. Discover

- a. **Counting.** How many unique users are in the dataset? A unique user is identified by @Id attribute.
- b. **Unique users.** Write a Hadoop MapReduce job that outputs unique users in the dataset.
- c. **Top Miners.** Write a Hadoop MapReduce job that outputs top 10 users in terms of their reputation.
- d. **Top questions.** Write a Hadoop MapReduce job that outputs top 10 questions in terms of their reputation (Score).
- e. **Favorite questions.** Write a Hadoop MapReduce job that outputs top 10 questions in terms of their FavouriteCount.
- f. **Average answers.** Calculate an average number of answers per question. You choose whether you want to use MapReduce or Pig.
- g. **Countries.** Discover users by countries, that is the output should be country and the number of users. For this task, use Location attribute.
- h. **Names.** What is the most popular name of a user? List top 10 names. Hint: you may want to split the name by space.
- i. **Answers.** How many questions do have at least one answer?

3. Numbers

- a. **Bigram.** A pair of adjacent words is called a bigram. For example, “big data” or “fast car” are examples of bigrams. Find the most common bigram in the titles of the questions.
 - b. **Trigram.** Do the same for word-level trigrams, i.e. a three words that appear consecutively.
 - c. **Combiner.** In 1a) you created a WordCounter. Add a combiner to it. What are implications of having a combiner?
 - d. **Useless.** The word “useless” is pretty much useless without any context. Count how many questions contain this word in the body.
4. Search engine.
- a. **Title index.** Search engines maintain an index over dictionary with reference to the documents where they appear. The purpose is to quickly locate documents so when you search it is very fast to locate the documents. Documents in the context of the project is a question.


In this task we create an index over titles and bodies of questions (including answers). What we are after is having a simple index that lists publications in which a search term/s occur/s. The reference to the question in which the terms appear should be the attribute key.

Lets have a look at an example (for readability the example is stripped down to minimum). Given the xml below:

```
<posts>
  <row Id="10" Title="Cannot obtain secure database connection".../>
  <row Id="25" Title="How can I secure my database connection?".../>
[...]
```

The index entries (output) for these two posts would be:

```
can 25
cannot 10
connection 10,25
```



database 10,25
how 25
i 25
my 25
obtain 10
secure 10, 25

In this dictionary we use the term as key and list of post ids as values in which the term appears. Our index is simple, it does not weight posts based on the number of occurrences of the terms in the posts.

Practical parts

Use your own input format class that reads the stream of XML. You can set the class name by calling 'setInputFormatClass()' method of the Job class in your MapReduce client (driver) code.

Groups

Students are encouraged to work in groups. Groups can consist of up to 2 students. Deadline for group registration is Wednesday, 9th of October. Send your registration to: taknai@westerdals.no.


Guidance hour (veiledningstime): is announced during lectures, and will take place for each team individually.

Deadline for project delivery: **Friday, 18th of October 2019, 15:00.**

Part-time students normally have extra time to deliver the project. Deadline for **part-time students** is: **Monday, 18th of November 2019, 15:00.**

Structuring the results

Tasks should be independent of each other. You should not mix the implementation of one task with another. This may interfere and affect the results of the output unintentionally. Therefore, you may organize your zipped project file for each task. Some



tasks depend on each other. You can copy the folder to the depending task change the code there.

Appendices

Appendix A ***readme.txt***

This description is taken from: <https://meta.stackexchange.com/questions/2677>

- the foreign key fields are formatted as [links](#) to their parent table
- *italic* table names are found in both the Data Dump on [Archive.org](#) as well as in [SEDE](#)

Posts

- Id
- PostTypeId (listed in the [PostTypes](#) table)
 1. Question
 2. Answer
 3. Orphaned tag wiki
 4. Tag wiki excerpt
 5. Tag wiki
 6. Moderator nomination
 7. "Wiki placeholder" (seems to only be the [election description](#))
 8. Privilege wiki
- [AcceptedAnswerId](#) (only present if PostTypeId is 1)
- [ParentId](#) (only present if PostTypeId is 2)
- CreationDate
- DeletionDate (only non-null for the SEDE [PostsWithDeleted](#) table. Deleted posts are not present on [Posts](#). Column not present on data dump.)
- Score
- ViewCount (nullable)
- Body ([as rendered HTML](#), not Markdown)
- [OwnerUserId](#) (only present if user has not been deleted; always -1 for tag wiki entries, i.e. the community user owns them)
- OwnerDisplayName (nullable)
- [LastEditorUserId](#) (nullable)
- LastEditorDisplayName (nullable)

- `LastEditDate`="2009-03-05T22:28:34.823" - the date and time of the most recent edit to the post (nullable)
- `LastActivityDate`="2009-03-11T12:51:01.480" - the date and time of the most recent activity on the post. For a question, this could be the post being edited, a new answer was posted, a bounty was started, etc.
- `Title` (nullable)
- `Tags` (nullable)
- `AnswerCount` (nullable)
- `CommentCount`
- `FavoriteCount`
- `ClosedDate` (present only if the post is closed)
- `CommunityOwnedDate` (present only if post is community wikied)

Users

- `Id`
- `Reputation`
- `CreationDate`
- `DisplayName`
- `LastAccessDate` ([The time this user last loaded a page on the site; updated at most once every 30 minutes](#))
- `WebsiteUrl`
- `Location`
- `AboutMe`
- `Views`
- `UpVotes` ([How many upvotes the user has cast](#))
- `DownVotes`
- `ProfileImageUrl`
- `EmailHash` (now always blank)
- `AccountId` (StackExchange Network profile Id of the user)
- `Age`

Comments

- `Id`
- `PostId`

- `Score`
- `Text` (The text of the comment.)
- `CreationDate`
- `UserDisplayName`
- `UserId` (Optional. Absent if user has been deleted)

Badges

- `Id`
- `UserId`
- `Name` (Name of the badge)
- `Date`, e.g.: "2008-09-15T08:55:03.923"
- `Class`
 1. Gold
 2. Silver
 3. Bronze
- `TagBased`, true if badge is for a tag, otherwise it is a named badge

CloseAsOffTopicReasonTypes

- `Id`
- `IsUniversal`
- `MarkdownMini` — the text of the close reason (in markdown syntax)
- `CreationDate`
- `CreationModeratorId`
- `ApprovalDate`
- `ApprovalModeratorId`
- `DeactivationDate`
- `DeactivationModeratorId`

PendingFlags

Despite the name, this table in fact contains close-related flags and votes.

- `Id`
- `FlagTypeId` (listed in the `FlagTypes` table)
- 13. canned flag for closure

- 14. vote to close
- 15. vote to reopen
- [PostId](#)
- [CreationDate](#)
- [CloseReasonTypeId](#) (listed in the [CloseReasonTypes](#) table)
- [CloseAsOffTopicReasonTypeId](#), if [CloseReasonTypeId](#) = 102 (off-topic) (listed in the [CloseAsOffTopicReasonTypes](#) table)
- [DuplicateOfQuestionId](#), if [CloseReasonTypeId](#) is 1 or 101 (old duplicate or current duplicate)
- [BelongsOnBaseHostAddress](#), for votes to close and migrate

PostFeedback

Collects up and down votes from anonymous visitor and/or unregistered users. See [here](#)

- [Id](#)
- [PostId](#)
- [IsAnonymous](#)
- [VoteTypeId](#) (listed in the [VoteTypes](#) table)
- 2. UpMod
- 3. DownMod
- [CreationDate](#)

PostHistory

- [Id](#)
- [PostHistoryTypeId](#) (listed in the [PostHistoryTypes](#) table)
- 1. Initial Title - The first title a question is asked with.
- 2. Initial Body - The first raw body text a post is submitted with.
- 3. Initial Tags - The first tags a question is asked with.
- 4. Edit Title - A question's title has been changed.
- 5. Edit Body - A post's body has been changed, the raw text is stored here as markdown.
- 6. Edit Tags - A question's tags have been changed.
- 7. Rollback Title - A question's title has reverted to a previous version.
- 8. Rollback Body - A post's body has reverted to a previous version - the raw text is stored here.

- 9. Rollback Tags - A question's tags have reverted to a previous version.
- 10. Post Closed - A post was voted to be closed.
- 11. Post Reopened - A post was voted to be reopened.
- 12. Post Deleted - A post was voted to be removed.
- 13. Post Undeleted - A post was voted to be restored.
- 14. Post Locked - A post was locked by a moderator.
- 15. Post Unlocked - A post was unlocked by a moderator.
- 16. Community Owned - A post has become community owned.
- 17. Post Migrated - A post was migrated. *superseded now with id 35 and 36 (away/here)*
- 18. Question Merged - A question has had another, deleted question merged into itself.
- 19. Question Protected - A question was protected by a moderator.
- 20. Question Unprotected - A question was unprotected by a moderator.
- 21. Post Disassociated - An admin removes the OwnerUserId from a post.
- 22. Question Unmerged - A previously merged question has had its answers and votes restored.
- 24. Suggested Edit Applied
- 25. Post Tweeted
- 31. Comment discussion moved to chat
- 33. Post notice added *comment contains foreign key to PostNotices*
- 34. Post notice removed *comment contains foreign key to PostNotices*
- 35. Post migrated away *replaces id 17*
- 36. Post migrated here *replaces id 17*
- 37. Post merge source
- 38. Post merge destination
- Additionally, in [older dumps](#) (all guesses, all seem no longer present in the wild):
- 23. Unknown dev related event
- 26. Vote nullification by dev (erm?)
- 27. Post unmigrated/hidden moderator migration?
- 28. Unknown suggestion event
- 29. Unknown moderator event (possibly de-wikification?)
- 30. Unknown event (too rare to guess)
- [PostId](#)
- RevisionGUID: At times more than one type of history record can be recorded by a single action. All of these will be grouped using the same RevisionGUID

- `CreationDate`: "2009-03-05T22:28:34.823"
- `UserId`
- `UserDisplayName`: populated if a user has been removed and no longer referenced by user Id
- `Comment`: This field will contain the comment made by the user who edited a post.
 - If `PostHistoryTypeId` = 10, this field contains the `CloseReasonId` of the close reason (listed in `CloseReasonTypes`):
 - **Old close reasons:**
 - 1. Exact Duplicate
 - 2. Off-topic
 - 3. Subjective and argumentative
 - 4. Not a real question
 - 7. Too localized
 - 10. General reference
 - 20. Noise or pointless (Meta sites only)
 - **Current close reasons:**
 - 101. Duplicate
 - 102. Off-topic
 - 103. Unclear what you're asking
 - 104. Too broad
 - 105. Primarily opinion-based
 - If `PostHistoryTypeId` = 33 or 34 this field contains the `PostNoticeId` of the PostNotice
- `Text`: A raw version of the new value for a given revision
 - If `PostHistoryTypeId` = 10, 11, 12, 13, 14, 15, 19, 20 and 35 this column will contain a JSON encoded string with all users who have voted for the `PostHistoryTypeId`
 - If it is a duplicate close vote, the JSON string will contain an array of original questions as `OriginalQuestionIds`
 - If `PostHistoryTypeId` = 17 this column will contain migration details of either from `<url>` or to `<url>`

PostLinks

- `Id` primary key
- `CreationDate` when the link was created

- `PostId` id of source post
- `RelatedPostId` id of target/related post
- `LinkTypeId` type of link
 - 1: Linked (`PostId` contains a link to `RelatedPostId`)
 - 3: Duplicate (`PostId` is a duplicate of `RelatedPostId`)

PostNotices

- `Id`
- `PostId`
- `PostNoticeTypeId`
- `CreationDate`
- `DeletionDate`
- `ExpiryDate`
- `Body` (when present contains the custom text shown with the notice)
- `OwnerUserId`
- `DeletionUserId`

PostNoticeTypes

- `Id`
- `ClassId`
- 1. Historical lock
- 2. Bounty
- 4. Moderator notice
- `Name`
- `Body` (contains the default notice text)
- `IsHidden`
- `Predefined`
- `PostNoticeDurationId` (-1: No duration specified, 1: 7 days (bounty))

PostsWithDeleted

Similar to Posts table but with deleted post as well. For deleted posts the extra column `DeletionDate` is not null. Listed here are the fields that contain info when `DeletionDate` is not null.

- Id
- PostTypeId (listed in the [PostTypes](#) table)
- ParentId (only on when PostTypeId = 2 (answer))
- CreationDate
- DeletionDate
- Score
- Tags
- ClosedDate

PostTags

- [PostId](#)
- [TagId](#)

ReviewRejectionReasons

Canned rejection reasons for suggested edits. See [Show all review rejection reasons](#)

- Id
- Name
- Description
- PostTypeId (for reasons that apply to specific post types only)

ReviewTaskResults

- Id
- [ReviewTaskId](#)
- ReviewTaskResultTypeId (listed in [ReviewTaskResultTypes](#))
- 1. Not Sure
- 2. Approve (suggested edits)
- 3. Reject (suggested edits)
- 4. Delete (low quality)
- 5. Edit (first posts, late answers, low quality)
- 6. Close (close, low quality)
- 7. Looks OK (low quality)
- 8. Do Not Close (close)
- 9. Recommend Deletion (low quality answer)

- 10. Recommend Close (low quality question)
- 11. I'm Done (first posts)
- 12. Reopen (reopen)
- 13. Leave Closed (reopen)
- 14. Edit and Reopen (reopen)
- 15. Excellent (community evaluation)
- 16. Satisfactory (community evaluation)
- 17. Needs Improvement (community evaluation)
- 18. No Action Needed (first posts, late answers)
- `CreationDate`
- `RejectionReasonId` (for suggested edits; listed in `ReviewRejectionReasons`)
- `Comment`

ReviewTasks

- `Id`
- `ReviewTaskTypeId` (listed in [ReviewTaskTypes](#))
- 1. Suggested Edit
- 2. Close Votes
- 3. Low Quality Posts
- 4. First Post
- 5. Late Answer
- 6. Reopen Vote
- 7. Community Evaluation
- 8. Link Validation
- 9. Flagged Posts
- 10. Triage
- 11. Helper
- `CreationDate`
- `DeletionDate`
- `ReviewTaskStateId` (listed in [ReviewTaskStates](#))
 - 1. Active
 - 2. Completed
 - 3. Invalidated
- [PostId](#)

- [SuggestedEditId](#) (for suggested edits, which have their own numbering for historical reasons)
- [CompletedByReviewTaskId](#) id associated to the ReviewTaskResult that stores the outcome of a completed review.

SuggestedEdits

- Id
- [PostId](#)
- CreationDate
- ApprovalDate - NULL if not approved (yet).
- RejectionDate - NULL if not rejected (yet). If both approval and rejection date are null then this edit is still in review (and its corresponding entry in ReviewTasks will have an active state as well).
- [OwnerUserId](#)
- Comment
- Text
- Title
- Tags
- RevisionGUID

SuggestedEditVotes

- Id
- [SuggestedEditId](#)
- [UserId](#)
- VoteTypeId
 1. Approval
 2. Reject
- CreationDate
- [TargetUserId](#)
- TargetRepChange

Tags

- Id
- TagName


- Count
- [ExcerptPostId](#)
- [WikiPostId](#)

TagSynonyms

- Id
- [SourceTagName](#)
- [TargetTagName](#)
- CreationDate
- [OwnerUserId](#)
- AutoRenameCount
- LastAutoRename
- Score
- [ApprovedByUserId](#)
- ApprovalDate

Votes

- Id
- [PostId](#)
- VoteTypeId (listed in the [VoteTypes](#) table)
- 1. AcceptedByOriginator
- 2. UpMod ([AKA upvote](#))
- 3. DownMod ([AKA downvote](#))
- 4. Offensive
- 5. Favorite (UserId will also be populated)
- 6. Close (since 2013-06-25 close votes are ONLY stored in the PostHistory table)
- 7. Reopen
- 8. BountyStart (UserId and BountyAmount will also be populated)
- 9. BountyClose (BountyAmount will also be populated)
- 10. Deletion
- 11. Undeletion
- 12. Spam
- 15. ModeratorReview

- 
- 16. ApproveEditSuggestion
 - UserId (only present if VoteTypeid is 5 or 8)
 - CreationDate ([time data is purposefully removed](#) to protect user privacy)
 - BountyAmount (only present if VoteTypeid is 8 or 9)

