Openclassrooms - Parcours Data Scientist

Projet N° 6

Catégorisez automatiquement des questions

Classify questions automatically

|  |  |
| --- | --- |
| **DOCUMENT** | |
| **Reference** | **File Name** |
|  | Openclassroms Data Scientist projet 6 rapport Christian Muths.docx |

|  |  |
| --- | --- |
| **Versions** | |
| **Date** | **Reason for change** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table of contents

[0. Introduction 2](#_Toc514321633)

[1. Principle of tag recommendation system based on semi-supervised algorithm 2](#_Toc514321634)

[1.1 List of stems 2](#_Toc514321635)

[1.2 Matrix of topics per corpus 3](#_Toc514321636)

[1.2.1 Topic extraction 3](#_Toc514321637)

[1.2.2 Corpus-topic probability 4](#_Toc514321638)

[1.3 Tag decomposition 4](#_Toc514321639)

[1.4 Tag weight per topic 5](#_Toc514321640)

[1.5 New questions classification 5](#_Toc514321641)

# Introduction

The target of this project is to propose to a user creating a question on the stackoverflow.com site, a set of relevant tags.

The algorithms propose 5 to 10 tags of which the user is invited to pick and choose some of them to add to the question.

The performance measurement is based on a comparison with the real tags the user has given. Out of the real tags, we measure how many we guessed. For example, if the user has set 4 tags, and if in the proposed list 3 of them were proposed, the score for this question is 75%



The overall performance indicator is the relative number of questions for which the proposed tags contain at least 50% of the tags the user had chosen.

# Principle of tag recommendation system based on non-supervised algorithm

This algorithm is based on a non-supervised topic extraction, associated with a statistical approach to get a list of most likely tags to propose.

Based on a list of questions having titles, descriptions and human-selected tags, several data structures are being built:

## List of stems

A stem is the root of a word, regardless of the suffixes, conjugation, gender, etc... This reduces the number of values and helps grouping similar notions. However, it may loose some useful information.

A column is added into the questions dataset with the list of stems for each question. Stems are extracted both from the title and the body.

The HTML content of the description gets its HTML tags cleaned out, then a stemmer is used to extract the useful words into stems.

The steps to get the list of stems for each question are:

* Concatenate title and body
* Use BeautifulSoup to remove HTML tags
* Convert into lower case
* Tokenize the resulting string into an array of words
* Remove stopwords using the nltk.stowords list for English language.
* Convert remaining words into stems using nltk SnowballStemmer
* Append the list of stems into a new column of the questions dataframe.

Out of 15.000 questions, we got a total of approx. 1 Million stem occurrences, with 40.000 unique stems.

The set of stems gets vectorized. Each question is a line of the matrix and each stem is a column. The values are the number of occurrences of each stem in each question. Any word that is used in more than 95% of the questions or less than 5 times across all questions is removed, because its either too common or too specific to be used for the topic determination.

The result is the stem\_matrix which has 5.755 stems in columns, and 15.000 lines. Only approx. 25% of the initial stems were kept.

5700 stems as columns

15.000 questions as lines

stem\_matrix

Maximum number of stems per question: 2386

Minimum number of stems per question: 2

Average number of stems per question: 56

Average number of unique stems per question: 32

## Matrix of topics per corpus

### Topic extraction

The method used to extract topics is the Latent Dirichlet Allocation. An optimization is done on the number of topics as a parameter.

The measurement of score and Perplexity shows that:

* The score doesn’t vary much with the number of topics
* the perplexity strictly increases with the number of topics when tested between 2 and 90. It almost doubles with each additional topic. It's close to 1000\*2^(ntopics-1).

These measurements aren’t useful to choose the optimal number of topics. The final evaluation will be based on the ratio of correct tags proposed.

Example of 15 topics top words. Some of them are intuitive (Topic 0 = git, Topic 1 = Python, Topic 3 = numbers, dates and times, Topic 6 = SQL databases, Topic 11 = CSS), while some others are less meaningful in a human perspective.

Topic #0: file git command use directori commit run chang project branch local work path want window folder line repositori tri script

Topic #1: python test line number print import self py modul def virtualenv get key return code str file function tri call

Topic #2: java class public method object new string static void return thread privat call except system null properti code use get

Topic #3: date 00 10 11 05 12 format 15 time 01 02 03 datetim 24 androidruntim 19 13 20 26 18

Topic #4: use differ code understand googl mean question map read vs oper implement data time one warn could applic compil know

Topic #5: div use control set class html work item id model tag element name scope like get properti content ng want

Topic #6: tabl column databas key sql select null queri data mysql row id valu insert creat name server updat set db

Topic #7: function var name js javascript json valu node php type foo npm option script data input get return jqueri text

Topic #8: android instal version org error id packag build com lib eclips librari app activ java xml usr project view depend

Topic #9: error user app get server http request use applic com system url web tri connect log run messag client password

Topic #10: int array return valu function std type size integ byte swift length fals true doubl element char convert const number

Topic #11: text color css imag background style width left center border 100 right height box margin size pad align red label

Topic #12: button page view click event jqueri want use div li chang imag class show element like disabl load work tab

Topic #13: net framework td asp height mvc entiti tr use width font differ size linq space pixel content scale pdf vs

Topic #14: string use way like list would one want someth know need exampl variabl get best two case look code question

### Corpus-topic probability

The result of the LDA transformation lda.fit\_transform(stem\_matrix) is the matrix of probability for a document to belong to a topic.

15.000 questions as columns

15 topics as lines

lda\_corpus.T

## Tag decomposition

First step is to vectorize the list of tags assigned by the author. The result is a sparse matrix with

5400 tags as columns

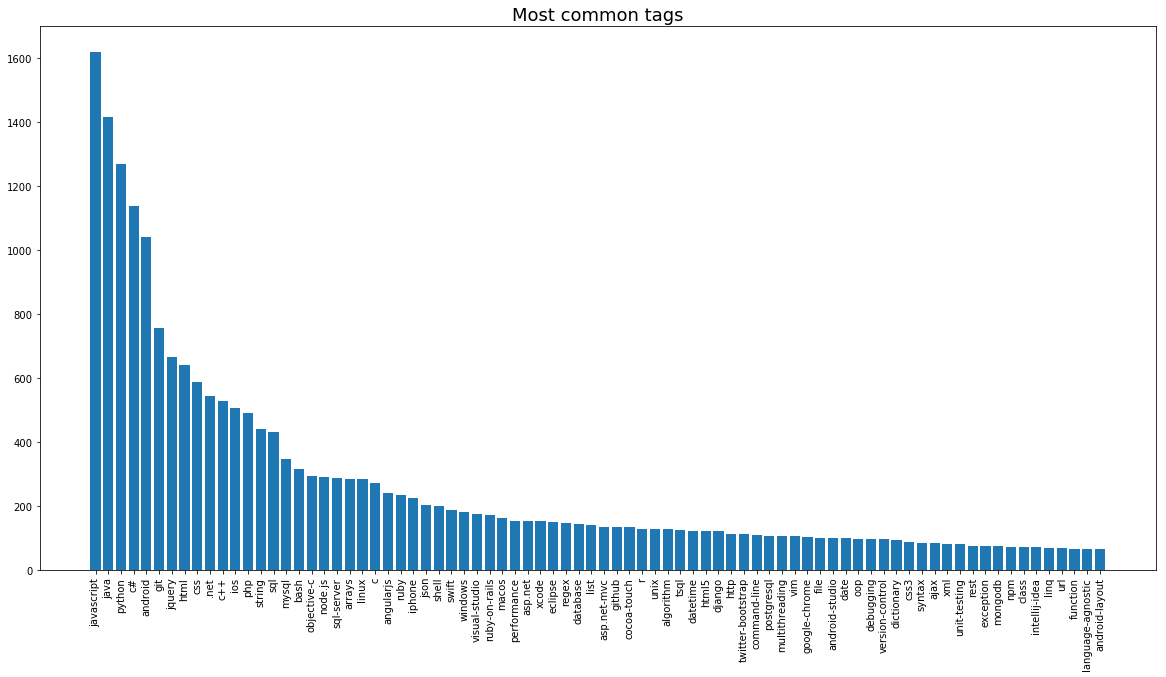
15.000 questions as lines

tag\_matrix

Maximum number of tags per question: 5

Minimum number of tags per question: 1

Average number of tags per question: 3



## Tag weight per topic

A matrix that gives the weight of each tag for each topic is created by multiplying the lda\_corpus matrix by the tag\_matrix. Weight can be greater than 1 if a tag is used several times in a topic.

lda\_corpus.T tag\_matrix topic\_tag\_weight

## New questions classification

New questions are classified into topics using the classifier which was trained on the train set. The result is the vector of the question’s probabilities to belong to each topic.

The steps followed to reach this are very similar to the process done on the known questions:

* Concatenate title and body
* Use BeautifulSoup to remove HTML tags
* Convert into lower case
* Tokenize the resulting string into an array of words
* Remove stopwords using the nltk.stowords list for English language.
* Convert remaining words into stems using nltk SnowballStemmer
* Vectorize the list of stems using the vectorizer which was fitted with the training set. This ensures to stick to the same set of stems.
* Transform the stem matrix using the LDA transformer which was fitted with the training set. This ensures to use the same exact topics and gets the probabilities for each new question to belong to each topic.

## Tag suggestions for new questions

# Principle of tag recommendation system based on supervised algorithm

END OF DOCUMENT