Analyzing comments published by developers on pull requests on Github

Maël Fabien

Data Science Assignment for Veamly

1 The context

The dataset carries 4000 comments that were published on pull requests on Github by developer teams.

The feature columns are the following :

- Comment: the comment made by a developer on the pull request.
- Comment date: date at which the comment was published
- Is merged: shows whether the pull request on which the comment was made has been accepted (therefore merged) or rejected.
- Merged at: date at which the pull request was merged (if accepted).
- Request changes: each comment is labelled either 1 or 0: if it's labelled as 1 if the comment is a request for change in the code. If not, it's labelled as 0.

The aim of this study is to dig deeper into the nature of blockers and analyze the requests for change.

The following questions will be addressed :

- What are the most common problems that appear in these comments?
- Can we cluster the problems by topic/problem type?
- How long is the resolution time after a change was requested?

2 Exploratory data analysis

2.1 Numerical features

After a first analysis, the key facts we can extract are the following :

- there are no missing values, except for the merging date when no merge occurred
- 2842 pull requests have not been merged, leading to an overall rate of pull request acceptance of 30%.
- the average comment length is 31 words
- 23% of the comments labeled as requiring a change in the code have lead to a change in the code
- most comments were written in 2017
- the average merge time for comment that required changes in code is 3 days

- there are in the comments replies of the developers that were written after the merge of the request. These comments are deleted since they cannot be used when we try to forecast the time with which the update will be released
- the comment date was split into several features including : Year, Month, Day of Month, Day of Week and Hour
- most comments are published at winter time, and between Monday and Friday

The scree plot of the eigen values indicates no "cliff" in the eigenvalues when assessing the importance of the different dimensions. However, when looking at the circle of correlations (cf. Figure 3), we notice that :

- when the year at which the comment is posted increases, there is more chances that the pull request is merged (which seems logic since the project has been more active lately)
- when the day of the month increases (closer to end of the week), we also have more chances to see a merge in the future, although the relationship is weaker
- the day of the week at which the comment is posted is uncorrelated with the fact of having the pull request merged
- if a code change is required, the likelihood of merging decreases (which we observed empirically above)

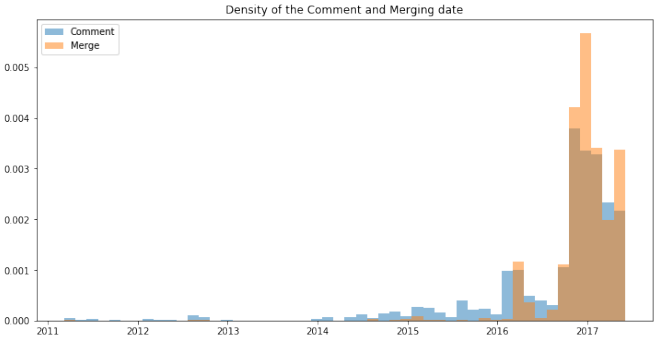

Fig. 1. Density of the message date, sorted by comment and merge date

2.2 Cleaning the comments

The vocabulary of the comments is quite specific since the comments have been extracted from GitHub. Some comments contain copy-pasted code, others contain web links... The most frequent words used are shown in the Word Cloud of Figure 2.

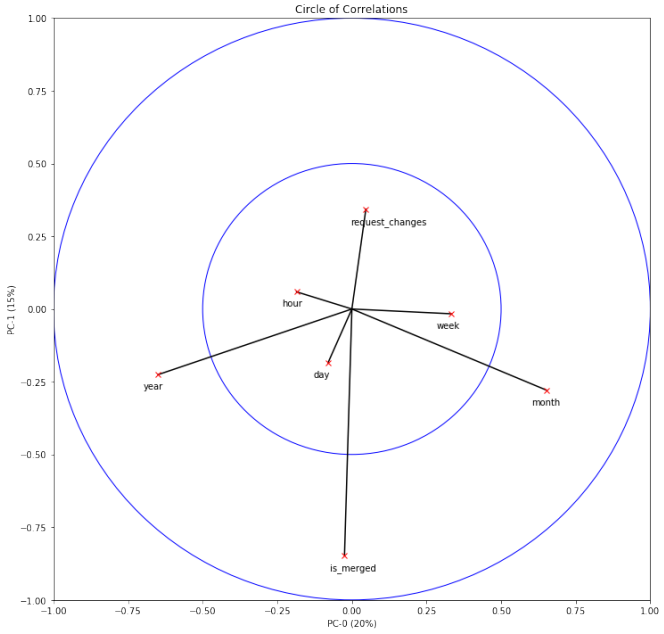

Fig. 2. Correlation circle of the PCA mapping

Fig. 3. Word cloud after cleaning the comments

The following pipeline has been developed in order to clean efficiently the comments :

- replace typical code expressions
- tokenizing
- remove small words of less than two characters
- remove special characters linked to code copy pasting
- remove stop words using NLTK english stop words
- lemmatizing using WordNet Morphy

This reduces the overall number of words from 149'000 to 59'000. The figure 5 presents the evolution of the overall number of words we are dealing with while cleaning the comments.

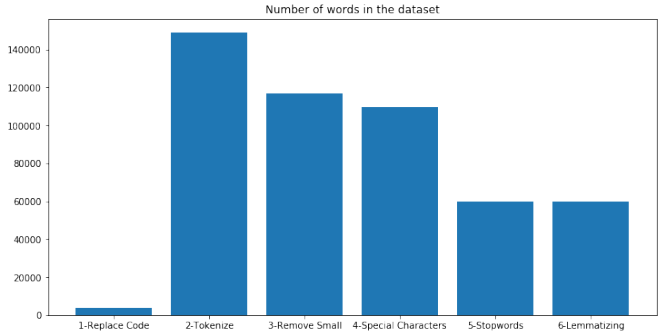

Fig. 4. Number of words of each step of the cleaning

3 What are the main topics in the comments ?

In order to answer the question of the main topics that arise in the comments, we will quickly cover the theory of topic modelling. One famous algorithm in the field of natural language processing that allows us to do this is the Latent Dirichlet Allocation (LDA). LDA was developped in 2003 by David Blei, Andrew Ng and Michael Jordan.

We will be relying on *gensim*, an open source topic modelling in Python, as well as a more visual library called *pyLDAvis.gensim*.

How does LDA work ? The model takes a set of words as input. Each word has a given probability to belong to a certain topic. The LDA should take as an input the number of topics that we do expect. We take a collection of documents as an input, and learn which model describes it best.

There are two other hyper parameters :

- α that sets the prior on the per-document topic distribution. A high α value means that each document is likely to contain a mix of topics, and not only focus on some topics.
- β that sets the prior on the per-topic word distribution. A high β value means that each topic is likely to contain a mix of most words, and not only topic related words.

LDA produces a matrix with that gives for each word the probability that it belongs to each topic. Each document becomes a probability distribution over topics, and each topic a probability distribution over words.

Fig. 5. Intertopic Distance Map via Multidimensional Scaling

The LDA reaches an coherence score of 52% which could be improved by identifying the optimal number of topics either empirically, or by gaining additional information on the data set itself. A better text cleaning would also probably improve this score.

4 Can we cluster the comments by problem type?

This question is related to building an unsupervised clustering method in order to identify potential clusters among our data.

The methodology that has been chosen is the following :

- project the comments into a vector space using Term Frequency - Inverse Document Frequency (TF-IDF)
- apply K-Means clustering
- use T-Distributed Stochastic Neighbouring (t-SNE) dimension reduction to project the clusters on a 2D plane

4.1 Term Frequency - Inverse Document Frequency (TF-IDF)

The TF-IDF is a method that is widely used to project a text corpus onto a vector space. The two main steps are :

- Count each word occurrence by document, and put it in a term frequency matrix.
- Apply the term frequency-inverse document frequency weighting: words that occur frequently within a document but not frequently within the corpus receive a higher weighting as these words are assumed to contain more meaning in relation to the document.

We will be working with the distance matrix, defined as :

$$1 - cosinesimilarity(tf - idfmatrix))$$

Cosine similarity is measured against the TF-IDF matrix and can be used to generate a measure of similarity between each document and the other documents in the corpus.

4.2 K-Means Clustering

K-Means clustering partitions the observations into a chosen number k of clusters in which each observation belongs to the cluster with the nearest mean.

Since we are working in an unsupervised framework, measuring the coherent of the result might be challenging. Figure 6 displays the clusters, made of words and related "friends" created by the K-Means clustering.

The next step will be to create a graphical representation of the clusters. This step requires dimension reduction.

```
Cluster 0 words: pr, could, fields, would, excellent, new,
Cluster 1 words: mean, path, spring, much, resource, branch,
Cluster 2 words: need, update, explain, review, version, pr,
Cluster 3 words: details, branch, much, could, documentation, make,
Cluster 4 words: rebase, update, fields, reasoning, os, path,
Cluster 5 words: constraint, find, probably, documentation, rebase, fields,
Cluster 6 words: least, equal, good, could, change, version,
Cluster 7 words: fields, excellent, path, agreement, tag, suggest,
Cluster 8 words: excellent, much, agreement, documentation, app, master,
Cluster 9 words: thanks, excellent, fields, os, agreement, option,
Cluster 10 words: think, could, sign, documentation, thanks, soon,
Cluster 11 words: lot, support, want, remark, try, email,
```

Fig. 6. List of the clusters created by the K-Means algorithm

4.3 T-Distributed Stochastic Neighbouring (t-SNE)

An efficient (but computationally heavy) algorithm for clustering is the T-Distributed Stochastic Neighbouring Entities (t-SNE). It looks at the original data that is entered into the algorithm and looks at how to best represent this data using less dimensions by matching both distributions. The projection on a 2D plane is then straightforward.

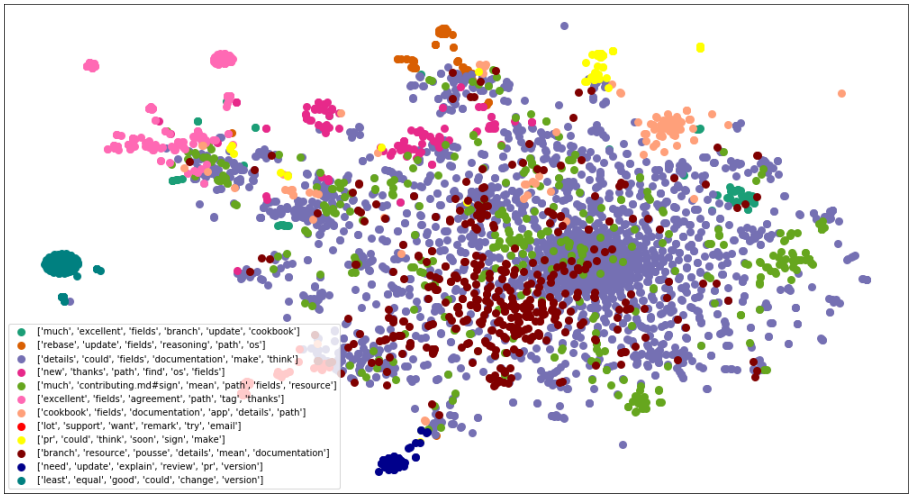

Fig. 7. t-SNE 2D clustering

The results of the t-SNE show us that some clusters have well been identified, whereas the result for others is not so convincing. Additional word cleaning would be required here in order to provide a cleaner result.

5 How long is the resolution time after a change was requested ?

Before moving on to this question, I have tried to bring an answer to the following question : Can we predict from a developer's comment if it will lead to an update of the code ? The accuracy reaches 87% using a K-Nearest Neighbors classifier. This would typically be useful for a team of developers to be able to focus on some specific discussions.

Now in order to predict how much time will be required before a change is made in the code, we can focus :

- on the time at which the comment was posted, as it is highly related with the activity of the developers,
- on whether a code change request is made,
- and on the content of the comment itself.

This is quite challenging since we are now mixing sparse matrices build from the comments with numerical features extracted from dates.

One of the issues that arises when working with that many features is that our data set becomes too wide. We have 4000 observations, and around 5000 columns created by the concatenation of a Bag-Of-Words Count Vectorizer or a TF-IDF sparse matrix with a set of numerical features.

There are several dimension reduction techniques, and the outcome of each algorithm highly depends on the input. The approach that works best in the example we consider is the Latent Semantic Analysis, an algorithm that uses a truncated singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text.

The metric chosen on the regression problem is the R^2. A KNN with 3 neighbors achieves a R^2 score of 75%.

6 Conclusion

The key elements we have covered are that :

- it is possible to extract topics from the comments, although is requires a lot of iterations to identify the right amount of topics to focus on (or additional knowledge on how many github projects have been included here)
- it is possible to build clusters, but the results can be improved
- it is challenging to predict whether a comment will require a code change
- it is possible to forecast if a comment will lead to a merge with a decent accuracy
- it is challenging to estimate the time required before the update of the code is done, since the result is highly influenced by some keywords.

There are several sources of improvements in the work proposed. The main ones being that :

- our model is too sensitive to hyperparameters, but especially to some keywords used that influence the dimension reduction
- a better text cleaning would be needed
- more training samples would allow a more robust model
- more training samples would also allow deep learning approaches to be developed.

Natural Language Processing is a dynamic field, and research is evolving quickly. The approach I chose might be largely improved by applying recent research papers findings.