ENS Data Challenge
Drug-related questions classification by Posos

Project report

December 2nd, 2019

Authors :
Saimourya SURABHI
Maéva CAILLAT

Supervisor :
Bertrand MICHEL

https ://challengedata.ens.fr/

Contents

Abbreviations

CNN	Convolutional Neural Network
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
SVC	Support Vector Classifier
SVM	Support Vector Machine
TF-IDF	Term Frequency–Inverse Document Frequency

Introduction

The objective of this report is to present the Statistical Learning project we have worked on for a month and a half. Indeed, we have participated in the ENS Data Challenge competition, which consists in addressing a machine learning issue provided by large groups, startups and research labs. Additional detailed information about this content are available in [**1**].
In our case, the topic was provided by Posos, a Natural Language Processing (NLP) start-up based in Paris aiming at answering questions about medication from non-structured biomedical sources. The goal of Posos challenge is to predict for each question the associated intent, using NLP. [**2**]

The first part of this report is composed of data description and issues raised by the sets provided by Posos. The second part is about tools used in this project. Then come explanations about the way data is handled, imported and transformed. The fourth part is dedicated to analysing the implemented machine learning models and commenting on the results achieved. Finally, the last part of this report is devoted to overall conclusions and food for thought.
To tackle this challenge, we have tried two methods. Even if one model does not provide convincing results, we will present it in this report. Moreover, we did not only use methods studied in class but also new techniques whose basic principles will be described later on.

Chapter 1

Data description and according issues

1.1 Data description

Data set is composed of three csv files: *input$_t$rain.csv*, *output$_t$rain.csv* and *input$_t$est.csv*, respectively the input and output data of the training set, and the input data of the testing set.The csv files are in UTF-8 encoding.

1.1.1 Input of the training set

The input data of the training set is a list of 8028 drug-related questions written in French. Each line is constituted with a unique ID, varying from 0 to 8027, followed by one question. These ID relate to the ID in the output file. The questions are 1 to 180 words-long sentences whose intent has to be predicted. In the input file, the first line is the header and columns are separated by "," characters.
Here below is an example of the input file content:
6,atrax n'est-il pas dangereux au long terme ?
Which means in English: "6,isn't atrax dangerous in the long term?"

1.1.2 Output of the training set

The output data of the training set contains the intent of each question in the input file. In this way, each line is composed of the same ID as in the input file and an intent encrypted by a number between 1 and 50. The first line is the header and columns are separated by "," characters.
Here is an example of the input file content: 6,48
In this example, the intent of the question "isn't atrax dangerous in the long term?" corresponds to the intent encrypted by 48.

1.1.3 Input of the testing set

The input file of the testing set has the same structure as the input file of the training set, that is to say ID,question. It gathered 2035 questions in total.

1.2 Data issues

There are many issues with this data set. First difficulty is an unequal distribution of questions and intentions in the training and testing sets. Furthermore, questions are full of spelling mistakes and abbreviations. Finally, not every drug name appears in the training set whereas it is an important data to consider.

Chapter 2

Presentation of useful tools

2.1 Versioning with Git

Git is an open source distributed version control system.[3] Here is some explanations about it.

2.1.1 About version control

Version control is a system that records changes to a file or set of files over time. For example, in this project, Git has recorded the changes to Jupyter notebooks.

2.1.2 How does Git work?

Files in a Git repository can have three states: modified, staged and committed:

- Modified means there are changes in the files that are not committed to the database yet.
- Staged means that the system knows there are changes in the files and that the following commit will include these changes.
- Commited means the files are safely stored in the local database of the user.

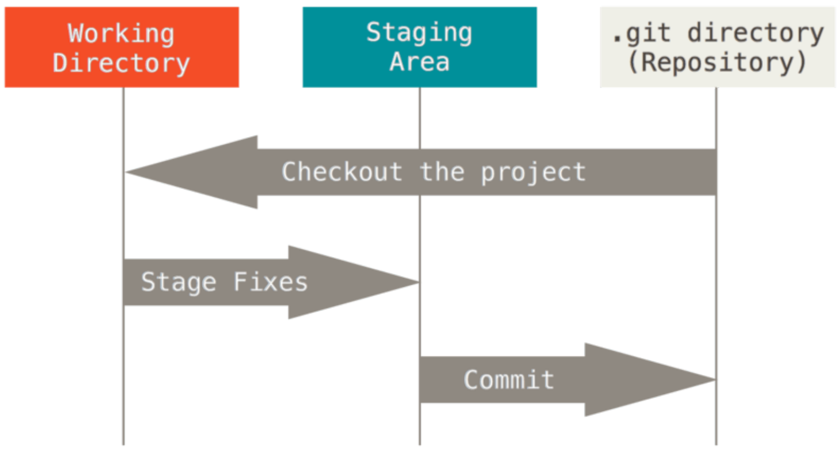
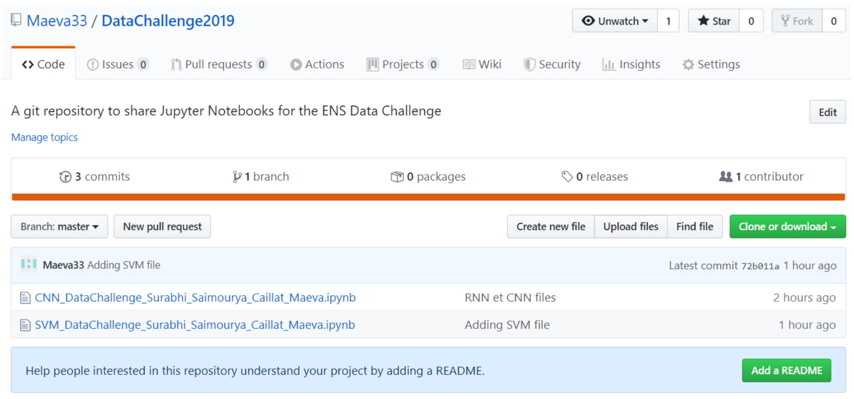


Table 2.1 – The three main sections of a Git project [4]

2.1.3 Using Git in this project

A Git repository was created for this Data Challenge [**5**]. As we worked on two different machine learning models, Convolutional Neural Network (CNN) and Support Vector Machine (SVM), Git helped sharing the numerous changes in the Jupyter notebooks. Here below is an example of the Git repository of the project in its early stages:



A git repository to share Jupyter Notebooks for the ENS Data Challenge

3 commits | 1 branch | 0 packages | 0 releases | 1 contributor

Maeva33 Adding SVM file — Latest commit 72b011a 1 hour ago

CNN_DataChallenge_Surabhi_Saimourya_Caillat_Maeva.ipynb	RNN et CNN files	2 hours ago
SVM_DataChallenge_Surabhi_Saimourya_Caillat_Maeva.ipynb	Adding SVM file	1 hour ago

Help people interested in this repository understand your project by adding a README.

Table 2.2 – Git repository of the project in its early stages

Here is the standard Git procedure followed for this project:

- Click right and "Git bash here" to open the Git command.
- "git pull origin master" to update the current local working branch with all new commits from the corresponding remote branch on GitHub.
- "git add [file]" to add a modified file that is to be committed
- "git commit -a -m [commit name]" to record all added files in version history
- "git push origin master" to upload all local branch commits to GitHub.

2.2 Overleaf

This report was written using Overleaf, an online LaTeX editor. It enables modifying simultaneously a same LaTeX file and does not require any installation.

Chapter 3

Data manipulations and transformations

3.1 An overall picture of the data

First, let's take a look at the training data to see how intents are distributed:

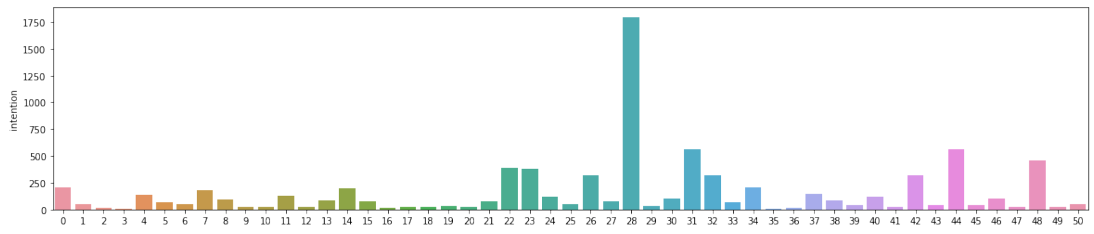


Table 3.1 – Intents distribution in the training set

We can see that some intents are more present than others. For example, intent 28 appears approximately 1700 times whereas intent 3 does not appear.

3.2 Tokenizing and pre-processsing the text using spaCy and NLTK

3.2.1 Decoding UTF-8

The data files are encoded in UTF-8 format so we have decode before preprocessing the data.

3.2.2 spaCy and NLTK

spaCy is an open-source software library for advanced NLP, written in Python [**6**]. Natural Language Toolkit or simply NLTK is a suite of libraries and programs for symbolic and statistical NLP written in Python [**7**].

3.2.3 Removing special characters

The first thing to do is to remove all characters other than words and numbers, for instance ?!|, in the strings using the re package.

3.2.4 Tokenizing strings and lemmatising words

After that, one should tokenize strings and then lemmatise the words so that they can be analysed as single items using spaCy. Tokenization is a way to split texts into tokens. These tokens could be paragraphs, sentences, or individual words [**8**]. CountVectorizer tokenizes the text according to the number of occurrences of each token and a sparse matrix is created. Lemmatisation is a way to reduce the word to root synonym of a word. Lemmatisation returns for example the infinite of the verbs or the words in their singular form entries. In this way, "is" and "am" will be both considered as the word "be", and occurrences of words will appear [**9**].

3.2.5 Removing stop words

The next process consists in removing stop words. In computer search engines, a stop word is a commonly used word (such as "the","is" in English and "le","la","de" etc. in French) that a search engine has been programmed to ignore, both when indexing for searching and when retrieving them as the result of a search query.

3.2.6 Lowercasing words

In order to handle words easily, they have to be lower cased.

3.2.7 train test split

Finally, data is split into both random training set and testing set, containing respectively proportions of 0,7 and 0,3 of the overall data.

Chapter 4

Analysis and results

4.1 Choosing a NLP model for this project

There are various classification methods and we have chosen the Support Vector Machine (SVM) model for this project. We relied on the paper "CNN for sentence classification", from Kim Yoon [**10**], to make such a decision. Indeed, in the table2 "Results of our CNN models against other methods" of this article, SVM has got an accuracy score of 95.0 percent on the TREC question data set, which is the highest accuracy score obtained by a sentence classification model Kim Yoon considered. TREC question data set—task involves classifying a question into 6 question types, which an issue similar to the drug-related intent classification problem of this project. Since the number of features are generally large in text case, the linear kernel generally performs best.

4.2 What is SVM?

A Support Vector Machine is a discriminative classifier defined by a separating hyper-plane. In other words, given labeled training data, the algorithm outputs an optimal hyper-plane which categorizes new examples. In a two-dimensional space, this hyper-plane is a line dividing a plane into two parts, in which classes stand.

4.3 Converting words to vectors

As this is an NLP we have text as iput for train and test data we can't directly use them in the machine learning models we have to convert the words to vectors to use them in machine learning models there are various ways like models like Word2vec, ELMO ,BOW ,we have used Bag-of-words(BOW) model to do in our Code.

4.3.1 Bag of words

A bag-of-words is a representation of text that describes the occurrence of words within a document. It consists of two things: 1.A vocabulary of known words. 2.A measure of the presence of known words. The name "bag" of words, because any information about the order or structure of words in the document is discarded. It is only concerned with whether known words occur in the document, not where in the document.The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document.The scikit-learn library provides 3 different schemes that we can use.

4.3.1.1 CountVectorizer

CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. Python provides an efficient way of handling sparse vectors in the scipy.sparse package.

4.3.1.2 TF-IDF or TfidfTransformer

TF-IDF. This is an acronym than stands for “Term Frequency – Inverse Document” Frequency which are the components of the resulting scores assigned to each word. 1.Term Frequency: This summarizes how often a given word appears within a document. 2.Inverse Document Frequency: This down scales words that appear a lot across documents. How to calculate TF-IDF value? Suppose a review contains 100 words, including the word "Awesome" which appears five times. The term frequency (i.e. TF) for "Awesome" is then $\frac{5}{100} = 0.05$. Again, suppose there are a million reviews in the corpus and the word "Awesome" appears a thousand times. Then, the inverse document frequency (i.e. IDF) is calculated as $\log \frac{1000000}{1000} = 3$. Thus, the TF-IDF value is calculated as: 0.05 x 3 = 0.15 . This is similar to countvectorizer followed by Tdiftransformer in sklearn.

4.4 Multi-class classification

Another challenge here is the multi-class classification. While implementing the SVM model, the OneVs-Rest Classifier concept is used to tackle this issue. The OneVsRest strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negative ones. This strategy requires the base classifiers to produce a real-valued condence score for its decision, rather than just a class label; discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample. To make the classifier design production ready, a pipeline of all the processes discussed above was created making it easier to transpose to other systems.

4.5 Tuning Hyperparameters

Parameters which define the model architecture are referred to as hyperparameters and thus this process of searching for the ideal model architecture is referred to as hyperparameter tuning.we have used GridSearch in our model.

4.5.1 GridSearch

Exhaustive search over specified parameter values for a model.With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model, and selecting the architecture which produces the best results.The parameters of the model used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

4.6 Model

Since the number of features are generally large in text case, the linear kernel generally performs best. we are using LinearSVC estimator for classification. The objective of a LinearSVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyper plane that divides, or categorizes, your data. From there, after getting the hyper plane, you can then feed some features to your classifier to see what the "predicted" class is.

4.7 Fitting and testing the model

The model is trained and then evaluated against the test data set. The accuracy score of the model is 0.66 . We have used the accuracy-score from sklearn.metrics as specified in the rules of the challenge .

4.8 Another model we have tried: CNN

Even if SVM seems to be the best NLP model to use, according to Kim Yoon, we have also implemented a CNN applied at word scale, as recommended by Posos. This is the structure of our model:

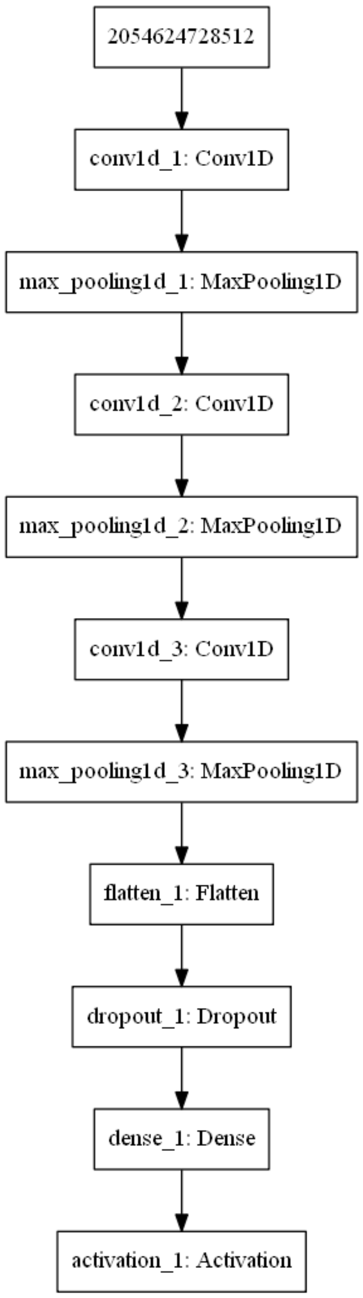


Table 4.1 – CNN model structure

As we knew SVM was way more efficient than CNN, we did not optimize the CNN model so it can explained why the validation accuracy is so low (0.21).

Chapter 5

Conclusion and food for thought

5.1 Learning curve

Learning curve helps us determine cross-validated training and test scores for different training set sizes. A cross-validation generator splits the whole data set k times in training and test data. Then the subsets of the training set with varying sizes will be used to train the model and a score for each training subset size and the test set will be computed. Afterwards, the scores will be averaged over all k runs for each training subset size. It usually refers to a plot of the prediction accuracy/error vs. the training set size (i.e: how better does the model get at predicting the target as you increase the number of instances used to train it). Usually both the training and test/validation performances are plotted together so we can diagnose the bias-variance trade-off (i.e determine if we benefit from adding more training data, and assess the model complexity by controlling regularization or number of features).

5.2 Learning curve for LinearSVC model

We can see clearly that the training score is still around the maximum and the validation score is less in the start and increases at the end, it could be noted that the validation score can be increased with more training samples i.e if the variance is higher.

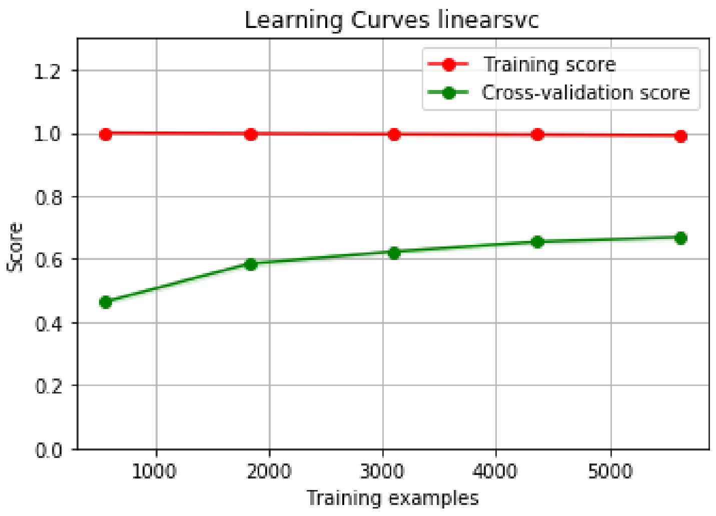


Table 5.1 – Training and cross-validation scores for LinearSVC model

List of Tables

Bibliography

Websites & databases

[1] ENS Data Challenge, *https://challengedata.ens.fr/*

[2] POSOS, *https://www.posos.fr/en*

[3] Git, *https://git-scm.com/*

[4] What is Git?, *https://git-scm.com/book/en/v2/Getting-Started-What-is-Git*

[5] The DataChallenge2019 Git repository created for this project, *https://github.com/Maeva33/DataChallenge2019*

[6] spaCy, *https://spacy.io/*

[7] NLTK, *https://www.nltk.org/*

[8] Word Tokenization with Python NLTK, *https://text-processing.com/demo/tokenize/*

[9] Stemming and lemmatization, *https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html*

[10] Convolutional Neural Networks for Sentence Classification ,*https://arxiv.org/pdf/1408.5882.pdf*

[11] TF-IDF from scratch in python on real world dataset, *https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089*