Projets grand debat

Alexandre Garcia, Alex Lambert

March 2019

1 Introductory Remarks: On the representation of text based data

The SD210 project is a first step towards real life projects you may encounter in your future engineer's life. A common difficulty of these projects is the requirement to choose a representation of the data that will allow you to best answer to a problematic. In practice, most of the work of a data scientist consists in building the mathematical input and output objects by (1) filtering the raw data and (2) making some representation choices. The following section intends to detail some classical preprocessings and representations of the data used in natural language processing, together with their specificities.

1.1 Preprocessing - Tokenization

The tokenization process identifies a broad class of normalization methods used to define the elementary unit of a text. It usually includes the lower casing (replace capital letters by lower case), number encoding (replaced by a token ⟨NUMBER⟩), punctuation encoding (⟨PUNC⟩) or removing, and many others.

1.2 Preprocessing - Stemming

The stemming (in french "racinisation") consists in replacing a word by its root. It is typically a dimensionality reduction method that projects the different grammatical and semantic variants of a word on its root (typically 'fishing', 'fished' , 'fish' and 'fisher' are projected on the same root 'fish' when stemming). Stemmed data are more robust to orthographic faults (on faults appearing after the root) and reduce the imbalance of the data by projecting rare and unique variants of the words on the same representation. A drawback is that sometimes the suffix carries some information which is lost in the stemming process.

1.3 Preprocessing - Stop Words removal

Stop Words are words that have a grammatical function but usually do not carry some semantic information. They are thus often removed in a preprocessing of

the data. Some stop words lists can be found for each language and completed with your own stop words depending on the task at hand.

1.4 Preprocessing - Ressources

The preprocessings are generally performed using libraries such as nltk (a set of examples and ressources using nltk for french can be found here :

`https://github.com/cmchurch/nltk_french/blob/master/french-nltk.ipynb`

Some pretrained ressources include these preprocessing in their own pipeline. Such libraries include the Stanford Core NLP tool : `https://stanfordnlp.github.io/CoreNLP/index.html` (available functions : part-of-speech (POS) tagger, named entity recognizer (NER), parser, coreference resolution system, sentiment analysis, open information extraction tools)

SpaCy `https://spacy.io/models/fr` includes the same type of function with a different toolkit.

You can also find some state of the art dense word embeddings dedicated to french. (Recent state of the art models rely in fact on multilingual representation and french is one available among them, see here for the BERT multilingual embeddings `https://github.com/google-research/bert/blob/master/multilingual.md`. Dense word encoders also usually come with their own preprocessing pipeline.

1.5 Embeddings - Bag of words

The bag of words representation of a text is the simplest model available. It consist in representing each token (see 1.1) by a one hot vector (of the size of the entire token vocabulary). Then a text is represented by the sum of its token one hot vector. (see OneHotEncoder in sklearn)

1.6 Embeddings - tf-idf

The previous representation does not take into account the 'importance' of each token. We expect that a token that appears only in a few texts carries more information than a token appearing in all the texts. The goal of the tf-idf weighting scheme is to reduce the weights of frequent tokens and to increase the ones of rare tokens (see tfidfVectorizer in sklearn)

The two embeddings described above rely on a sparse representation of the data. A consequence of this type of representation is that the distance between two tokens in the feature space does not carry any information. The representations described below represent each token in a low dimensional space (low with respect to the vocabulary space) with a common specificity: that the distance between the token representations is an indicator of their semantic similarity.

1.7 Embeddings - dense word embeddings (word2vec, Glove, BERT, Elmo, ...

Contrarily to the previous embeddings, dense representations carry a semantic information. A particular example of this information appears in the semantic arithmetic introduced in the word2vec article: Embedding(Queen) = Embedding(King)-Embedding(Man) + Embedding(Woman). The common feature of dense representations is that they are derived from a language model learned on a big dataset. It is well known that these dense embeddings provide a good representation when building a token level supervised model. Some variants also exist to build dense representations at the sentence level (sent2vec) or at the text level (doc2vec).

1.8 Implicit feature representation, the kernel based / graph based approaches

In some cases, you are not interested in building a feature representation (because you don't know how to correctly represent your data) but you're more interested in injecting some similarity knowledge about different samples. This similarity knowledge can be encoded under the form of a gram matrix (i.e. a $N \times N$ matrix such that the i, j entry contains an indicator of the similarity of elements i and j of the dataset) or as the adjacency matrix of a graph (where texts i and j are connected together if their similarity is above a certain level). This type of graph can be built with some handcrafted rules (if two texts contain some positive sentiment AND they contain at least a certain number of common tokens, then they are connected together). In the case of gram matrix, they can be directly used in kernel based methods (binary / multiclass SVM, One class SVM for anomaly detection, kernel ridge regression, etc). In the case of graph based data, they can be used for visualisation or spectral clustering (see pointers in project T1 below).

2 Project T1: Clustering and dimensionality reduction, representation learning

The goal of this project is to analyze the contents of contributions by testing different metrics or data representation. Dimension reduction and representation learning (Autoencoder, Multidimensional scaling) allow you to to discard redundant information in the data by compressing it in a small dimensional space. It provides i) a order to improve the performance of the predictors we learn. A second aspect of low dimensional representation lies in the interpretability of the extracted patterns. In this project your goal is to provide interpretable insights based on a big text-based dataset. In order to guide your work we propose the following milestones that will guarantee the scientific validity of your analysis.

1. State clearly your goal at the beginning. A good work has to be moti-

vated by a clear question. Examples of such questions include " - What are the most common propositions in the debate? - Can we summarize the energetic transition arguments? Which propositions are the most controversial? ". Note that in each of these questions, the scientific procedure to answer it is unclear and will have to be chosen and justified.

2. Propose a scientific approach to answer this question. It can be based on a pipeline including - a filtering the data following some criterions that meet your problematic (specific vocabulary, text length, ...), - a cluster based projection in some well chosen space, - a topic model, ...
3. Implement your approach
4. DISPLAY and DISCUSS empirical results
5. Provide an evidence based answer to your problematic.

Keep in mind that this project is close to a real data scientist use case: We expect from you a justification of your choices as well as an empirical evidence for every conclusion you draw.

Technical references for the T1 project:
Dimensionality reduction for text representation
Ressources on clustering:
General overview of clustering methods: [3]
It is possible to inject some information that biases your models in such a way that they will focus on the type of clusters you are expected to find: [5]
Clustering based on a graph representation (typically imagine that each text of the corpus is a node in a graph and there is an edge between two nodes if the corresponding texts are 'similar enough'. Then spectral clustering provides a mean to find (relatively efficiently) the subparts of the graphs with a strong connection level corresponding to clusters of similar texts: [4]

3 Project T2: "Explaining" clusters by supervised learning

This project shares some similarity with the first thematics (T1), although a main difference appears: whereas the project T1 focuses on **interpreting** the results of a dimensionality reduction / clustering method, project (T2) focuses on **understanding** these results. One approach to understand the results of generative modelling is typically to use supervised retro-fitting (i.e. choose a target such as the elements of a cluster) and try to predict this target using a supervised approach.We advice you to use decision trees for that supervised task. Once again your work will have to be motivated by a problematic. See here for more details concerning the interpretability of model based predictions `https://christophm.github.io/interpretable-ml-book/tree.html`

4 Project T3: Supervised learning for location, opinion, or any available specificity prediction from the text content

This project is more technical than the two previous ones. It relies on the availability of data easily convertible to mathematical targets (such as yes/no questions, one choice among many questions). Thus a problematic can be formulated with the goal of understanding whether this target can be predicted using the rest of the data, typically can we build a predictive model of an opinion (obtained from a yes / no questions) using the propositions detailed in the open questions? This type of project will mainly focus on the target construction choices and the representation choices for the input data to obtain good performances.

5 Project T4: Topic modelling

This project shares the same structure as project 1 with a difference in the type of method used: you are expected to use models that rely on a hierarchical description of the token observed. Under the topic model, a text contains a (restricted) set of topics and each topic corresponds to a distribution probability over each token. The most classical hierarchical model which follows this structure is the Latent Dirichlet Allocation [2] (available in sklearn or gensim for better scalability). Other variants of topic models can be built using Non Negative matrix factorization (see sklearn examples) or Latent Semantic analysis (also relying on a low rank matrix factorization of the data). One of the possible goal for this project would be the conception, parameterization and interpretation of a pyldavis based application to dig into the data : `https://nbviewer.jupyter.org/github/bmabey/pyLDAvis/blob/master/notebooks/pyLDAvis_overview.ipynb`.

6 Project T5: Generative modelling

In this project the goal is to build a model able to generate some new sentences coming from a distribution similar to the one of the data. Building language generating models is a difficult task for which we propose some good practices and try to warn you about dangerous directions.

- The first step consists in building a neural language model based on some classical models. You will find available language models relying on the LSTM model and the GRU model. GRU is known to be a bit easier to train than LSTM (less sensible to the parameters of the gate). RNN based layers will not be able to predict long sentences. You can reuse an existing code and try to understand how to train the model and generate a sentence based on a random initialization vector.

- Once you have a generative language model, you can try to build a conditional generative model. The goal is to generate some sentences with a control on them. In the simplest case, this control is injected by choosing carefuly the initialization state of the generative language model. This technique is used in the seq2seq models (common in machine translation, summarization) or in image captioning where one object is fed to neural network that output a fixed size vector representation used as the initialization hidden state of a language model. You could typically choose as the initialization step a representation of the question or a representation of the words that you would like to find in the generated sentence.

- We do not encourage you to work with GANs (Generative Adversarial Networks) despite the beautiful generated pictures you will be available to find on internet, GANs are hard to train, even harder on text and they do not really provide better models.

An example of simple GRU based text generator can be found here: `https://github.com/Arfua-zz/Text-generation-with-GRU`. Then instead of simply generating sentences, one can generate them by controling its semantic `https://github.com/hit-computer/SC-LSTM` (note that in this example, LSTMs can easily be replaced by GRU layers). Other methods consisting in generating sentences based on a meaning vector (such as in machine translation or summarization) are simply variants of this. `https://github.com/tensorflow/nmt`

7 Checklist for your project

When working on your project you must answer the following questions and present the answers in your final report.

- **Task definition:** say clearly what task you solve and which **evaluation metric** will be useful to measure performance of your model.

- **Dataset description** volume, input/output, potential bias, pre-processing

- **Data representation:** features, kernels ...

- **Model Description:** which family of functions are you using (for instance in clustering, say which metric you use for comparing data)

- **Learning problem:** (surrogate) loss, penalties and constraints, discuss clearly what are the ingredients of the learning problem

- **Learning/Optimization algorithm and its implementation**: local/global convergence, complexity in time

- **Validation:** cross-validation, separation train/validation/test, ...

8 Conclusion

"I always thought something was fundamentally wrong with the universe" [1]

References

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy*. San Val, 1995.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[4] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[5] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.