Université de Paris

M1 LI

Linguistic Probes with ℓ_1-Regularization

Guillaume Wisniewski
guillaume.wisniewski@u-paris.fr

November 2021

1 Context: Linguistic Probes

The very strong performance neural networks achieve in many NLP tasks is arguably due to their ability to automatically build very good representations of words and sentences: features are no longer extracted manually beforehand; the model is rather able to automatically find the best features for the task. For example in `Transformers`, the current state of the art architecture, a neural network starts by representing each word of a sentence by a vector of $\mathbb{R}^{768}$.[1]

These representations are learned automatically in a completely unsupervised way: the neural networks are estimated by masking randomly some words in a sentence an finding the parameters that allow the neural network to 'fill in the gap'. Many works have analyzed these representation to uncover the information that are encoded in it (see, for instance, [3] for an overview) and see if this information is consistent with linguistic theories. *Linguistic probes* are one of the method used for these analyzes: a *probe* [1] is trained to predict linguistic properties from the representation of language; achieving high accuracy at this task implies these properties were encoded in the representation.

In this lab, we will train a linguistic probe to see if the information necessary to predict the agreement of the past participle with the direct object in French direct relative is encoded in the representation of the direct object. Agreement tasks consist in asking a neural network to predict the correct form of a word (generally a verb) in a sentence such as:

(1) Iel nous a écouté/écoutés

(2) Iel nous a parlé/parlés

(3) Ces fruits sont délicieux ; je les ai acheté/achetés

(4) Les mots que J.-P. Sartre a écrit/écrits à S. de Beauvoir ont été publié/publiés sous le titre *Lettres au Castor*

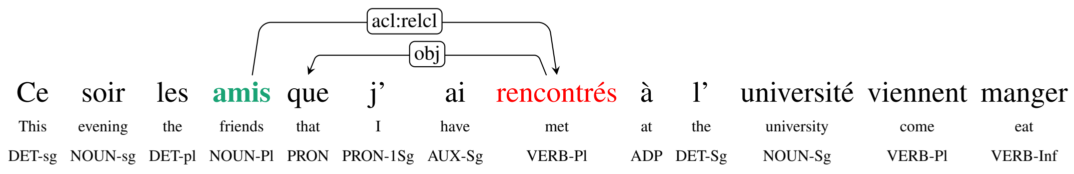
[1]The size of the feature vector depends on the task and on the language considered.

Note that, in these examples, we only consider agreement in number and not agreement in gender. The ability of neural networks to solve agreement tasks is, today, one of the main proofs that they capture syntactic information (i.e. hierarchical structures).

1. Give the correct form of the past participle in the previous examples.
2. Why examples 3-4 are more interesting to study than examples 1–2 to assess the capacity of NN to capture syntactic information?
3. Why do we not consider agreement in gender?

We will consider the experimental setting of [2] that has gathered sentences such as the one represented in Figure. In practice, we will extract the representation of the *que* token and use is in a linear regression model to predict whether the past participle must be in its singular form or in its plural form. More precisely, we will address two questions:

- is the number information present?
- how many features are required to predict this information?



2 Setting up a Virtual Environment for Python

Virtual environments are a simple solution to solve three important issues faced when working on several Python project:

- it allows any user to install all the libraries even when they do not have the privileges to install program and with no impact on other users that could use the same computer (i.e. to perform a *local installation*);
- it allows to resolve requirements conflict (when project A needs version 1.0 of a specific library but project B needs version 2.0);
- it allows to keep track of all the libraries (and other dependencies) required to make the code work.

Different projects or applications can use different virtual environment and, in general, it is a good practice to create one virtual environment for each project you are working on.

- **In the following, we assume that you are in a directory containing all the code and the data required for the lab.**
- **If you are working on one of the computer of room 309, the following instructions *will not work*. Please refer to instructions in Section 5**

To create a virtual environment, decide upon a directory where you want to place it, and run the `venv` module as a script with the directory path:

```
€ python3 -m venv local
```

where `local` is the name of the directory in which the environment will be created. You can, of course, choose another name. This command creates several sub-directories within the `local` directory and notably:

- a `bin` directory that contains several executables and in particular a `python` command;
- a `lib` directory that contains all libraries installed in *this* environment.

Once the virtual environment has been created you can easily install new programs and library in it. The following commands will install all the library required for this lab:

```
€ ./local/bin/pip install halo sklearn bloscpack
```

Note that will explicitly specify that we want to run the `pip` command that is attached to the virtual environment stored in `local`. This is fundamentally different than running the command:

```
€ pip install halo sklearn bloscpack
```

This command will call the *system-wide* `pip` command that will try to install the requested programs so that they are available to any user using this computer. This is a very bad idea and the OS will usually forbid you to run such a command.

You can now use this virtual environment with the command:

```
€ ./local/bin/python
```

As before, you have to explicitly specify the path to the python interpreter to be sure to run the command associated to a specific virtual environment and have access to all installed libraries.

4. Which executables have been installed in this virtual environment?
5. What is the `pip` command doing?
6. Why is running the `python` command (without specifying any path) and than the `import bloscpack` command will result in an error (and which exception is raised)?

3 Is number information encoded into token representation?

To load the corpus, you can use the following two commands:

```python
import bloscpack

X = bloscpack.unpack_ndarray_from_file(X_file)
y = bloscpack.unpack_ndarray_from_file(y_file)
```

where `X_file` and `y_file` are a `string` with the path to the two files that can be found on the lecture website.

Observations, stored in the `X` variable, are represented by a `numpy.array` that is to say a matrix that has as many rows as there are examples in the corpus and as many columns as there are features. The `y` variable contains is a column vector, the *i*-th row of which correspond to the label of the *i*-examples (i.e. the *i*-th row of `X`). This is exactly the format expected by `sklearn`.

7. how many features and examples are there in the corpus? (hint: look for the size of `X`)
8. using `sklearn`, split the corpus into a training set containing 80% of the examples and a test set containing the remaining examples.

A logistic regression model can be trained with the following code snippet:

```python
from sklearn.linear_model import LogisticRegression
from halo import Halo

with Halo(text="Training", spinner='dots') as spinner:
  clf = LogisticRegression()
  clf.fit(X_train, y_train)
spinner.succeed("Training")

print(clf.score(X_test, y_test))
```

The `halo` library provide a simple way to see that the program is running and using it is purely cosmetic.

Using a classifier with `sklearn` involves three steps

1. create a new instance of the classifier by calling the constructor. Calling the constructor allows to fix the value of the *hyper-parameters*. There are two kinds of hyper-parameters:
   - hyper-parameters that control the definition of the class of function considered. For logistic regression, this kind of hyper-parameters includes `penalty` and `C`;
   - hyper-parameters that control the way the objective function will be optimized during training. For logistic regression, this kind of hyper-parameters includes `solver` and `tol`.

2. call the `fit` method to find the optimal value of the parameters. The `fit` method will always take a matrix of observations and a vector of labels as parameters.

3. call the `score` or `predict` method to either evaluate the classifier performance of a (labeled) dataset or predict labels of new observations.

These steps are always the same, whatever classifier is considered. Note that hyper-parameters *can not* be optimized during training (they have to be defined before fitting the function) and their value must be chosen with a specific procedure (see below).

9. Considering value of C defined on a log-scale,[2], plot the accuracy achieved by the classifier when considering either ℓ_1 regularization or ℓ_2 regularization. Comment.
   Hint: a convenient way to store and manipulate the results of such experiments is to use a `DataFrame` (a convenient way to represent tabular data — like Google Sheet — provided by the `pandas` library[3]).

10. Why C can not be found in the same way other parameters are found? How could you choose the optimal value of C?

4 Where is number information encoded?

In this Section, we will try to determine whether number information is distributed all over the feature vector or only encoded in a few dimensions using ℓ_1 regularization.

11. Considering different regularization strength, plot the classifier accuracy with respect to the number of non-zero parameters in the classifier (parameters of the classifier are stored in the `coef_` attribute) and plot the number of non-zero coefficient with respect to the value of C. Interpret.

The results of the experiments described in the previous question do not allow us to conclude anything about the localization of the number information within the feature representation. Indeed, it is possible that this information is also encoded into features that were not selected as they were highly correlated or redundant with selected features.

12. Construct a new matrix of observations that only contains the features that have not been selected by the ℓ_1 regularization
    Hint: Selecting row or columns from a `numpy` array can be done conveniently using *masking*. See here for an introduction.

13. Train and evaluate a new classifier considering only this reduced set of features. Conclude.

14. Repeat the two previous step as many time as possible. Conclude.

[2]for instance, C can take the values in `[10 ** i for i in range(-5, 6)]`.
[3]An short introduction to the `pandas` library is available on the lecture website

5 Creating a virtual environment — the hard way

For a reason that nobody wants to know, the usual recipe to create a virtual environment is not working on the computer of room 309. You have to use the following instruction to create a virtual environment:

```
€ python -m venv local --without-pip
€ wget https://bootstrap.pypa.io/pip/3.5/get-pip.py
€ ./local/bin/python get-pip.py
```

References

[1] Guillaume Alain and Yoshua Bengio. “Understanding intermediate layers using linear classifier probes”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=HJ4-rAVtl.

[2] Bingzhi Li, Guillaume Wisniewski, and Benoit Crabbé. “Are Transformers a Modern Version of ELIZA? Observations on French Object Verb Agreement”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4599–4610. URL: https://aclanthology.org/2021.emnlp-main.377.

[3] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. “A Primer in BERTology: What We Know About How BERT Works”. In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 842–866. DOI: 10.1162/tacl_a_00349. URL: https://aclanthology.org/2020.tacl-1.54.