Data Collection & Experiments

Text Analytics - Andrea Esuli

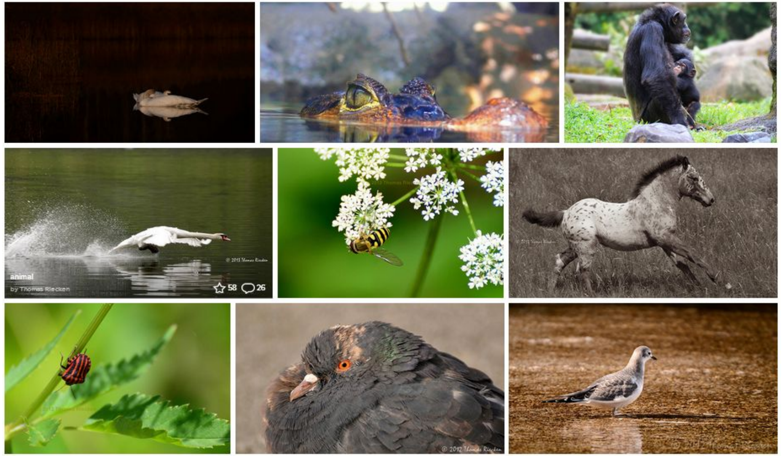
Data collection

Garbage In - Garbage Out

The **quality** of the **output** of any supervised learning process **is limited by** the **quality** of the supervised information fed in **input**.

$\Phi(i)$ = cat

$\Phi(i)$ = not cat



Garbage In - Garbage Out

The **quality** of the **output** of any supervised learning process **is limited by** the **quality** of the supervised information fed in **input**.

$\Phi(i)$ = cat

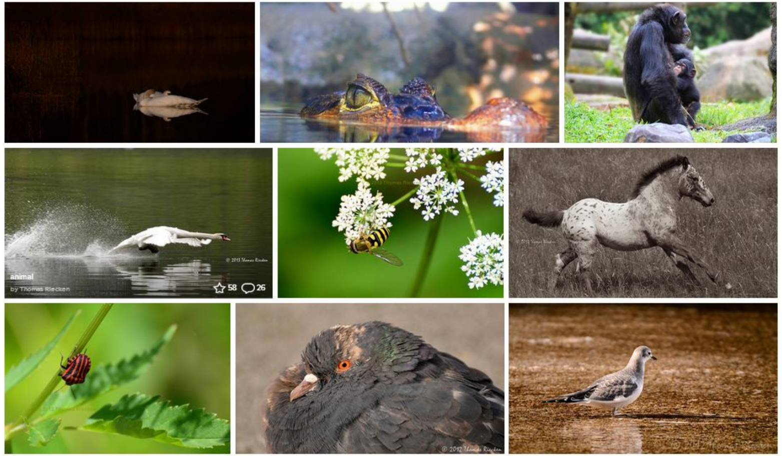
$\Phi(i)$ = not cat

Garbage In - Garbage Out

The **quality** of the **output** of any supervised learning process **is limited by** the **quality** of the supervised information fed in **input**.

$\Phi(i)$ = cat

$\Phi(i)$ = not cat

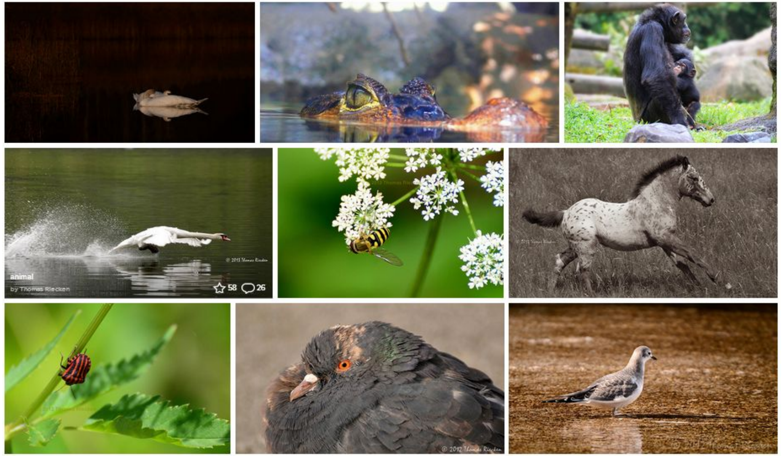


Garbage In - Garbage Out

The **quality** of the **output** of any supervised learning process **is limited by** the **quality** of the supervised information fed in **input**.

$\Phi(i)$ = cat

$\Phi(i)$ = not cat



Garbage In - Garbage Out

The **quality** of the **output** of any supervised learning process **is limited by** the **quality** of the supervised information fed in **input**.

- How to the get input data of good quality?
- How to measure the quality of input?

Corollary: output will be likely worse than input.

- How to determine the best quality of output we can expect?
- How to measure the quality of output?

Data Collection

Data collection is a crucial step of the process, since it determines the knowledge base on which any successive process will work on.

- Find the source/sources
- Set up the data collection method
- Get the data
- Prepare the data for successive processing.

Data Collection

Depending on problems and goals, there are many possible data sources.

Web based:

- Online survey, e.g., SurveyMonkey, Google survey, Google forms.
  - Survey services offer demographic targeting
- Web feeds, e.g., RSS, Atom.
  - Most news companies offer a RSS version of their content organized by topic.
- Social networks' APIs. E.g., twitter, facebook, and many other.
- Archives. E.g., Reddit, archive.org.
- Custom web crawling and scraping. E.g., Scrapy.

Data Collection

Companies may accumulate information from other sources:

- feedback channels (email, telephone, sms, handwritten)
- note in customer profiles

...or more traditional questionnaires and interviews:

- Computer-assisted telephone interviewing (CATI),
- Automated Computer Telephone Interviewing (ACTI)

Building a training set

A training set is composed by samples of documents correctly annotated with respect to the goal of the task.

A few sources already provide annotations, e.g., product reviews.

- These are the typical scenarios tested in research because they avoid the cost of data annotation.

Most practical applications obviously come without annotations, e.g., real time filtering of a stream of tweets with respect to a topic of relevance.

- Domain experts are required to annotate the data
- Semi/distant supervision may produce some automatic annotations

Building a training set

Whenever possible, the annotation should be performed by more than one annotator.

- Annotators work **together** on an initial set of documents, to agree/align on how to annotate documents.
- Annotators work **separately** on a **shared** set of documents, to make possible to measure the **inter-annotator agreement.**
- Each annotator works a **distinct** set of documents, to increase the **coverage** of the training set (i.e., a larger number of different documents is annotated)

Inter-annotator agreement

Given a set of documents independently annotated by two or more annotators, it is possible to measure the agreement between annotators.

- Considering in turn the annotations of one annotator as the correct ones
- Then considering those produced by another annotator as predictions and evaluating its accuracy/recall/precision/f1/...

It will be hard for a ML predictor to score a level of accuracy better than the one measured between humans.

Inter-annotator agreement defines a good **upper bound** on the achievable accuracy.

- Yet, super-human performance happen [1] [2] [3] [4]

Experiments

Training-Validation-Test

When running an experimental activity annotated data is usually split in two/three parts:

- A training set, which is the actual data on which the ML algo is **trained**.
- A validation set, which is **held out** data used for **optimization**
    - The validation set is often not explicitly identified as it is up to the research to choose to use it or not.
- A test set, which is the data on which the **optimized model** is **evaluated**.

<u>Information from test set must be NEVER used in training data or in any decision regarding the definition of the training process.</u>

There are many ways to actually perform the split.

Single fixed split

Data is split once and for all in a single training set and a single test set.

Pros:

- easy to reproduce
- reasonable to do on time-related data (training data comes before test data)
- experiments are quick to run

Cons:

- risk of overfitting test data on the long run
- risk of low statistical relevance (test set must be large)

K-fold validation

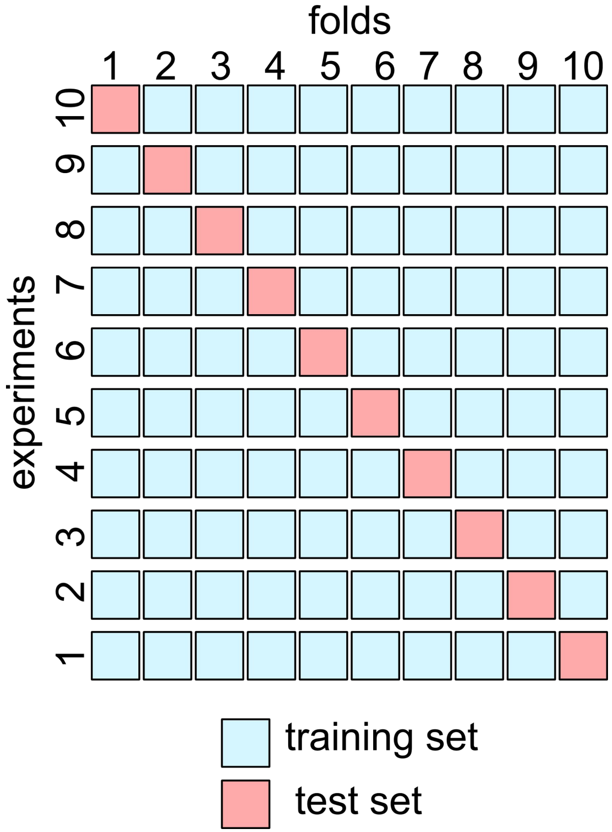
Data is split in k equal sized sets.
For k times, k-1 sets are used as the training set and the remaining one as the set set.

Pros:

- improved statistical relevance (the whole dataset is a test set)

Cons

- reproducible by knowing how splits are made
- must check fold composition
- cost of experiment grows linearly with k



Leave-one-out validation

This is an extreme setup of k-fold validation in which k is set to be equal to the dataset size. Test set for each fold is just one document.

Pros:

- really easy to reproduce
- good statistical relevance

Cons:

- very high cost

Random splits

A split proportion is determined, e.g., 80%/20%. For an arbitrary number of times a random train/test split is created and the accuracy measures are recorded.

Pros:

- high statistical relevance
- cost is flexible, can run it until you have resources

Cons:

- hard to reproduce exactly
- requires statistical analysis to put results together

Optimization of parameters

Experimental setups may have many parameters that must be set and that can have an impact on the quality of results:

- Which features to extract?
- What lexicons to use, how?
- Use of tagging, parsing. How to use it?
- Feature selection: measures and amount
- Weighting functions
- Learner and its parameters

Optimization of parameters

Optimization is made against a specific evaluation measure.

A grid search on all the candidate values of all the parameters can produce an explosion in combinations.

For example:

- 5 feature types, testing each feature independently, all together, and all possible pairs.
- 5 feature selection levels
- 10 values for the C parameter of SVM

produce a total of $(5_{single} + 1_{all} + 10_{pairs}) \cdot 5 \cdot 10 = 800$ configurations to be tested

Optimization of parameters

Parameters with loose correlation can be optimized in sequence.

- First optimize feature selection amount the optimize C value for SVM

Parameters with lots of possible values can be optimized in two step: coarse search, and refinement.

$$k_{NN} \in \{1, 5, 10, 15, 20, 25, 30, 35, 40\} \rightarrow \{6, 7, 8, 9, 11, 12, 13, 14\}$$

For some numeric parameters a logarithmic search scale is fine.

$$C_{SVM} \in \{0.001, 0.01, 0.1, 1, 10, 10, 1000\}$$

Optimization of parameters

Once a grid of configuration for experiments is defined,

- all the experiments can be run exhaustively, or...
- configurations are randomly sampled from the grid, and the relative experiment is executed, until a given experiment budget is consumed.

Sklearn has implementations of both methods.

Beware of Machine Learning Gremlins!