# Natural Language Processing Applications

Lecture 10: Building NLP Applications

fit@hcmus
KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

- Sample Applications
- Notes in Building NLP Applications

fit@hcmus

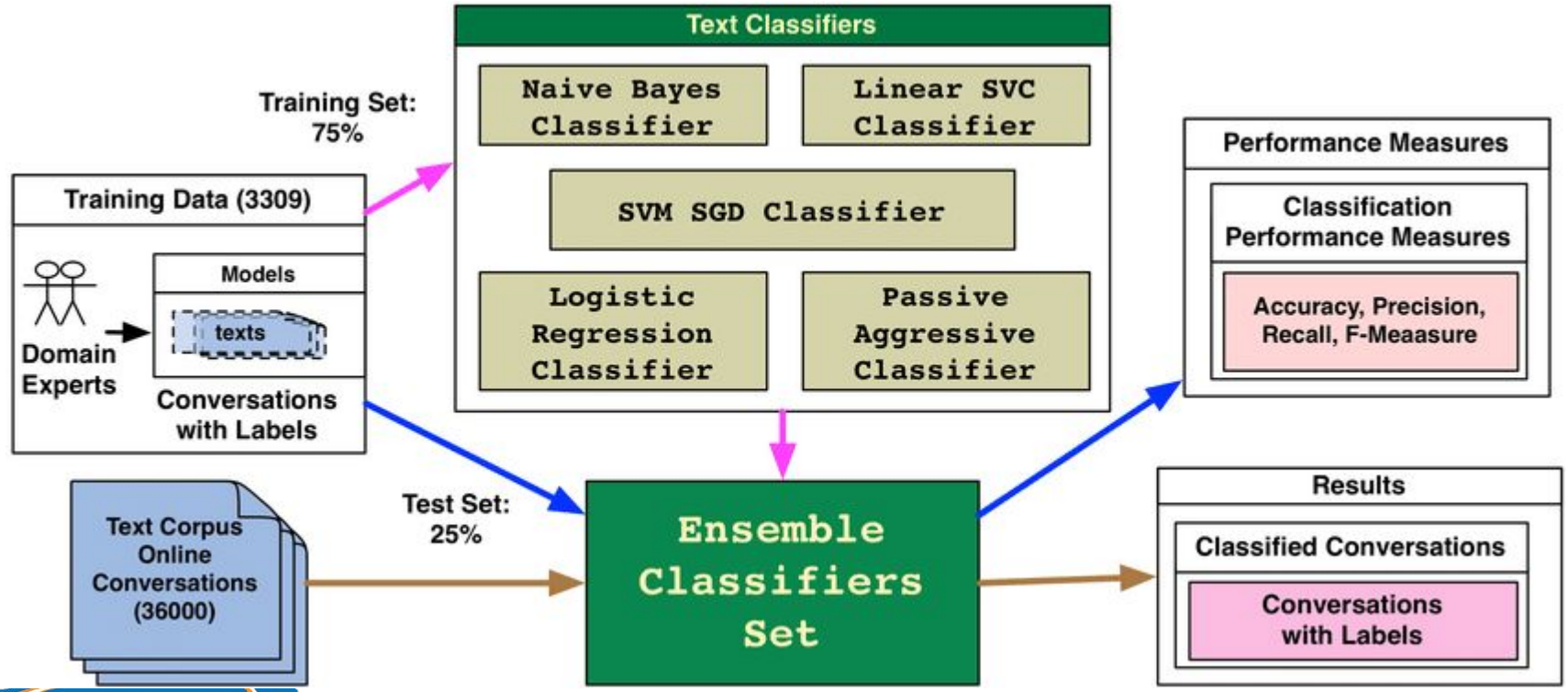
NLPA - Building NLP Applications

# SAMPLE APPLICATIONS

fit@hcmus

# Sample Applications

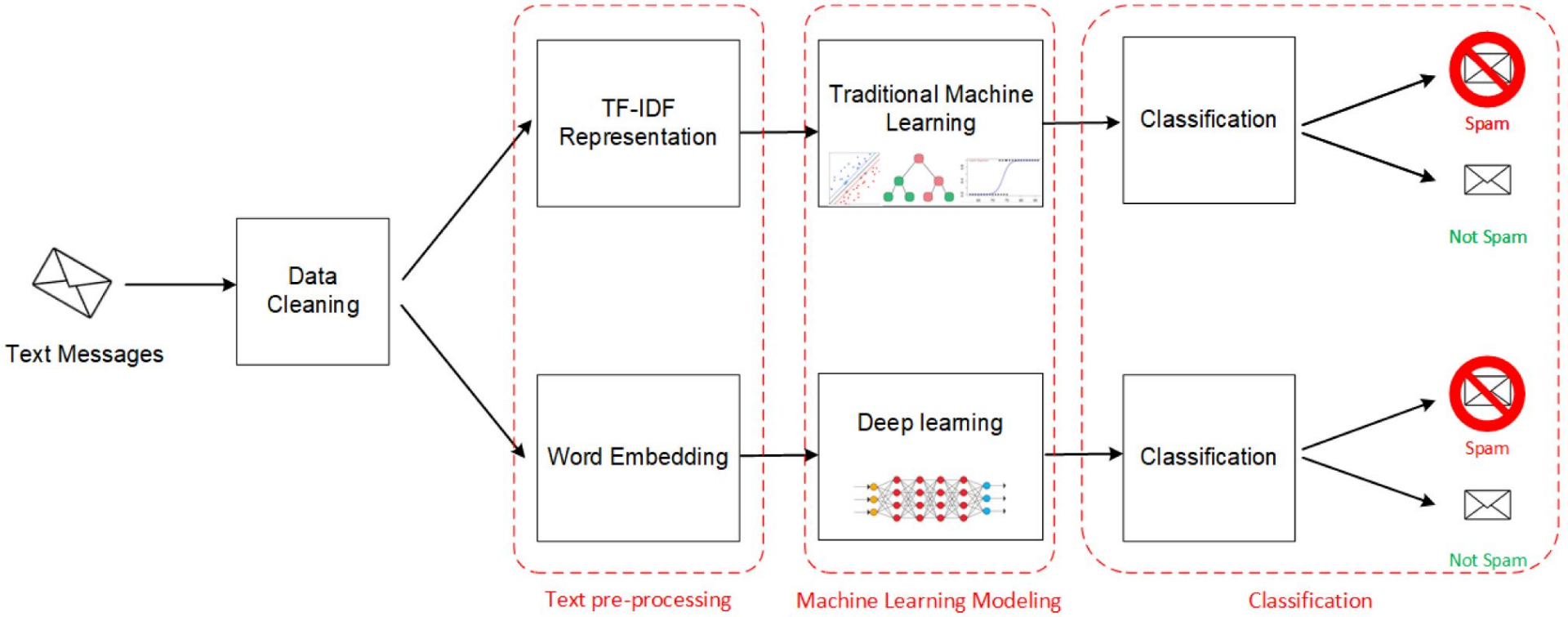
- Text classification:



fit@hcmus

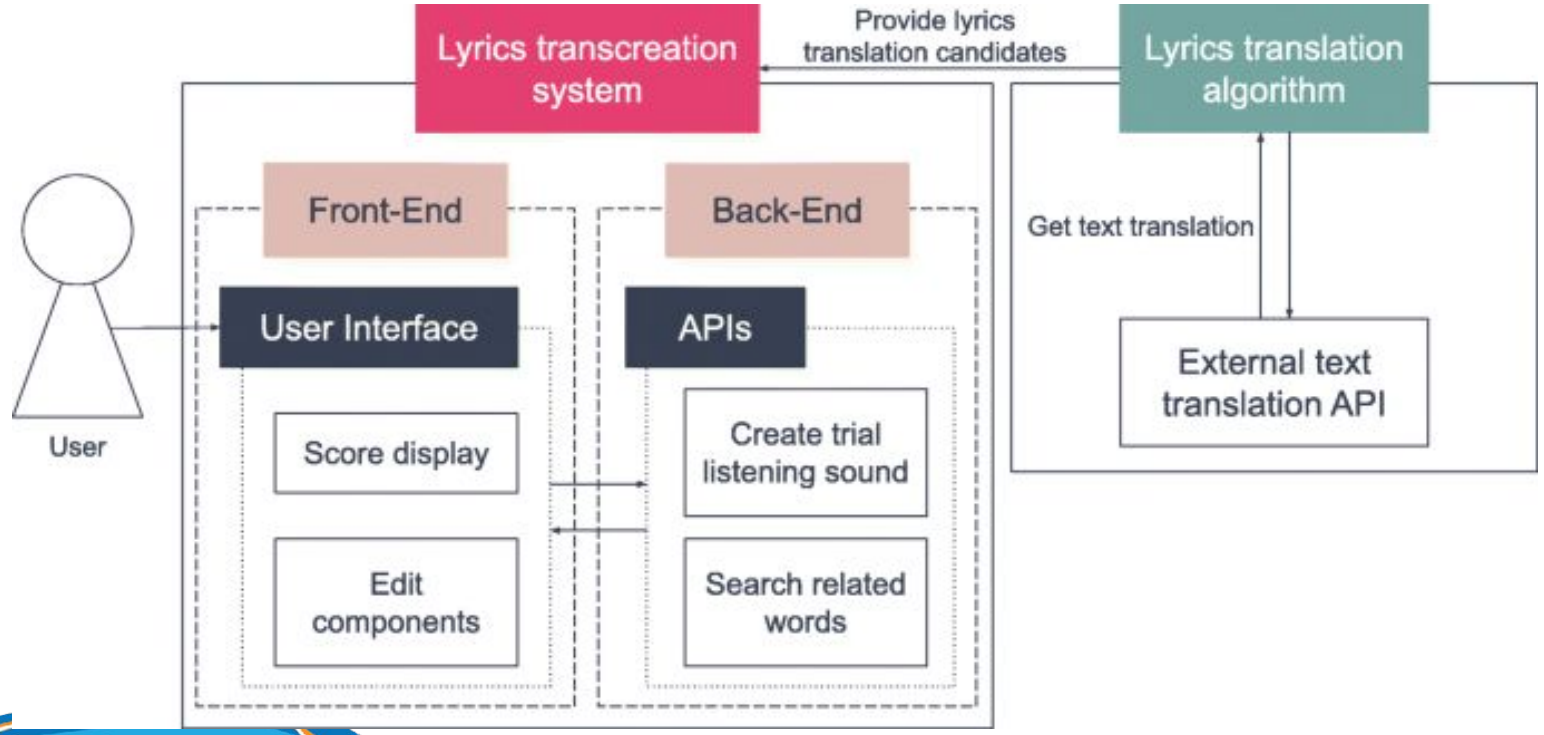
# Sample Applications

- Email classification:



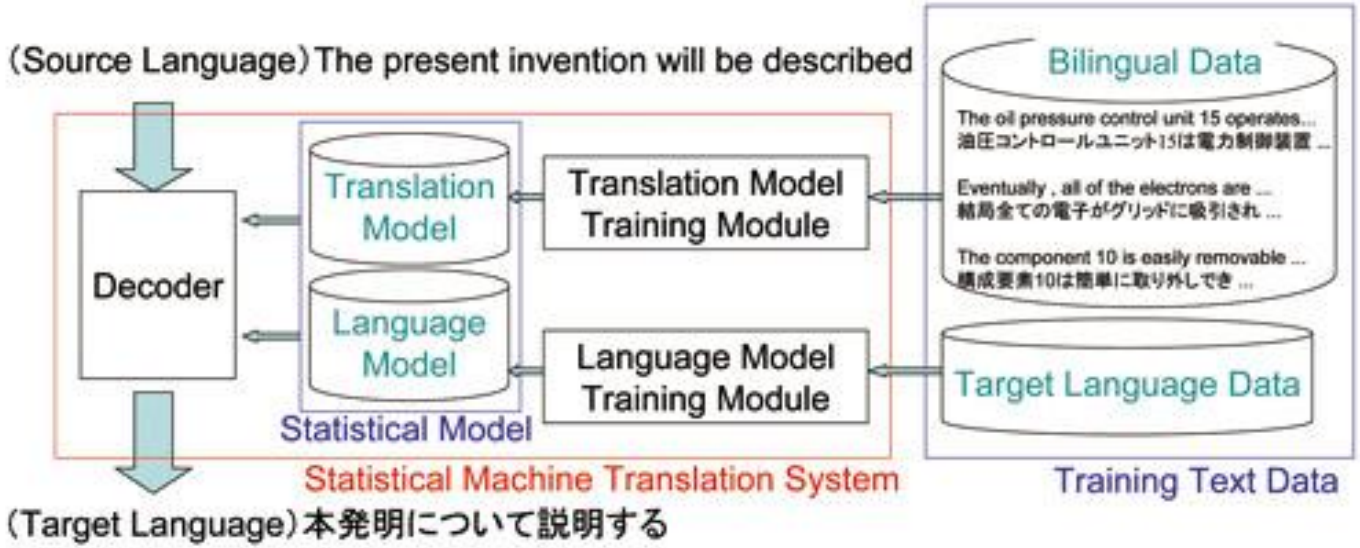
# Sample Applications

- Machine Translation:



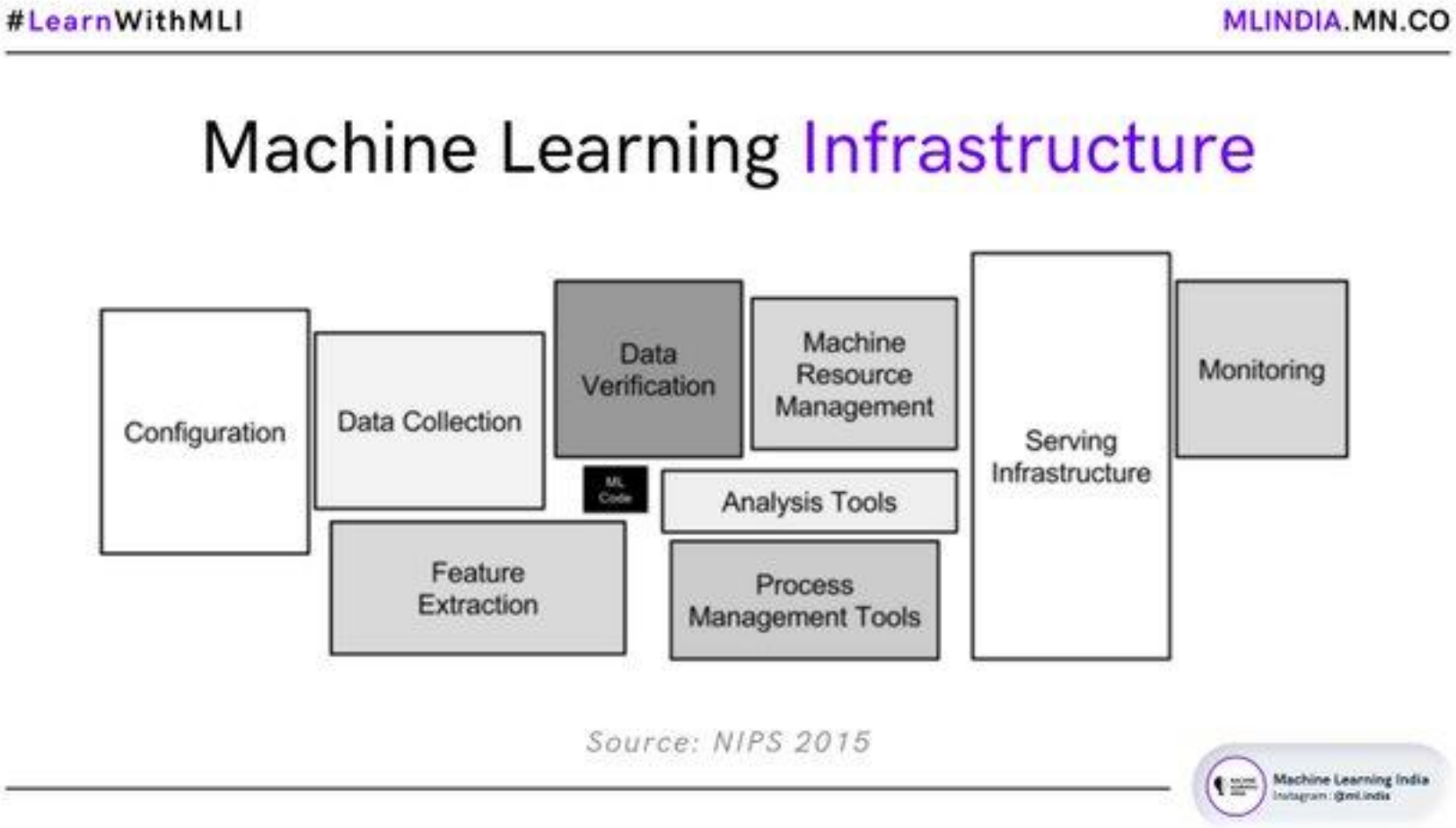
# Sample Applications

- Machine Translation:



fit@hcmus

# Sample Applications



fit@hcmus

# Sample Applications


fit@hcmus

NLPA - Building NLP Applications

# NOTES FOR NLP APPLICATIONS

fit@hcmus

# Notes for NLP Applications

- In-domain vs out-domain
- Size of dataset
- Rare and OOV words
- Short vs Long sentences
- Speed vs accuracy
- ...

NLPA - Building NLP Applications

# STRUCTURE AND MANAGEMENT OF NLP PROJECTS

# Structure and management of NLP Projects

- Reasons:
  - Quick data examination (Notebooks)
  - Model storage (research repo, github)
  - New corpora
  - Quick adaptation (new code, new members)

fit@hcmus

# Structure and management of NLP Projects

- Key points:
  - Project structure
  - Data versioning
  - Tracking of ML experiments
  - Evaluation, KPI
  - Model Deployment

fit@hcmus

# Structure and management of NLP Projects

- Project Structure: including multi elements:
  - Data,
  - Models,
  - Report,
  - Training scripts,
  - Hyperparameters,
  - …

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make trai
├── README.md          <- The top-level README for developers using this projec
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for deta
│
├── models             <- Trained and serialized models, model predictions, or
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for
│                         the creator's initials, and a short `-` delimited des
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reportin
│
├── requirements.txt   <- The requirements file for reproducing the analysis en
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- Make this project pip installable with `pip install -
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models t
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented vi
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.testr
```

fit@hcmus

# Structure and management of NLP Projects

- Data versioning: ML is a iterative process:
  - Research datasets are meant to be clean
  - Data that is used in production

fit@hcmus

# Structure and management of NLP Projects

- Data used in production: more serious issues like
  - Wrong or inaccurate annotations:
    - Who annotates/annotated the data?
    - Is there a separate team or is it annotated by the users while using the product?
    - Do you need to have deep domain knowledge to successfully annotate the data? (as is the case with, for example, healthcare-related data)

fit@hcmus

# Structure and management of NLP Projects

- Data used in production: more serious issues like (cnt.)
  - Timeline of the data
    - How frequently is the data generated?
    - Are there any gaps in the data generation process (maybe the product feature that generated the data was taken down for a while)?
    - How do I know if I am not modeling on data that was an old trend (for example in fashion – apparel recommendation)

fit@hcmus

# Structure and management of NLP Projects

- Data used in production: more serious issues like (cnt.)
  - Any biases in the data
    - Sampling Bias – Data collected does not represent the population data. If the data has an ‘Age’ feature, bias may lead to overrepresentation of young people.
    - Measurement bias – One part of the data is measured using one instrument and the other part with a different instrument. This can happen in heavy industries where machineries are frequently replaced and repaired.
    - Biases in labels – Labels in a sentiment analysis task can be highly subjective. This also depends if the label is assigned by a dedicated annotation team or it is assigned by the end user.

fit@hcmus

# Structure and management of NLP Projects

- Experiment tracking:
  - Hyper-parameters
  - Model size (for memory constraints)
  - Inference time
  - Gains over baseline
  - Pros and cons (if the model supports out of vocabulary words (like fasttext) or not (like word2vec)
  - Any useful comment (for example – used a scheduler with a high initial learning rate. Worked better than using a constant learning rate.

fit@hcmus

# Structure and management of NLP Projects

- Examining model predictions (error analysis)
  - Create a baseline:
    - Where does my baseline perform better than the complex model?
  - Metrics analysis
    - What is the precision and recall for each class? Where are my misclassifications ‘leaking’ towards?
  - Low confidence predictions analysis
  - Explanation frameworks
  - Look at length vs metric score

fit@hcmus

# Structure and management of NLP Projects

- Model deployment:
  - Do I need near-real-time inference? -- Ads targeting?
  - Where would the model be hosted? – cloud, on-premise,  edge device, browser?
  - Is the model too big?
  - Do you want to deploy the model on a CPU or GPU server?

fit@hcmus

# Structure and management of NLP Projects

- Summary:
  - DVC for data versioning,
    - https://dvc.org/doc/user-guide
  - Doccano for annotations,
    - https://github.com/doccano/doccano
  - Neptune for experiment tracking,
    - https://neptune.ai/
  - fastapi for ML model deployment
    - https://fastapi.tiangolo.com/tutorial/first-steps/