NLP Applications II

NLP Pipeline

Introduction

- When we build an application, we normally walk through the requirements and **break the problem down into several sub-problems**.
- The step-by-step processing of text is known as a **pipeline**.
    - It is the series of steps involved in building any NLP model.
    - These steps are **common in every NLP project**
- We will learn about these steps and how **they play important roles** in solving NLP problems.

Generic Pipeline

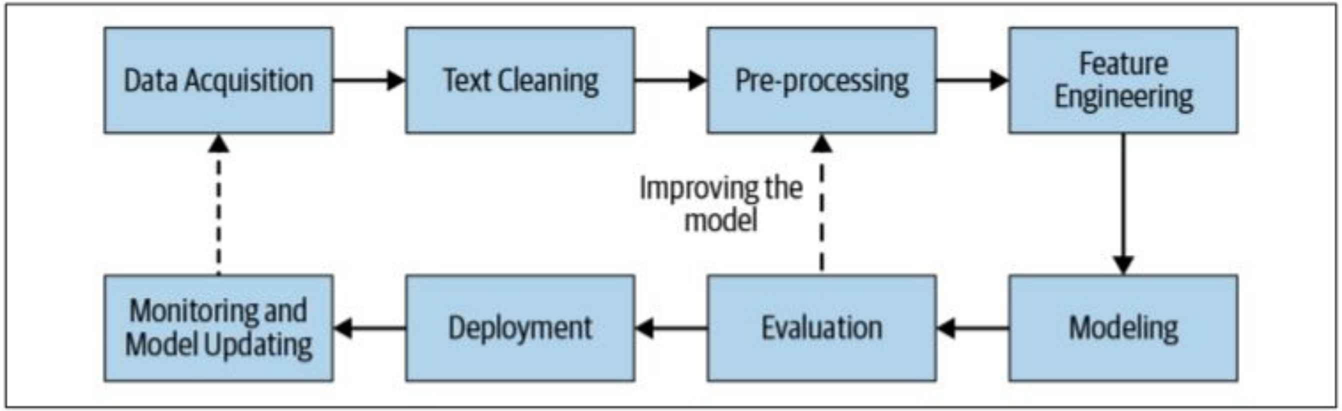


Figure 2-1. Generic NLP pipeline

Source: http://www.practicalnlp.ai/

Key steps in NLP

1. Data acquisition
2. Text cleaning
3. Pre-processing
4. Feature engineering
5. Modeling
6. Evaluation
7. Deployment
8. Monitoring and model updating

1. Data Acquisition

- In a ideal setting we would have dataset with **thousand or millions of data points**.
- **Realistic setting** (most of the industrial projects)
  - Lack of data becomes the bottleneck of many projects.
  - Important to know about different ways to gather data.
- **Common scenario**: Many companies have large amount of PDF documents, but no document is manually annotated.
- So how can we get useful annotated data?

Strategies for data acquisition

- Public datasets
  - Find a similar task and domain (see https://www.kaggle.com/, https://datasets.quantumstat.com/, https://datasetsearch.research.google.com/)
  - Build model and evaluate on your problem
- Scrape data from Internet
  - Extract data e.g discussion forum
  - Label/annotate data
- Product intervention
  - Get data from your company
  - E.g Netflix, Facebook, Microsoft, Google, Amazon...
- Data augmentation
  - Start from small dataset, create automatically more data

Data augmentation

Replacement strategies

- Synonym replacement
- TF-IDF based word replacement
- (Named) entity replacement

Adding noise

- Bigram flipping
- Character level noise
- Random insertion

Einstein was one of the outstanding figures of the 20th century"

Bohr was one of the great figures of the 20th century"

Einstein was one of the great figures of the 20th century

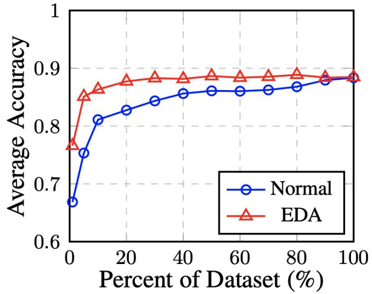
I am going to the supermarket

I am to going the supermarket

EDA: data augmentation

EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. Jason Wei, Kai Zou. EMNLP-IJCNLP. 2019

- paper:https://arxiv.org/abs/1901.11196
- code:https://github.com/jasonwei20/eda_nlp



- **Synonym Replacement (SR)**: Randomly choose n words from the sentence. Replace each of these words with one of its synonyms chosen at random.
- **Random Insertion (RI)**: Find a random synonym of a random word in the sentence. Insert that synonym into a random position in the sentence. Do this n times.
- **Random Swap (RS)**: Randomly choose two words in the sentence and swap their positions. Do this n times.
- **Random Deletion (RD)**: For each word in the sentence, randomly remove it with probability p.

Data augmentation: Backtranslation

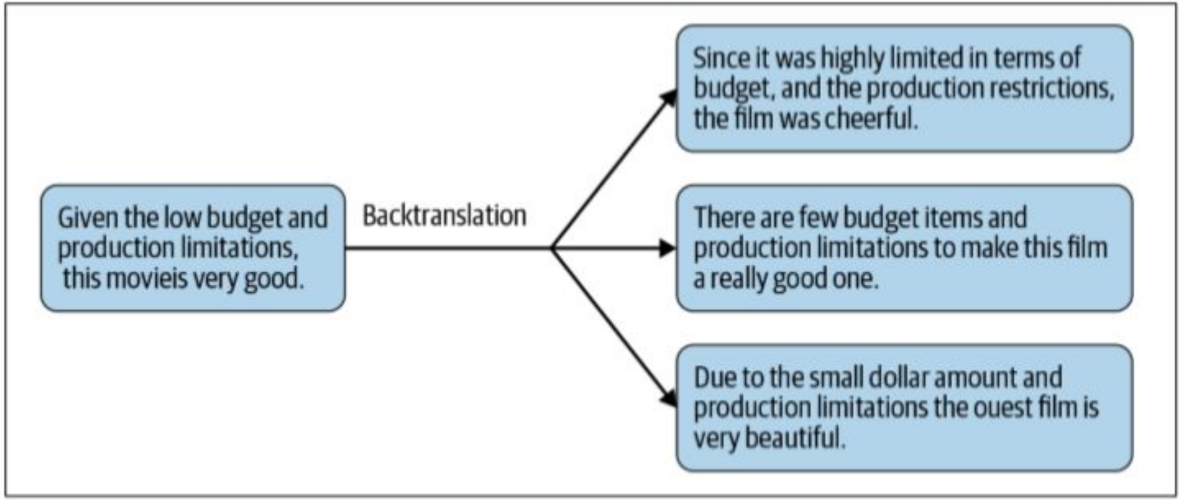


Figure 2-2. Back translation

- Use MT to generate paraphrases
  - MT: en→ es; MT: es→ en
- Easy to use MT implementations
  - https://github.com/pytorch/fairseq
- Gain more variety of augmented examples

2. Text extraction and cleaning

- Remove all non-textual information:
  - Mark-up., metadata
  - OCR
- HTML, PDF to text
- Do not require any NLP-specific technique
- **Most of the data is not just raw text!**

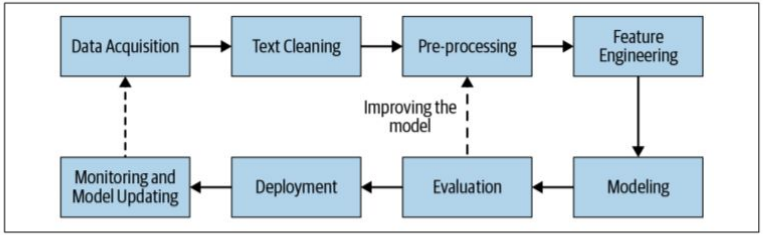


Figure 2-1. Generic NLP pipeline

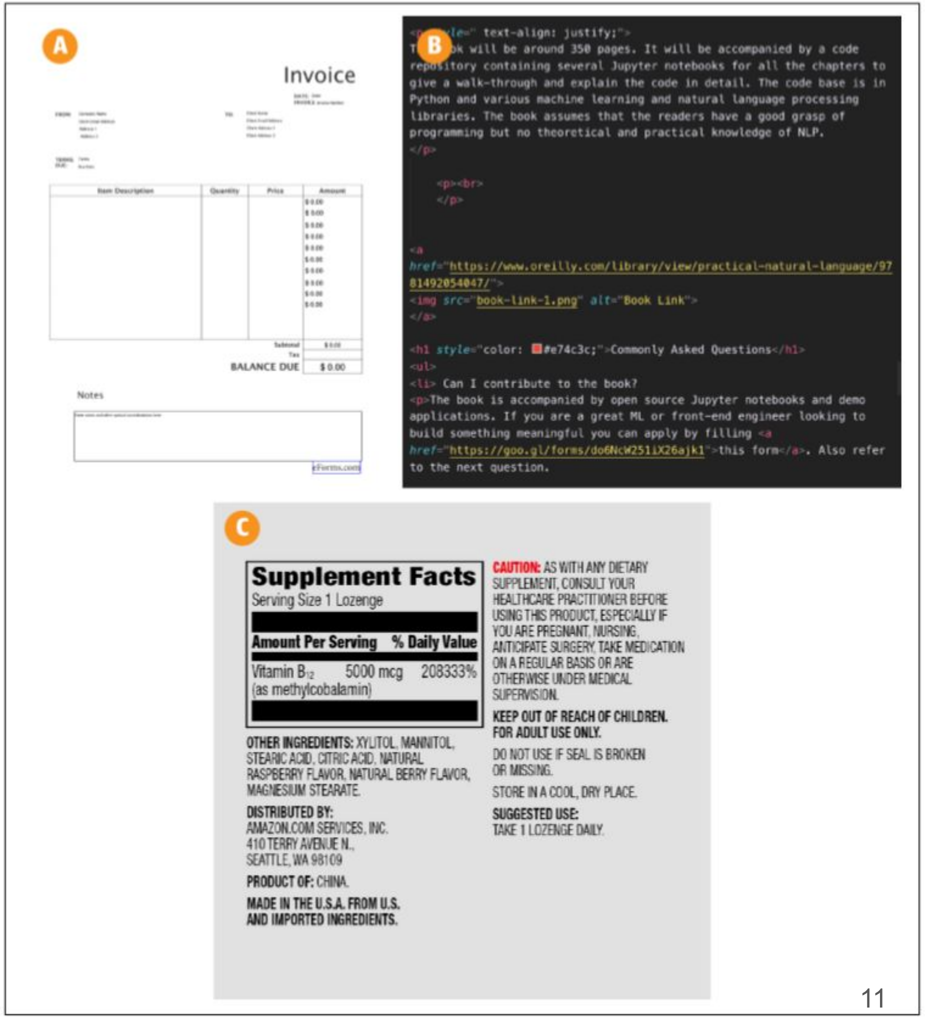


Figure 2-3. (a) PDF Invoice [13] (b) HTML texts (c) text embedded in an image [14]

3. Preprocessing

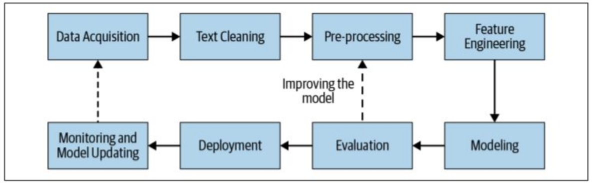
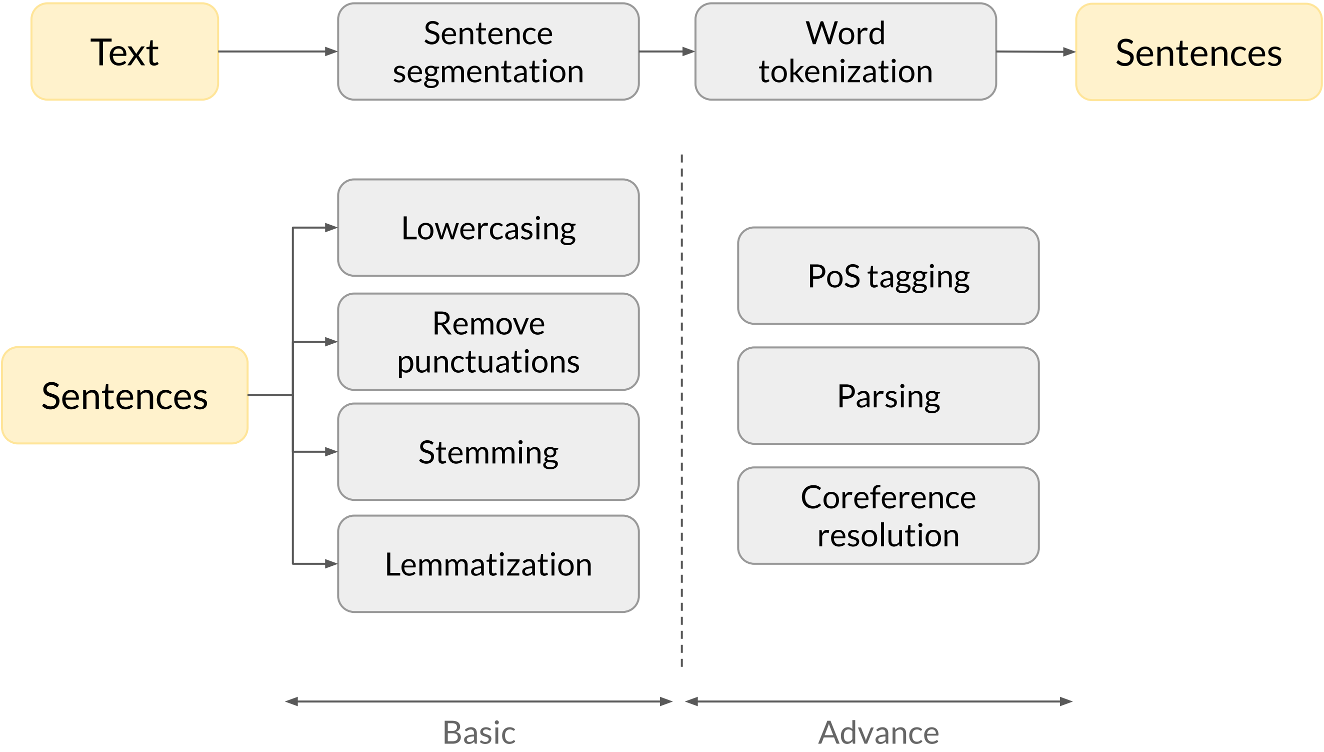


Figure 2-1. Generic NLP pipeline



Preprocessing tools

- Natural Language Toolkit (NLTK): http://www.nltk.org/
- SpaCy: https://spacy.io/
- StanfordNLP: https://stanfordnlp.github.io/stanfordnlp/
- Trankit: http://nlp.uoregon.edu/trankit

- **Similar way of use:**
  - Load text processor (tokenizer, tagger, NER)
  - Process document or text chunk
  - Extract the linguistic information you need
- **Many pretrained models and languages!**

Edit the code & try spaCy spaCy v3.0 · Python 3 · via Binder

```python
# pip install -U spacy
# python -m spacy download en_core_web_sm
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. “I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn’t "
        "worth talking to,” said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)
```

RUN

```python
from trankit import Pipeline
# initialize a pipeline on English
p = Pipeline(lang='english', gpu=True, cache_dir='./cache')

doc = '''Michael helped shoot the majority of my firm's website
and we could not have been happier.'''

# perform all tasks on the input
all = p(doc)

sents = p.ssplit(doc) # sentence segmentation
tokens = p.tokenize(doc) # tokenization
posdeps = p.posdep(doc) # upos, xpos, ufeats, dependency parsing
ners = p.ner(doc) # ner tagging
lemmas = p.lemmatize(doc) # lemmatization
```

```python
>>> import stanfordnlp
>>> stanfordnlp.download('en')   # This downloads the English models for the neural pipeline
>>> nlp = stanfordnlp.Pipeline() # This sets up a default neural pipeline in English
>>> doc = nlp("Barack Obama was born in Hawaii.  He was elected president in 2008.")
>>> doc.sentences[0].print_dependencies()
```

4. Feature engineering

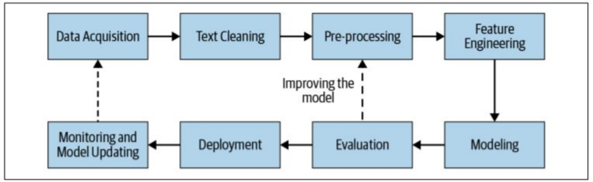


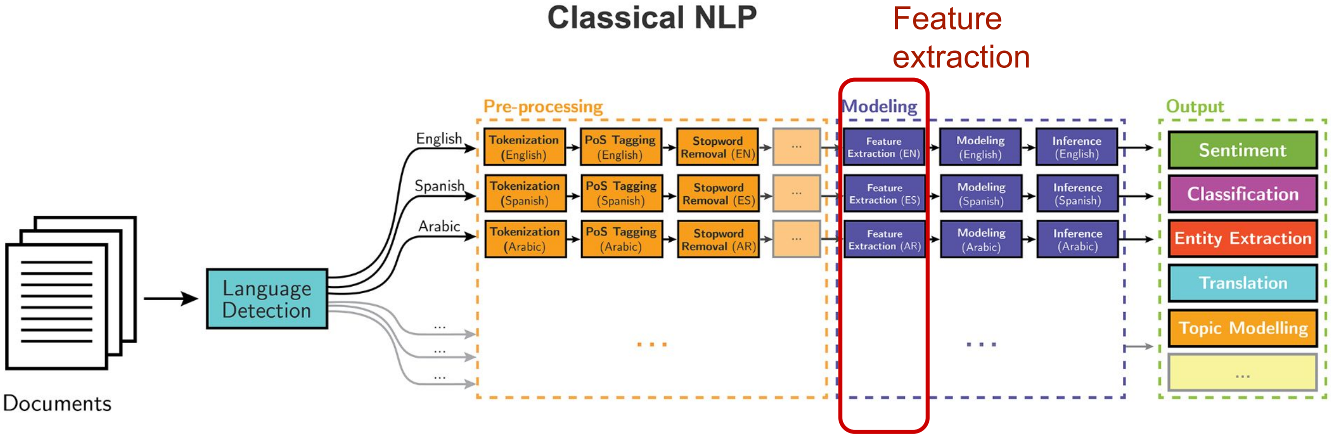
Figure 2-1. Generic NLP pipeline

- Set of methods that feed pre-processed text into ML algorithms.
- Capture the characteristics of the text into a numeric vector that can be understood by the ML algorithms.
- Also known as *feature extraction* or *text representation.*

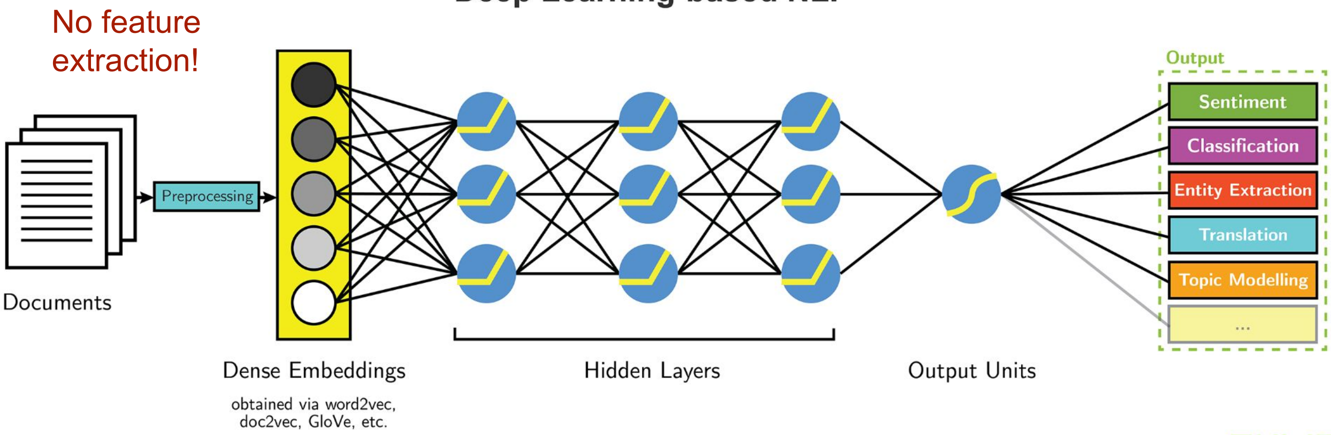
Two different approaches:

1. Classical NLP with traditional ML pipeline
   - Hand crafted features: e.g. `number_positive_words(sentence)`
   - Domain knowledge
   - Statistical measure to usefulness of features (interpretable)
   - Noisy/unrelated information
2. DL pipeline
   - Capable of learning *features* (hidden) inline with task requirements
   - Difficult to interpret

Classical NLP



Deep Learning-based NLP



https://aylien.com/blog/leveraging-deep-learning-for-multilingual

AYLIEN

5. Modeling

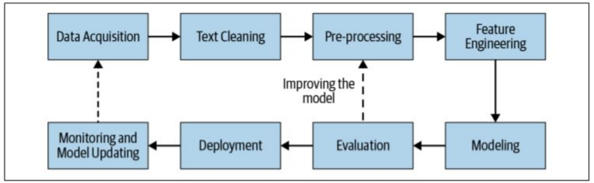


Figure 2-1. Generic NLP pipeline

Simple Heuristics

- Start building systems by encoding this knowledge in the form of rules/heuristics
- Spam detection. Use blacklist to learn words associated with spam emails
- Regular expression to extract phone numbers, names, etc.
- https://spacy.io/usage/rule-based-matching (rule-based matching)
  https://explosion.ai/demos/matcher

```
{(([{ ner:PERSON }]) /was/ /born/ /on/ ([ { ner:DATE } ]))
=>"DATE_OF_BIRTH" }
```

Use heuristics as features

- Combination of many heuristic can be fuzzy
- Use heuristics as features (e.g. number of words from the blacklist)

Modeling

AutoML

- provides automatic methods and processes to apply Machine Learning
- data pre-processing, feature engineering, algorithm selection, hyperparameter optimization
- Make ML available for non-Machine Learning experts

Some examples:

- AutoWEKA: simultaneous selection of a machine learning algorithm and its hyperparameters.
- Auto-sklearn: similar framework for scikit-learn
- H2O AutoML: provides automated model selection and ensembling for the H2O machine learning and data analytics platform.
- MLBoX: AutoML library with three components: preprocessing, optimisation and prediction.

Modeling

Ensemble and stacking

- Combining more than one prediction usually improve results
- **Model stacking**: Feed one model’s output as input for another.
- **Model ensembling**: Pool predictions from multiple models and make final prediction.

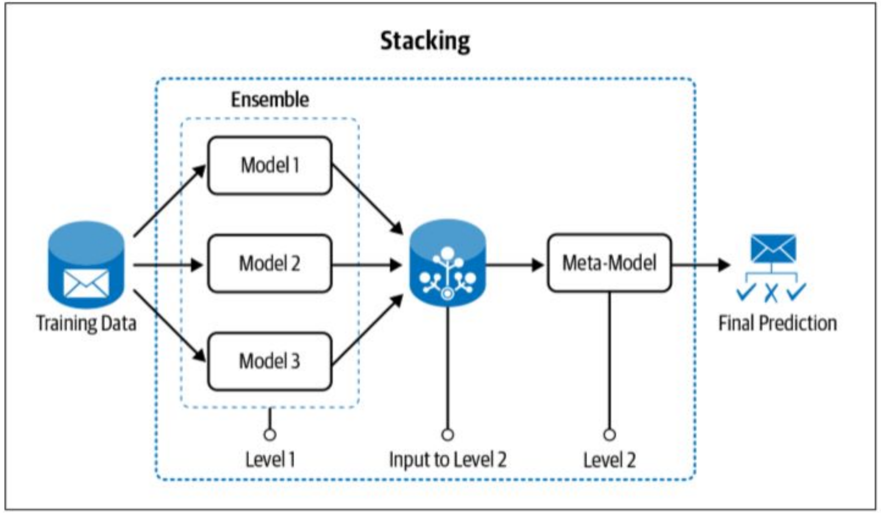


Figure 2-14. Model ensemble and stacking

Modeling

Better feature engineering

- Using many feature can be inconvenient
- Use f**eature selection** to find a better model

Transfer learning

- Transfer preexisting knowledge from a big, well-trained model to a newer model at its initial phase
- Provides a better initialization, which helps in the downstream tasks, especially when the dataset for the downstream task is smaller.

Modeling

Large data volume

- Use DL or a richer set of features

Small data volume

- Start with rule-based or traditional ML.
- Use transfer learning if possible (need pre-trained LM)
- Use prompt learning if possible (need pre-trained model)

6. Evaluation

- Goodness of model can have multiple meanings.
  - The most common interpretation is the measure of the model’s performance on unseen data.
- Success of evaluation depends on **evaluation metric and process**.
- **Evaluation metric** depends on task and phase
  - Model building, deployment, and production phases.
- Two types of evaluations:
  - **Intrinsic**: Focus on intermediary objective
  - **Extrinsic**: Focus on final objective
- E.g. spam-classification system.
  - The intrinsic metric will be precision and recall.
  - The business metric (extrinsic) will be “the amount of time users spent on a spam email.

Intrinsic evaluation

- Assume a test set with a ground-truth/labels (human annotated examples)
- Labels can be binary (text classification), one-to-two words (e.g. NER), or large texts (e.g MT)
- The output of the NLP model is compared against the corresponding label for that data point.
- Metrics are calculated based on the match (or mismatch) between the output and label.
    - **Text classification, NER, Relation Extraction**: Precision, Recall, Fscore, ROC,
    - **Text generation**: BLEU, METEOR, ROUGE
    - **Similarity**: Pearson correlation, MSE
    - **Ranking**: MAP, Recall@K

Extrinsic evaluation

- Evaluation of model performance on **final practical objective**
- Model with great intrinsic metric can fail achieving business objective
  - QA system makes great on SQUAD, but might fail answering large number of question in the production environment.
- More **expensive than intrinsic evaluation** (that’s why we need intrinsic evaluation).
- Bad results in intrinsic imply bad results in extrinsic.
- Good results in intrinsic do not imply good performance in extrinsic.

Wrap up

- We saw different steps involved in developing an NLP pipeline
- Specific details for each step will depend on the task at hand and the purpose of our implementation
- Differences between traditional NLP pipeline and a DL-based NLP pipeline
- Other languages might treated differently.