AASD 4004
Machine Learning - II

Applied AI Solutions Developer Program

Module 1
NLP Revisited

Vejey Gandyer

Agenda

- What is NLP
- NLP Tasks
- Applications
- NLP Building Blocks
- Challenges of NLP
- NLP Approaches
- NLP Pipeline

NLP

What is it?

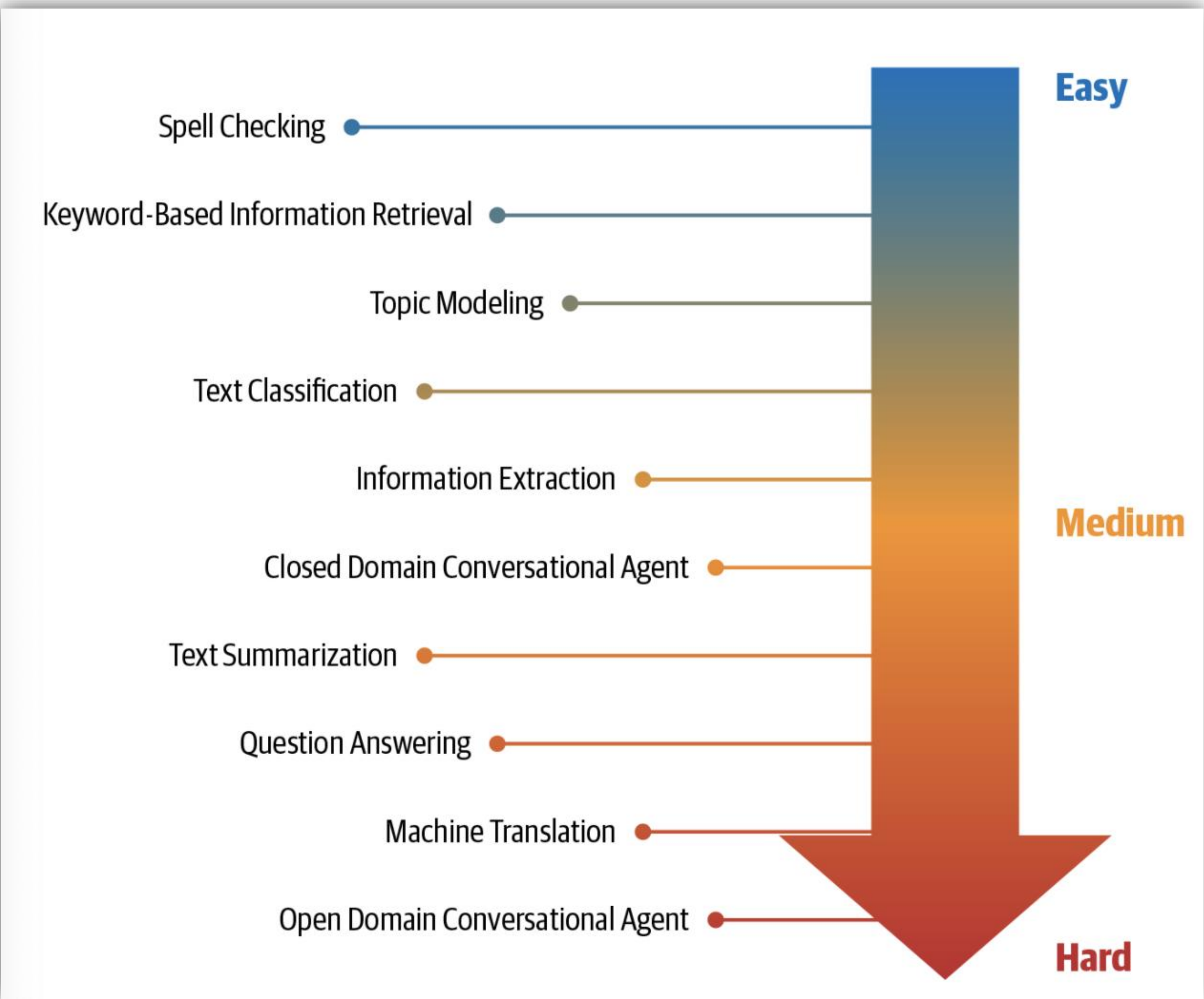
Natural Language Processing (NLP)

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, how to program computers to process and analyze large amounts of natural language data.

NLP Tasks

NLP Tasks

Language Modeling
Text Classification
Information Extraction
Topic Modeling
Information Retrieval
Text Summarization
Question Answering
Machine Translation
Conversational Agent



NLP Applications

Text Classification

- Predict Tags or Categories
- Predict Sentiment
- Filter Spam mails

Sequence Applications

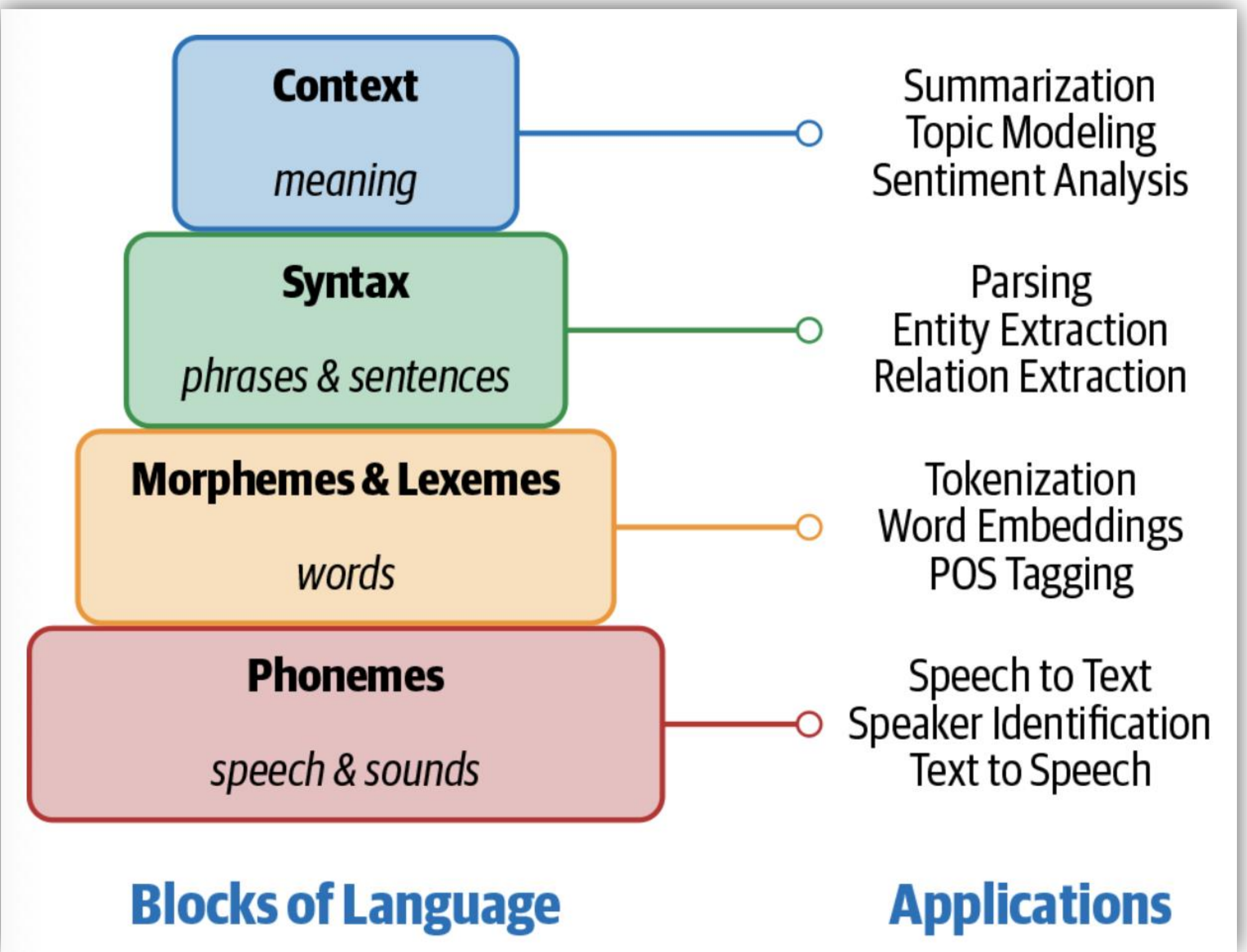
- Part Of Speech Tags
- Named Entity Recognition
- Semantic Slots

Sequence to Sequence

- Machine Translation
- Summarization
- Speech Recognition
- Question Answering

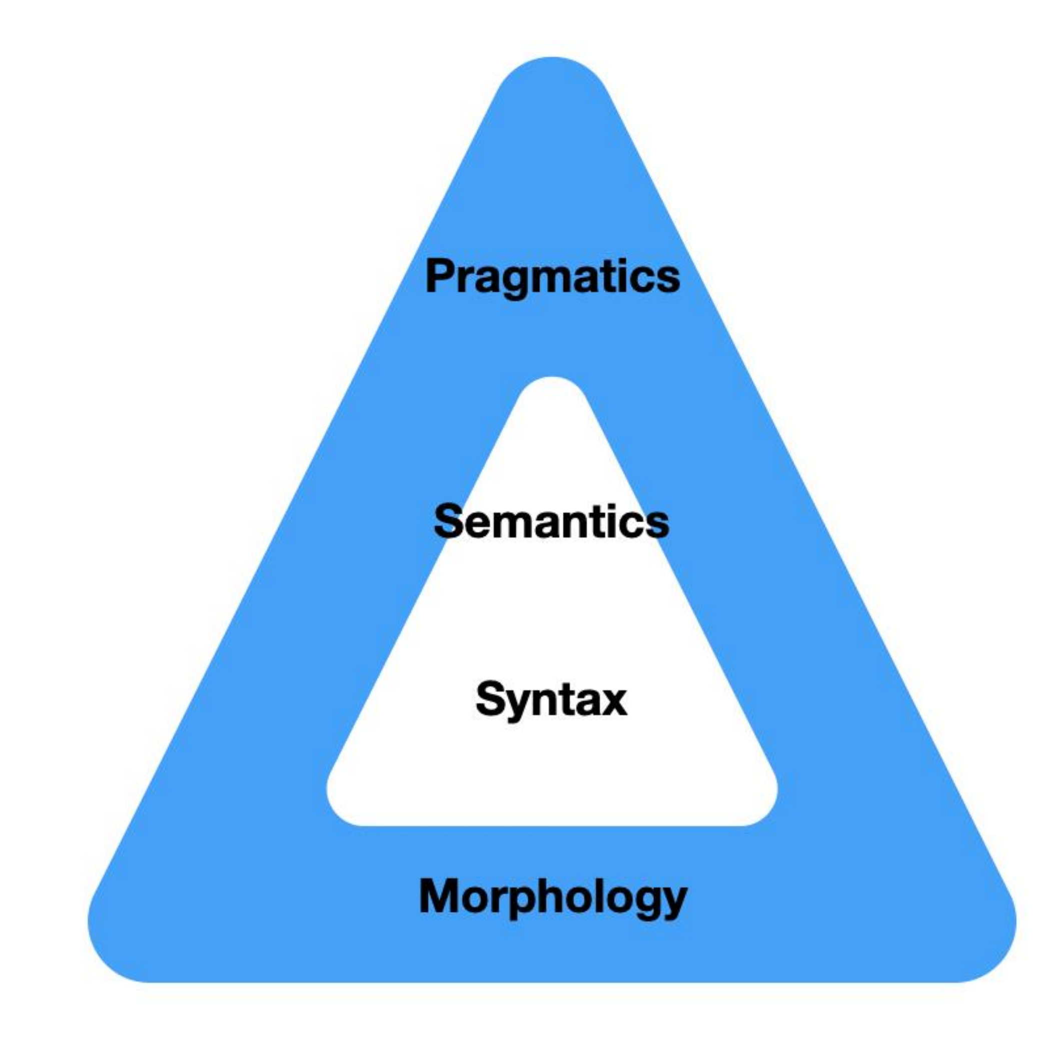
NLP Building Blocks

Building Blocks of Language



Linguistic Pyramid

- Morphology - Pre-processing
- Syntax
- Semantics
- Pragmatics



Representations

- Word Embeddings
- Sentence Embeddings
- Topic Models (Documents)
- Vector Space Models
- Similarity Graphs

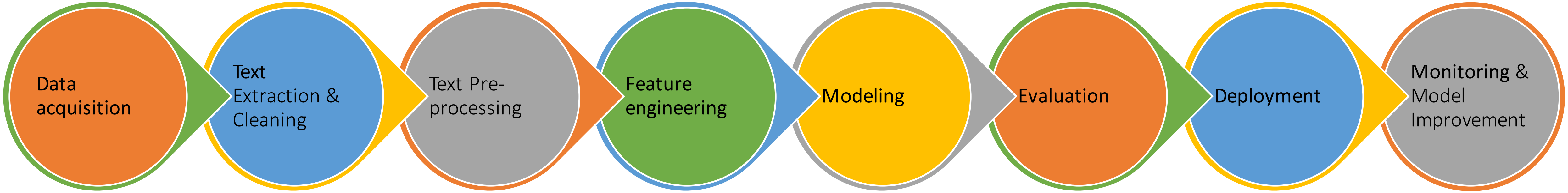
NLP Approaches

NLP Approaches

- Rule Based
  - Regular Expressions
  - Context-free Grammars
- Machine Learning
  - Probabilistic Modeling
  - Linear Classifiers
- Deep Learning
  - Recurrent Neural Networks
  - Convolutional Neural Networks

NLP Pipeline

NLP Pipeline



NLP Pipeline

Data Acquisition

Data Acquisition - Motivation

Data Acquisition – heart of any ML System

Hypothetical NLP project: Creating a NLP system to identify whether the user query is a sales query or a customer care query.

Ideal scenario: Millions of data points available already. No need for data acquisition

Real scenario: No data or less data with unlabelled classes. Need to acquire data

But how???

Data Acquisition

- Use a Public Dataset
- Scrape data from ungated websites
- Product Intervention
- Data Augmentation
  - Synonym Replacement
  - Back Translation
  - TF-IDF Word Replacement
  - Bigram Flipping
  - Replacing entities
  - Adding noise

NLP Pipeline

Text Extraction & Cleaning

Text Extraction and Cleaning - Motivation

Text Extraction - Process of extracting raw text from input data by removing all the other unwanted non-textual information (markup, metadata, …)

Text Cleaning – a must as real-world data is always messy, noisy and incomplete 99% of the time

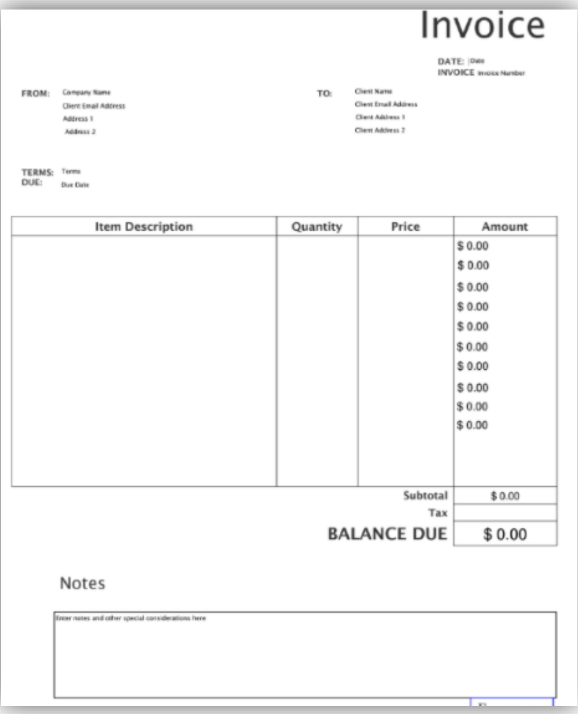
Ideal scenario: Cleaned data points already. No need for data cleaning

Real scenario: Noisy, incomplete, messy data. Need to clean data

But how???

Text Extraction and Cleaning

Various formats of input data (PDF, HTML, continuous stream, …)



Invoice

DATE: Date
INVOICE: Invoice Number

FROM: Company Name
Client Email Address
Address 1
Address 2

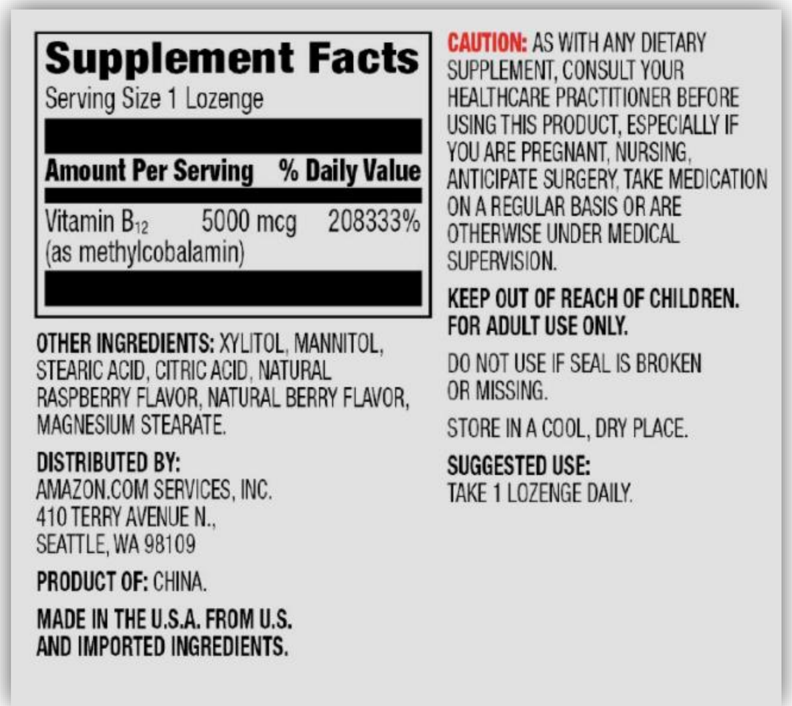
TO: Client Name
Client Email Address
Client Address 1
Client Address 2

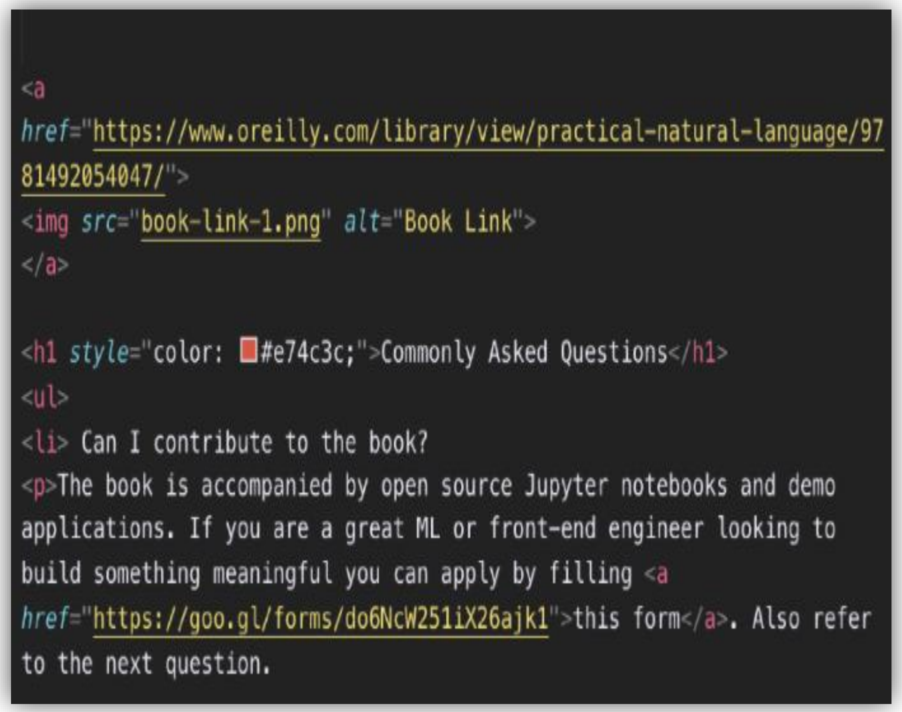
TERMS: Terms
DUE: Due Date

Item Description	Quantity	Price	Amount
			$ 0.00
			$ 0.00
			$ 0.00
			$ 0.00
			$ 0.00
			$ 0.00
			$ 0.00
			$ 0.00
			$ 0.00
			$ 0.00
		Subtotal	$ 0.00
		Tax	
		BALANCE DUE	$ 0.00

Notes

Enter notes and other special considerations here

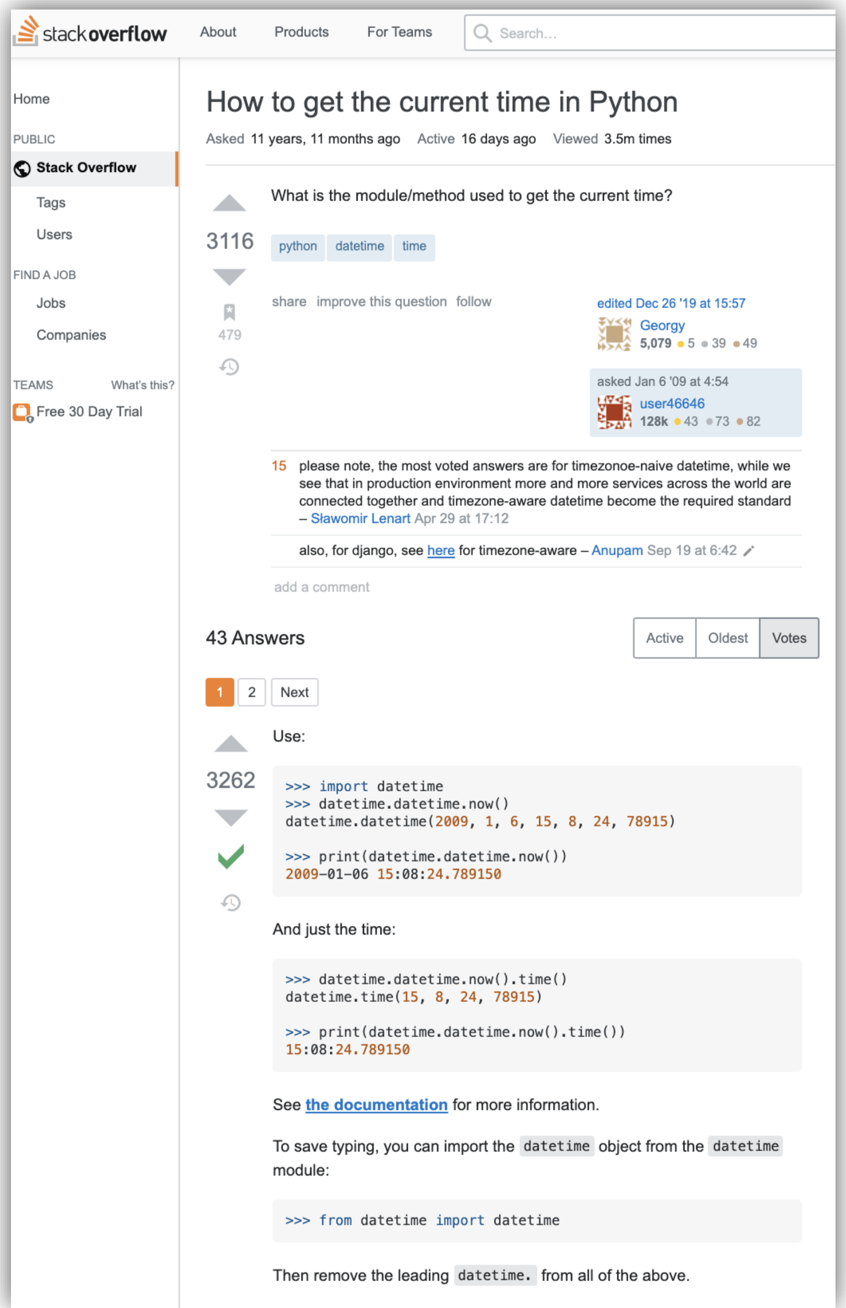




```html
<a
href="https://www.oreilly.com/library/view/practical-natural-language/97
81492054047/">
<img src="book-link-1.png" alt="Book Link">
</a>

<h1 style="color: #e74c3c;">Commonly Asked Questions</h1>
<ul>
<li> Can I contribute to the book?
<p>The book is accompanied by open source Jupyter notebooks and demo
applications. If you are a great ML or front-end engineer looking to
build something meaningful you can apply by filling <a
href="https://goo.gl/forms/do6NcW251iX26ajk1">this form</a>. Also refer
to the next question.
```

Web scraping




```
Text =
 What is the module/method used to get the current time?



Answer =
 Use:
>>> import datetime
>>> datetime.datetime.now()
datetime.datetime(2009, 1, 6, 15, 8, 24, 78915)

>>> print(datetime.datetime.now())
2009-01-06 15:08:24.789150

And just the time:
>>> datetime.datetime.now().time()
datetime.time(15, 8, 24, 78915)

>>> print(datetime.datetime.now().time())
15:08:24.789150

See the documentation for more information.
To save typing, you can import the datetime object from the datetime module:
>>> from datetime import datetime

Then remove the leading datetime. from all of the above.
```

Text extraction from Scanned images

In the nineteenth century the only kind of linguistics considered seriously was this comparative and historical study of words in languages known or believed to be *cognate*—say the Semitic languages, or the Indo-European languages. It is significant that the Germans who really made the subject what it was, used the term *Indo-germanisch*. Those who know the popular works of Otto Jespersen will remember how firmly he declares that linguistic science is historical. And those who have noticed

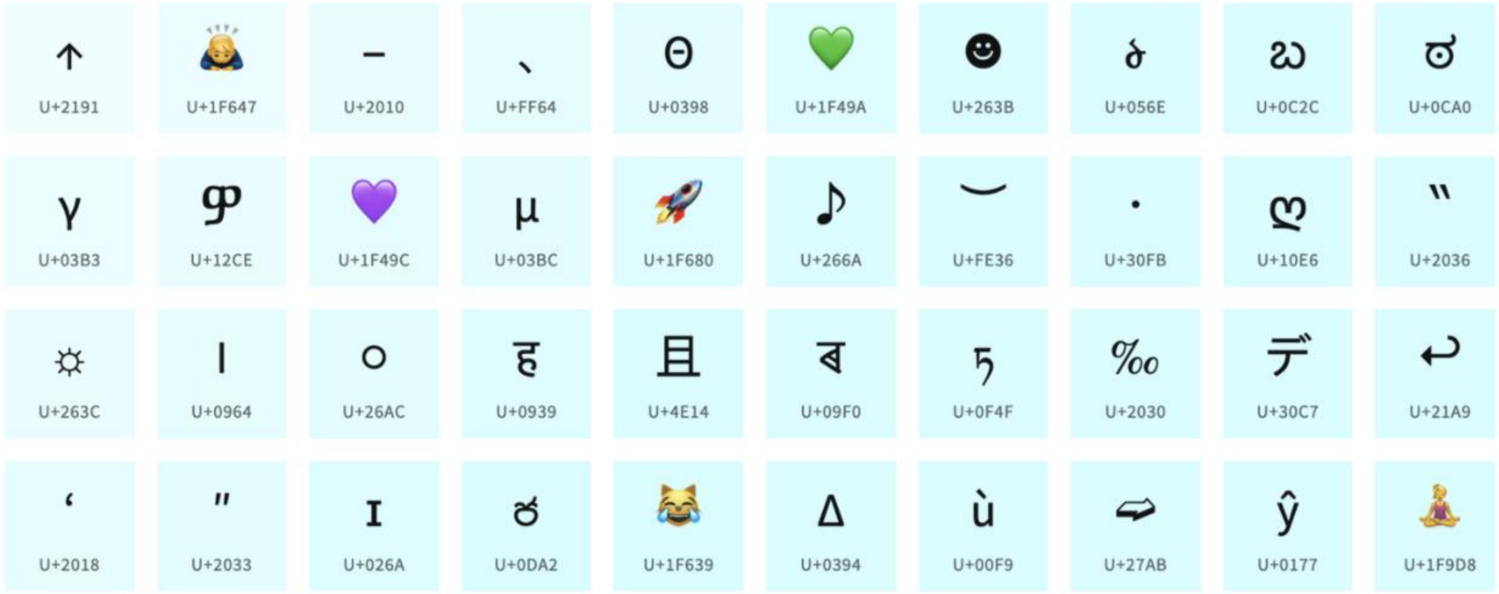
```python
from PIL import Image
from pytesseract import image_to_string
filename = "somefile.png"
text = image_to_string(Image.open(filename))
print(text)
```

```
'in the nineteenth century the only Kind of linguistics considered\nseriously
was this comparative and historical study of words in languages\nknown or
believed to Fe cognate—say the Semitic languages, or the Indo-\nEuropean
languages. It is significant that the Germans who really made\nthe subject what
it was, used the term Indo-germanisch. Those who know\nthe popular works of
Otto Jespersen will remember how fitmly he\ndeclares that linguistic
science is historical. And those who have noticed'
```

PyPDF, PDFMiner, … can be used to extract PDF text

If PDF text is a scanned image, use **Tesseract** library

Unicode removal




```
text = ’I love 🍕! Shall we book a 🚕 to gizza?’
Text = text.encode("utf-8")
print(Text)
```

```
b'I love Pizza \xf0\x9f\x8d\x95!  Shall we book a cab \xf0\x9f\x9a\x95 to get pizza?'
```

To remove non-textual symbols and special characters, use Unicode Normalization

Use string.encode("utf-8")

Spelling correction

```python
import requests
import json

api_key = "<ENTER-KEY-HERE>"
example_text = "Hollo, wrld" # the text to be spell-checked

data = {'text': example_text}
params = {
    'mkt':'en-us',
    'mode':'proof'
    }
headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
    'Ocp-Apim-Subscription-Key': api_key,
    }
response = requests.post(endpoint, headers=headers, params=params, data=data)
json_response = response.json()
print(json.dumps(json_response, indent=4))
```

Microsoft has APIs for spell checking

Use **Levenshtein distance** for detecting and correcting spelling mistakes


```
"suggestions": [
            {
                "suggestion": "Hello",
                "score": 0.9115257530801
            },
            {
                "suggestion": "Hollow",
                "score": 0.858039839213461
            },
            {
                "suggestion": "Hallo",
                "score": 0.597385084464481
            }
]
```

NLP Pipeline

Text Pre-processing

Text Pre-processing - Motivation

Text Pre-processing - Process of preparing raw text extracted from data sources by some processes like Sentence Segmentation, Word Tokenization, Stop words, Stemming & Lemmatization, Special characters removal, POS tagging, Coreference resolution, etc.

Text Processing – a must as real-world needs to be in a certain format for a machine learning algorithm to be accepted as input

But how???

Text Pre-processing

Sentence Segmentation

Word Tokenization

Stop words removal

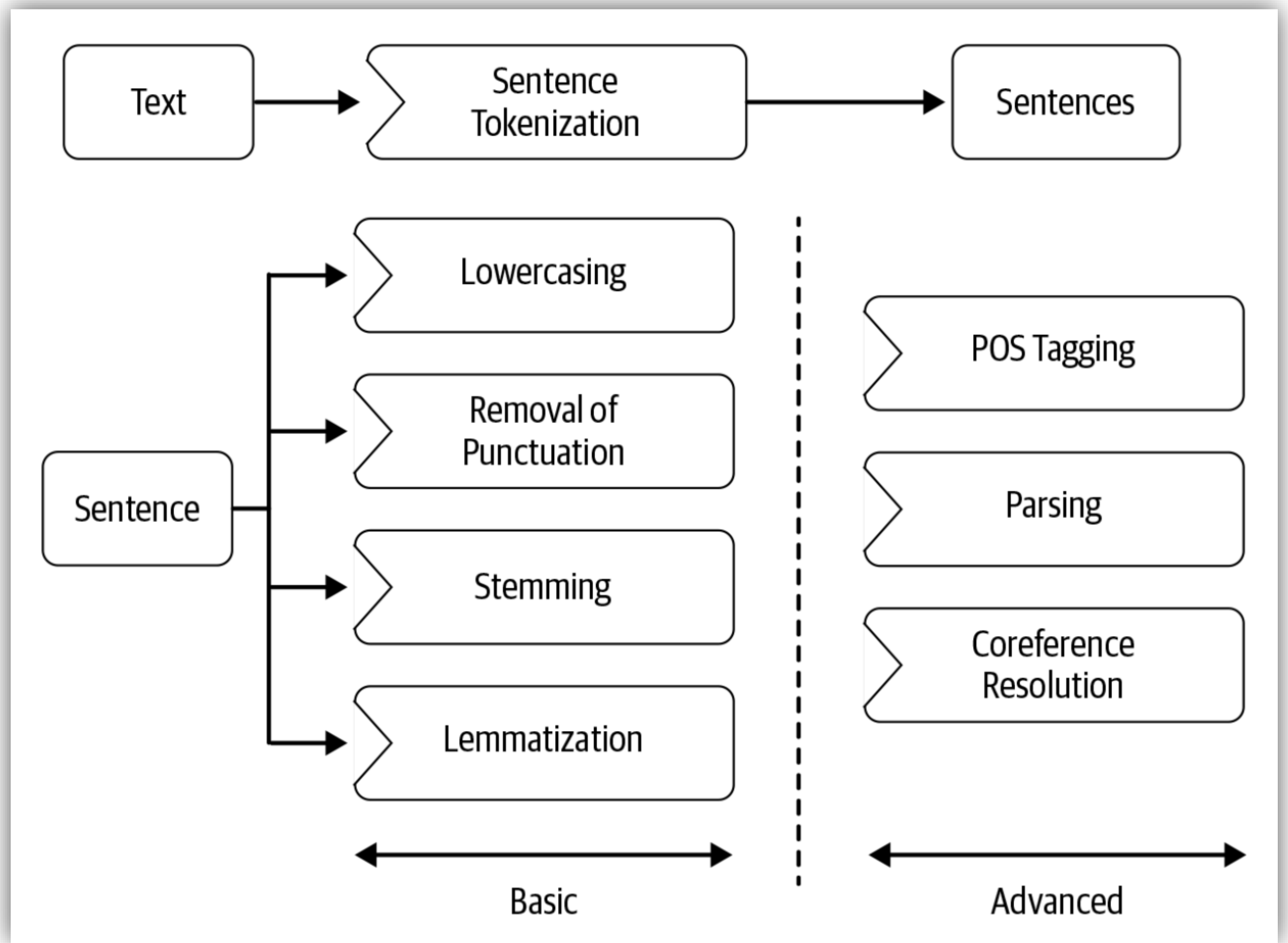
Stemming & Lemmatization

Normalization

POS tagging

Parse Tree

Coreference resolution



Sentence Segmentation

Breaking a big document text into sentences

```
mytext = """In the previous chapter, we saw examples of some common NLP
applications that we might encounter in everyday life. If we were asked to
build such an application, think about how we would approach doing so at our
organization. We would normally walk through the requirements and break the
problem down into several sub-problems, then try to develop a step-by-step
procedure to solve them. Since language processing is involved, we would also
list all the forms of text processing needed at each step. This step-by-step
processing of text is known as pipeline. """
```

```
from nltk.tokenize import sent_tokenize
my_sentences = sent_tokenize(mytext)
print(my_sentences)
```

```
['In the previous chapter, we saw examples of some common NLP applications that we might encounter in everyday life.',
'If we were asked to build such an application, think about how we would approach doing so at our organization.',
'We would normally walk through the requirements and break the problem down into several sub-problems, then try to develop
'Since language processing is involved, we would also list all the forms of text processing needed at each step.',
'This step-by-step processing of text is known as pipeline.'
]
```

Word Tokenization

```
['In the previous chapter, we saw examples of some common NLP applications that we might encounter in everyday life.',
'If we were asked to build such an application, think about how we would approach doing so at our organization.',
'We would normally walk through the requirements and break the problem down into several sub-problems, then try to develop
'Since language processing is involved, we would also list all the forms of text processing needed at each step.',
'This step-by-step processing of text is known as pipeline.'
]
```

```python
from nltk.tokenize import word_tokenize
for sentence in my_sentences:
    print(sentence)
    print(word_tokenize(sentence))
```

```
This step-by-step processing of text is known as pipeline.
['This', 'step-by-step', 'processing', 'of', 'text', 'is', 'known', 'as', 'pipeline', '.']
```

Breaking a big sentence into words or tokens

Stop words removal

```
"In the previous chapter, we saw examples of some common NLP applications that we might encounter in everyday life."
```

```
from nltk.corpus import stopwords
from string import punctuation
def preprocess_corpus(texts):
    mystopwords = set(stopwords.words("english"))
    def remove_stops_digits(tokens):
        return [token.lower() for token in tokens if token not in mystopwords and
                not token.isdigit() and token not in punctuation]
    return [remove_stops_digits(word_tokenize(text)) for text in texts]
```

```
['in', 'previous', 'chapter', 'saw', 'examples', 'common', 'nlp', 'applications', 'might', 'encounter', 'everyday', 'life']
```

Stemming

```
cars revolution
```

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
word1, word2 = “cars”, “revolution”
print(stemmer.stem(word1), stemmer.stem(word2))
```

```
car revolut
```

Removes suffixes and reduces a word to some **base form**

Lemmatization

Also removes suffixes and reduces a word to some **base form** or **lemma**

Its lemma will be there in the dictionary (meaningful)

better

```python
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("better", pos="a"))
```

good

Stemming	Lemmatization
adjustable -> adjust	was -> (to) be
formality -> formaliti	better -> good
formaliti -> formal	meeting -> meeting
airliner -> airlin	

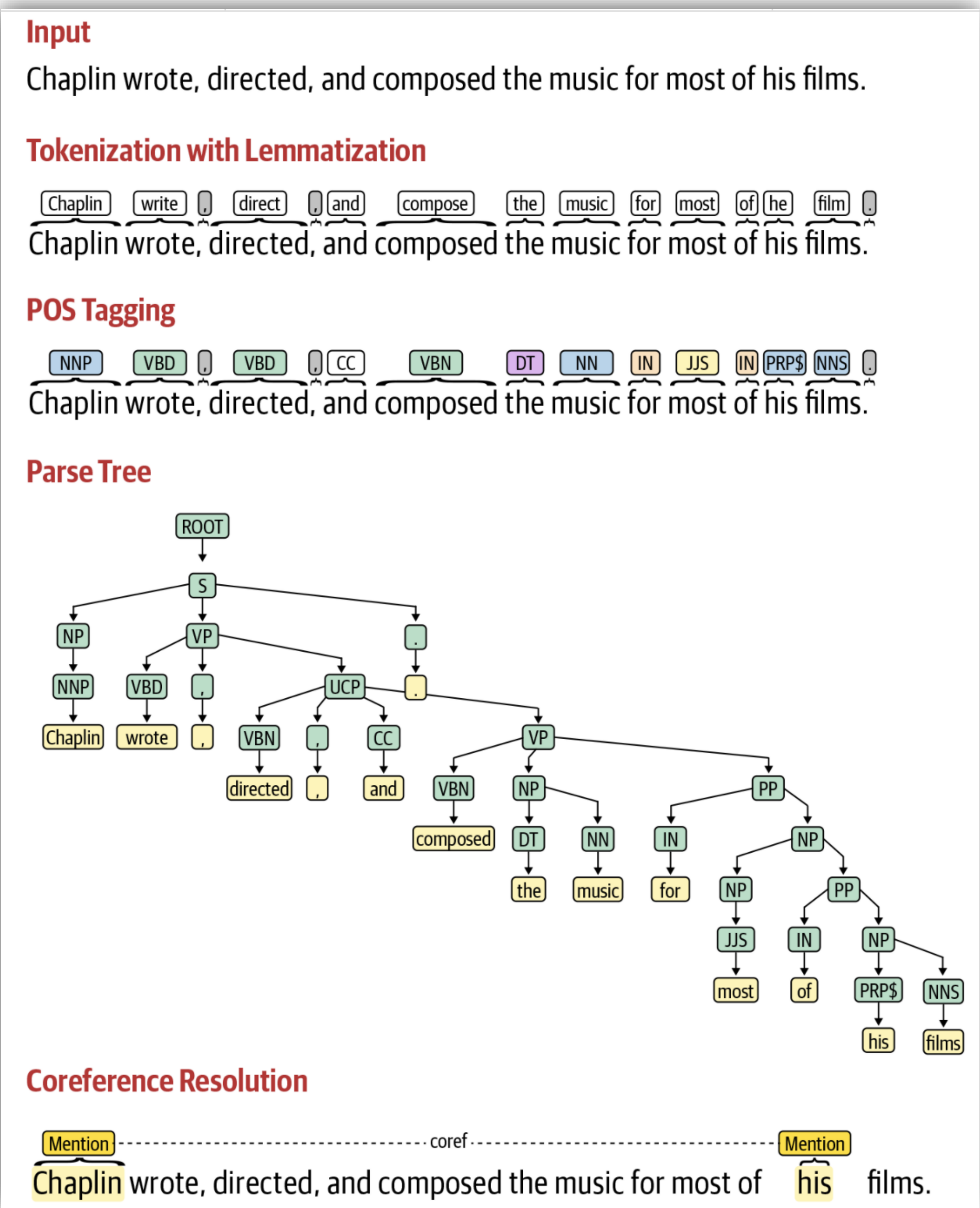
Advanced pre-processing

To use spacy, first install them in your conda / miniconda / virtual environment

pip3 install spacy

python3 -m spacy download en_core_web_sm

```
import spacy
nlp = spacy.load('en_core_web_sm')
doc = nlp(u"""Charles Spencer Chaplin was born on 16 April 1889
    to Hannah Chaplin (born Hannah Harriet Pedlingham Hill)
    and Charles Chaplin Sr""")
for token in doc:
    print(token.text, token.lemma_, token.pos_,
          token.shape_, token.is_alpha, token.is_stop)
```



NLP Pipeline

Feature Engineering

Feature Engineering - Motivation

Feature Engineering - Process of set of methods that will accomplish the task of feature extraction (converting text into numeric vectors)

Feature Engineering is dealt in the future lecture in detail.

Two major categories are defined here.

1. Classical NLP / ML Pipeline
2. DL Pipeline

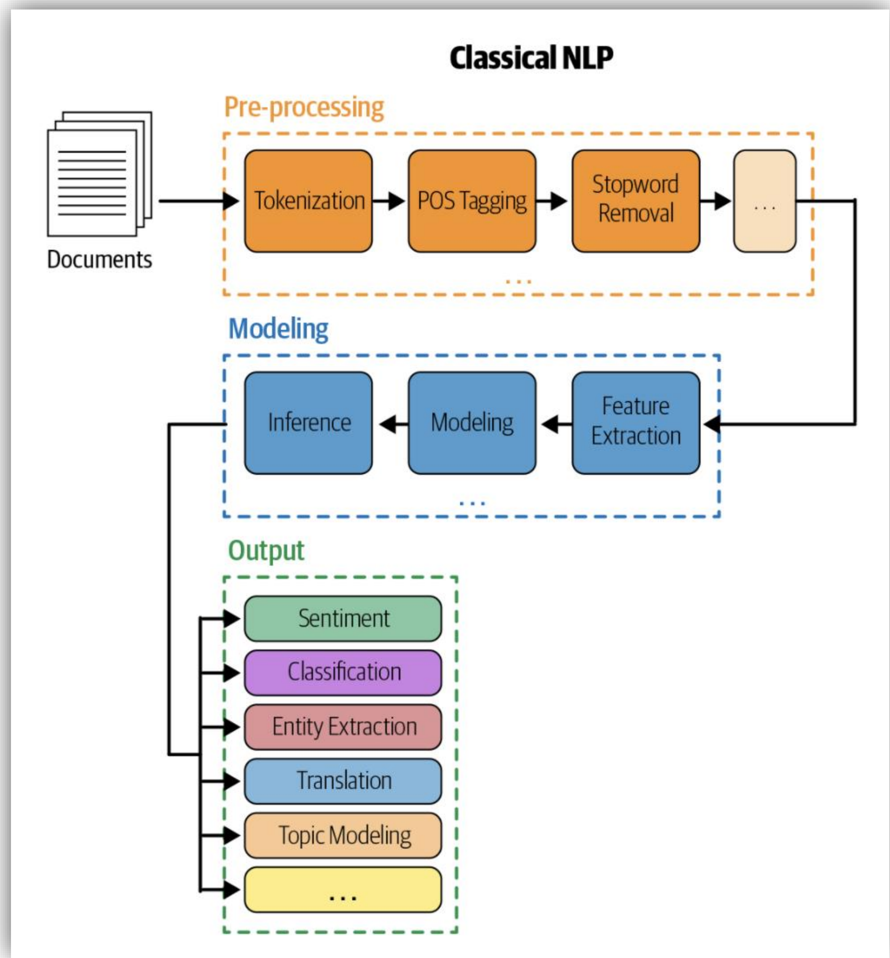
Feature Engineering - Classical NLP

Converts the raw data into a format that can be consumed by a machine.

In Classical ML, we have turned categorical variables into numbers and fed into the model.

In NLP, we need to convert the text into some form of numeric vectors and will feed it into the model

In Classical NLP, the feature extraction process is **handcrafted** and done by engineers who have **domain expertise** in the area of the problem in hand

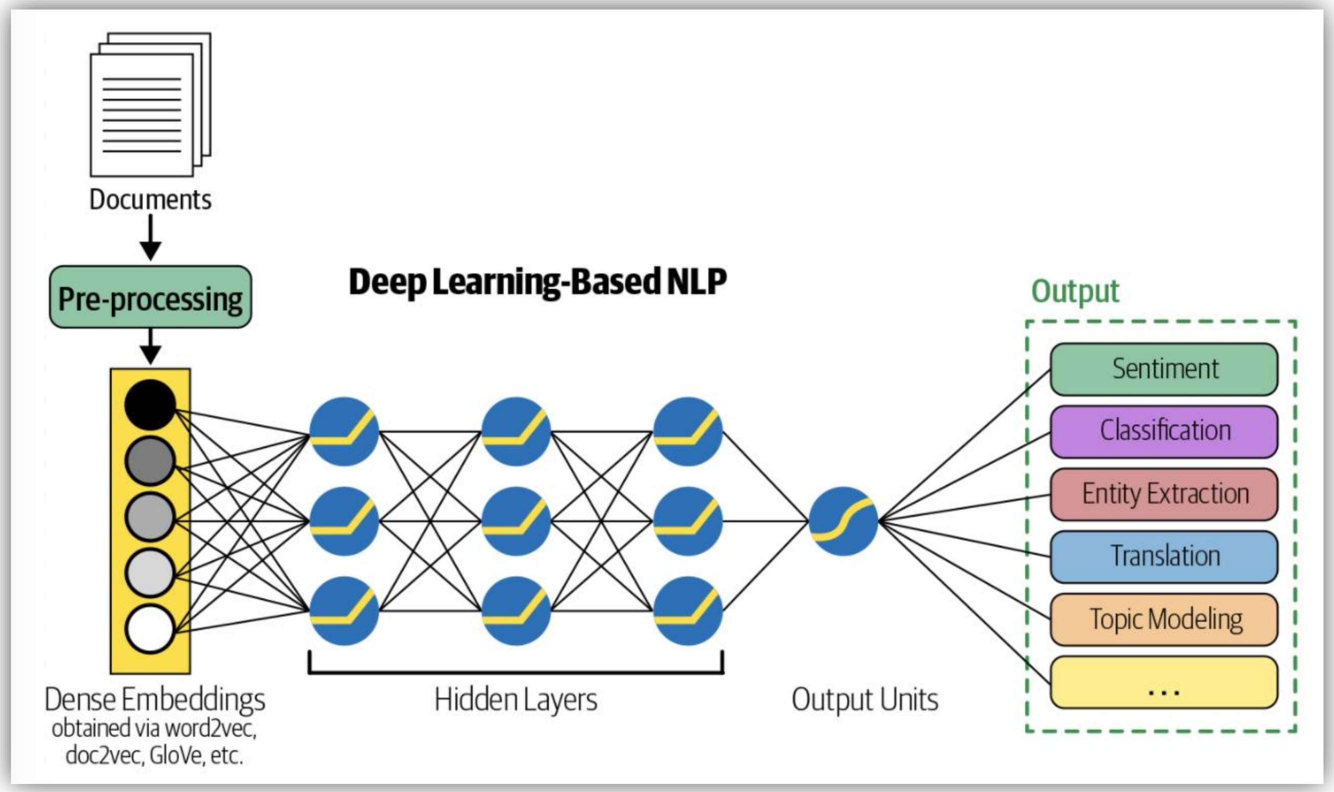


Feature Engineering - Deep NLP

Converts the raw data into a format that can be consumed by a machine.

In Deep Learning NLP, the model takes care of the feature extraction process

Raw data after pre-processing is sent to the model directly



NLP Pipeline

Modeling

Modeling - Motivation

Modeling - Process of building a model depending on the amount of data we have in hand

Simple Heuristics - Regular Expressions, Rule-based approaches

Create a feature from the heuristic for the ML model

Pre-process input to the ML model

Modeling

Ensembles of Models

Feature Engineering

Transfer Learning

Data attribute	Decision path	Examples
Large data volume	Can use techniques that require more data, like DL. Can use a richer set of features as well. If the data is sufficiently large but unlabeled, we can also apply unsupervised techniques.	If we have a lot of reviews and metadata associated with them, we can build a sentiment-analysis tool from scratch.
Small data volume	Need to start with rule-based or traditional ML solutions that are less data hungry. Can also adapt cloud APIs and generate more data with weak supervision. We can also use transfer learning if there's a similar task that has large data.	This often happens at the start of a completely new project.
Data quality is poor and the data is heterogeneous in nature	More data cleaning and pre-processing might be required.	This entails issues like code mixing (different languages being mixed in the same sentence), unconventional language, transliteration, or noise (like social media text).
Data quality is good	Can directly apply off-the-shelf algorithms or cloud APIs more easily.	Legal text or newspapers.
Data consists of full-length documents	Choose the right strategy for breaking the document into lower levels, like paragraphs, sentences, or phrases, depending on the problem.	Document classification, review analysis, etc.

NLP Pipeline

Evaluation

Evaluation - Motivation

Evaluation – Measuring how good the model is

1) Use the right metric
2) Follow the right evaluation process

Intrinsic Evaluation

Extrinsic Evaluation

Intrinsic Evaluation

Metric	Description	Applications
Accuracy [48]	Used when the output variable is categorical or discrete. It denotes the fraction of times the model makes correct predictions as compared to the total predictions it makes.	Mainly used in classification tasks, such as sentiment classification (multiclass), natural language inference (binary), paraphrase detection (binary), etc.
Precision [48]	Shows how precise or exact the model's predictions are, i.e., given all the positive (the class we care about) cases, how many can the model classify correctly?	Used in various classification tasks, especially in cases where mistakes in a positive class are more costly than mistakes in a negative class, e.g., disease predictions in healthcare.
Recall [48]	Recall is complementary to precision. It captures how well the model can recall positive class, i.e., given all the positive predictions it makes, how many of them are indeed positive?	Used in classification tasks, especially where retrieving positive results is more important, e.g., e-commerce search and other information-retrieval tasks.
F1 score [49]	Combines precision and recall to give a single metric, which also captures the trade-off between precision and recall, i.e., completeness and exactness. F1 is defined as (2 × Precision × Recall) / (Precision + Recall).	Used simultaneously with accuracy in most of the classification tasks. It is also used in sequence-labeling tasks, such as entity extraction, retrieval-based questions answering, etc.
AUC [48]	Captures the count of positive predictions that are correct versus the count of positive predictions that are incorrect as we vary the threshold for prediction.	Used to measure the quality of a model independent of the prediction threshold. It is used to find the optimal prediction threshold for a classification task.
MRR (mean reciprocal rank) [50]	Used to evaluate the responses retrieved given their probability of correctness. It is the mean of the reciprocal of the ranks of the retrieved results.	Used heavily in all information-retrieval tasks, including article search, e-commerce search, etc.
MAP (mean average precision) [51]	Used in ranked retrieval results, like MRR. It calculates the mean precision across each retrieved result.	Used in information-retrieval tasks.
RMSE (root mean squared error) [48]	Captures a model's performance in a real-value prediction task. Calculates the square root of the mean of the squared errors for each data point.	Used in conjunction with MAPE in the case of regression problems, from temperature prediction to stock market price prediction.
MAPE (mean absolute percentage error) [52]	Used when the output variable is a continuous variable. It is the average of absolute percentage error for each data point.	Used to test the performance of a regression model. It is often used in conjunction with RMSE.

Intrinsic Evaluation

BLEU (bilingual evaluation understudy) [53]	Captures the amount of n-gram overlap between the output sentence and the reference ground truth sentence. It has many variants.	Mainly used in machine-translation tasks. Recently adapted to other text-generation tasks, such as paraphrase generation and text summarization.
METEOR [54]	A precision-based metric to measure the quality of text generated. It fixes some of the drawbacks of BLEU, such as exact word matching while calculating precision. METEOR allows synonyms and stemmed words to be matched with the reference word.	Mainly used in machine translation.
ROUGE [55]	Another metric to compare quality of generated text with respect to a reference text. As opposed to BLEU, it measures recall.	Since it measures recall, it's mainly used for summarization tasks where it's important to evaluate how many words a model can recall.
Perplexity [56]	A probabilistic measure that captures how confused an NLP model is. It's derived from the cross-entropy in a next word prediction task. The exact definition can be found at [56].	Used to evaluate language models. It can also be used in language-generation tasks, such as dialog generation.

Extrinsic Evaluation

Involves the business metrics outside the AI/ML team

First, check to see if you achieve good intrinsic evaluation metrics

Then, go for extrinsic evaluation

NLP Pipeline

Deployment

Deployment

NLP model is deployed as a web service*

Some Cloud providers

Google Cloud Platform (GCP)

Amazon Web Services (AWS)

Microsoft Azure

* Will be seen in detail in Full Stack Data Science Systems course

NLP Pipeline

Monitoring & Model Improvement

Monitoring & Model Updation

Monitoring must be done on a constant real-time basis

Performance dashboards to be included in the project

Project attribute	Decision paths	Examples
More training data is generated post-deployment.	Once deployed, extracted signals can be used to automatically improve the model. Can also try online learning to train the model automatically on a daily basis.	Abuse-detection systems where users flag data.
Training data is not generated post-deployment.	Manual labeling could be done to improve evaluation and the models. Ideally, each new model has to be manually built and evaluated.	A subset of a larger NLP pipeline with no direct feedback.
Low model latency is required, or model has to be online with near-real-time response.	Need to use models that can be inferred quickly. Another option is to create memoization strategies like caching or have substantially bigger computing power.	Systems that need to respond right away, like any chatbot or an emergency tracking system.
Low model latency is not required, or model can be run in an offline fashion.	Can use more advanced and slower models. This can also help in optimizing costs where feasible.	Systems that can be run on a batch process, like retail product catalog analysis.

Further Reading

Practical Natural Language Processing

Soumya Vajjala, Anuj Gupta, Harshit Surana, Bodhisattwa Majumder