NRC·CNRC

NRC.CANADA.CA

Natural Language Processing

Essentials for working with text data

National Research Council, Canada; May 14th, 2024

National Research Council Canada

Conseil national de recherches Canada

Canada

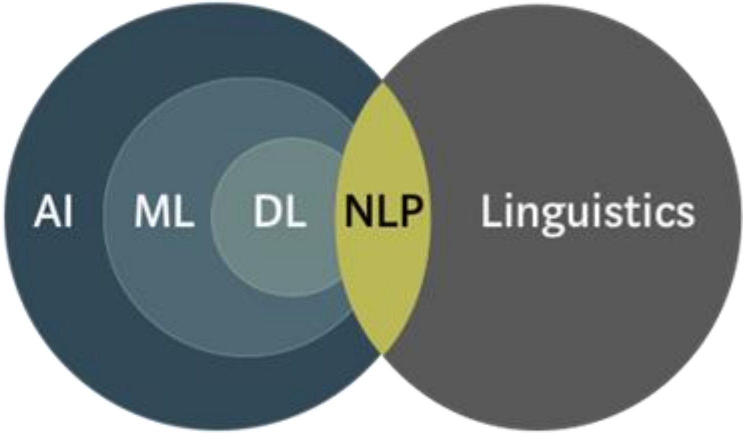
Outline

- What is NLP?
- What makes NLP challenging?
- Some common NLP tasks
- NLP Methods
- Some practical advice
- Code walk-through

What is NLP

What is NLP?

Natural Language Processing (NLP) is all about making computers understand and interact with humans, in their language(s).



source

Other related disciplines: cognitive science, human-computer interaction

Where is NLP useful?

It is a part of many day to day applications we use now

- Email filters, virtual assistants, information seeking, language translation etc.

There are so many business use cases where NLP plays a prominent role.

- Customer support, analytics, content generation, data protection etc.

It is also used across various disciplines to address domain specific questions

- E.g., analyzing political speeches for social scientists to protecting enterprise communications in cybersecurity experts

A brief history of NLP

First few decades: logic based language understanding systems, creating elaborate grammars to teach human language to computers, rule based systems, and automatic language generation.

Late 90s on: Advent of statistical methods and machine learning methods into NLP

2010s: Deep learning methods for NLP

2020s: LLMs, multimodality, generative AI, ethics, explainability etc.

What makes NLP challenging?

Language is Ambiguous

See these newspaper headlines:

- *"Children make delicious snacks"*
- *"Dead expected to rise"*
- *"Republicans grill IRS chief over lost emails"*

Normal, grammatical sentences can be ambiguous too:

- *"I saw a man on a hill with a telescope."*
- *"Look at the man with one eye"*

We are not even talking about ambiguities involving speech or alternative interpretations due to stress/emphasis on some word.

There are many forms of ambiguity

1. Lexical ambiguity: *I am at a bank vs I am at a river bank*

2. Structural ambiguity: *I saw the man on the hill with a telescope.*

3. Semantic ambiguity: *John and Mary are married (to each other? or to different people?)*

4. Referential ambiguity: *She dropped the plate on the table and broke it*

5. Ambiguity from non-literal language use: *Time flies like an arrow.*

(To read more: *http://cs.nyu.edu/faculty/davise/ai/ambiguity.html)*

World Knowledge

What is common knowledge for humans may not be so for a computer.

Dog bit man.
Man bit dog.

Linguistically, both of them are similar. But, we know only the first one is "normal" English sentence because we have "world knowledge". How can an NLP system/a computer know that?

Language is Diverse

What is “language?”

- Many different forms: News articles, tweets, logs, legal texts, chats, etc
- Creative use, and keeps changing over time.
- Many spelling variations, slangs, dialects, styles etc.
- Above all, thousands of languages in the world.

The challenge of NLP is to account for all these things and understand and process human language data!

Today, a language model can tell stories

In 50 words, tell me the story of a Tomato that wanted to become a Pineapple.

Tommy the tomato aspired to be a pineapple, enamored by its spiky exterior and golden hue. His efforts to change failed, and the wise old pumpkin urged him to accept his uniqueness and focus on enhancing his flavors instead. With this advice, Tommy found success as a beloved, plump tomato.

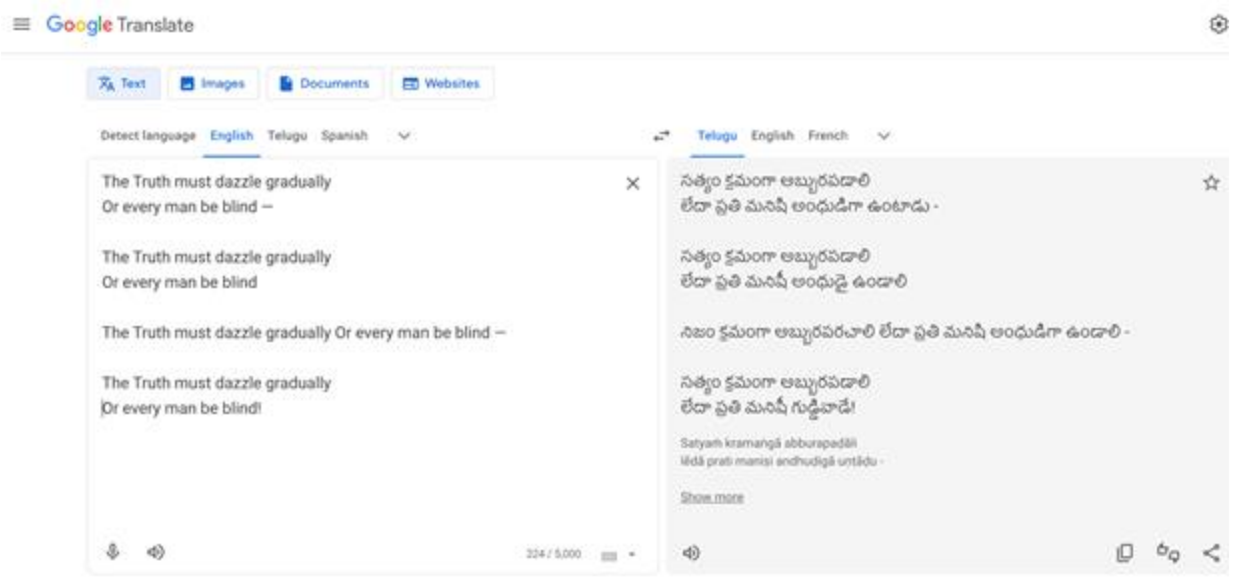
In 50 words, tell me the story of a Tomato that wanted to become a Pineapple, and indeed became one.

Against all odds, Tommy the tomato transformed into a vibrant pineapple with help from a magical spell cast by a kind witch. Overjoyed, Tommy basked in his new identity, thrilled by his spiky exoskeleton and sweet aroma. However, his joy soon faded when customers mistook him for a cactus due to his unusual appearance. Dejected, Tommy realized that fitting in wasn't always easy and returned to being a cherished tomato, embracing his originality and charm.

Using: https://huggingface.co/spaces/HuggingFaceH4/zephyr-chat

And yet, can make silly mistakes

Last two lines of an Emily Dickinson poem "Tell the truth, but tell it slant"



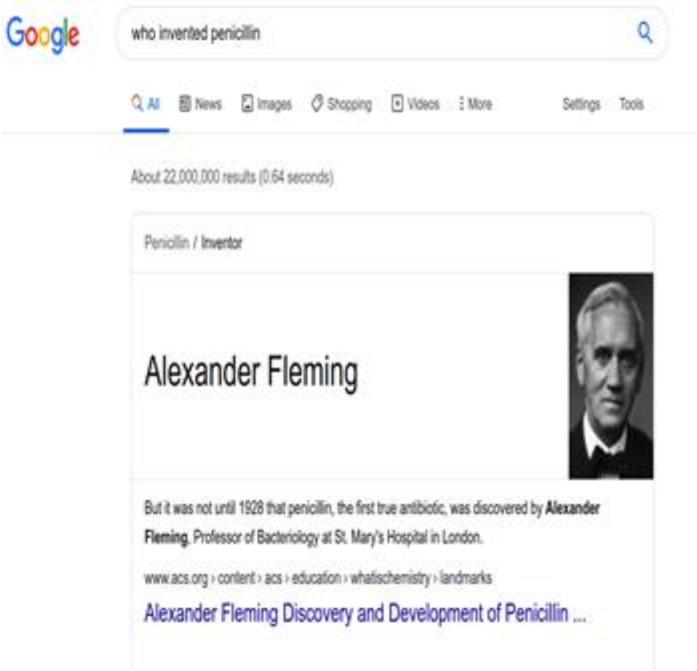
This is English -> Telugu, you can try in a language of your choice!

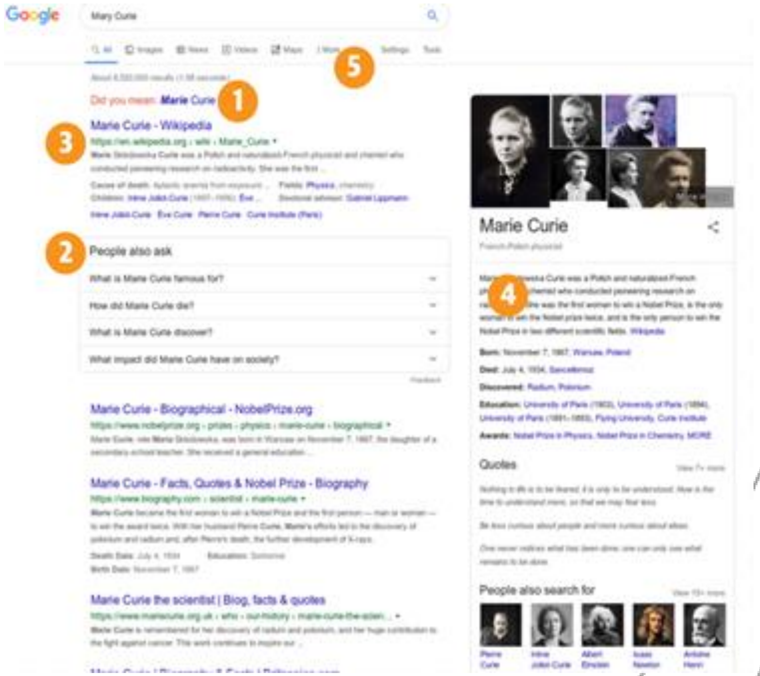
Is NLP "solved"?

- We have all seen a lot of buzz around large language models over the past 1.5 years.
- There are many tasks where NLP can amaze us today e.g., text generation, machine translation, being a human-like chatbot etc.
- **So, can all these challenges I described earlier be considered solved?**
- **What do you think?**

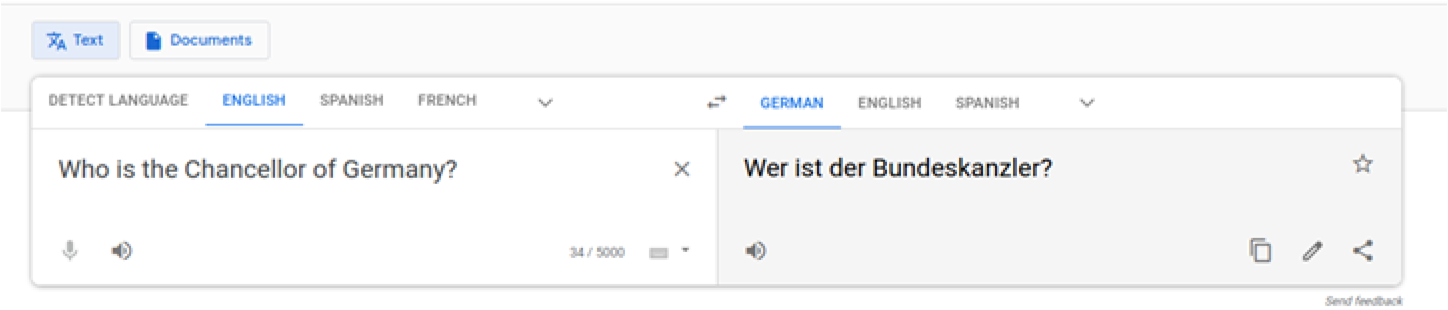
Some common NLP tasks

Search

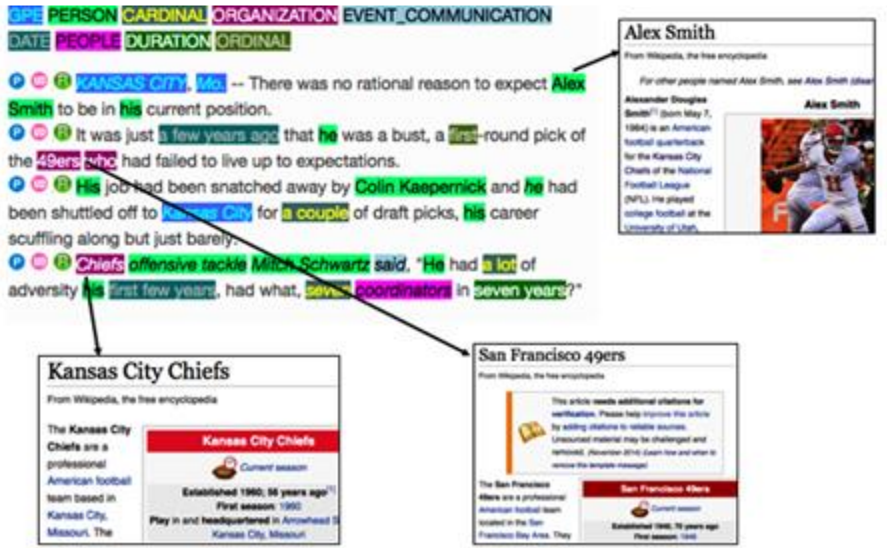

Google
who invented penicillin
All
News
Images
Shopping
Videos
More
Settings
Tools
About 22,000,000 results (0.64 seconds)
Penicillin / Inventor
Alexander Fleming
But it was not until 1928 that penicillin, the first true antibiotic, was discovered by Alexander Fleming, Professor of Bacteriology at St. Mary's Hospital in London.
www.acs.org › content › acs › education › whatischemistry › landmarks
Alexander Fleming Discovery and Development of Penicillin ...


Google
Mary Curie
1
2
3
4
5
Did you mean: Marie Curie
Marie Curie - Wikipedia
People also ask
What is Marie Curie famous for?
How did Marie Curie die?
Marie Curie - Biographical - NobelPrize.org
Marie Curie - Facts, Quotes & Nobel Prize - Biography
Marie Curie the scientist | Blog, facts & quotes
Marie Curie
Quotes
People also search for

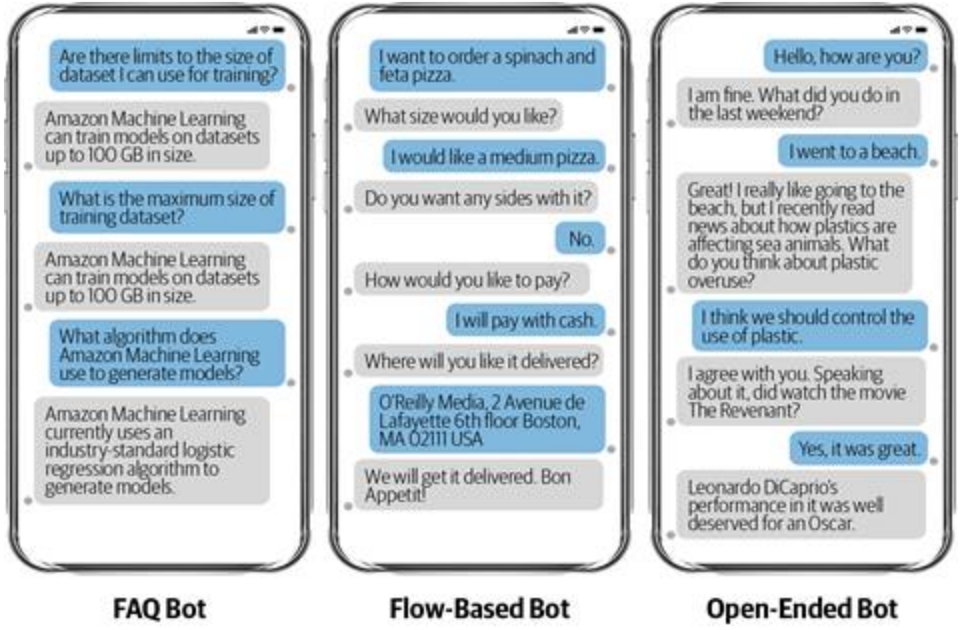
Machine Translation



Information Extraction

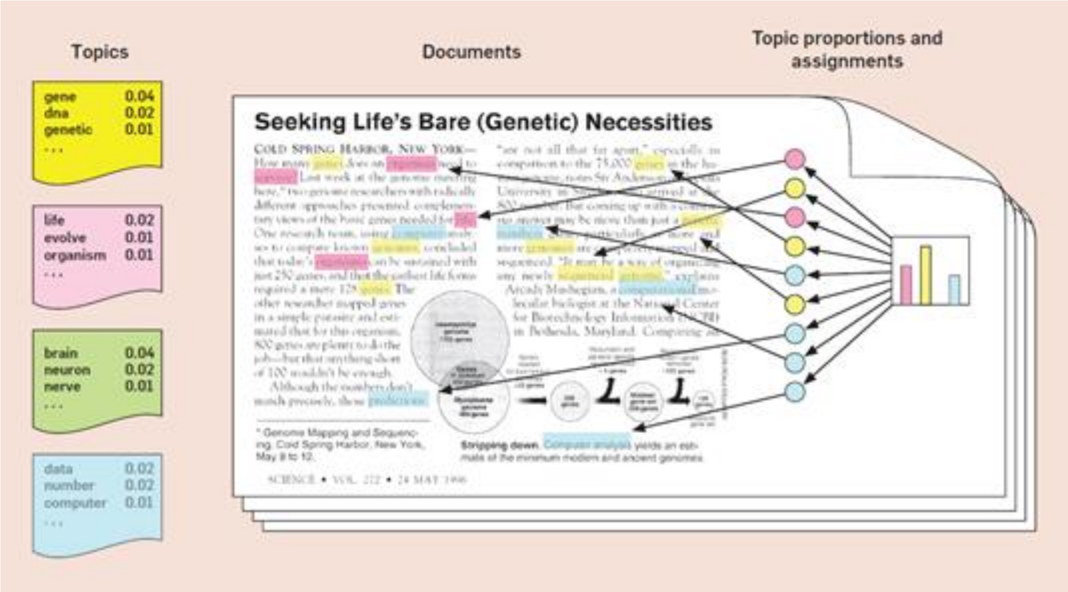


Chatbots



Are there limits to the size of dataset I can use for training?
Amazon Machine Learning can train models on datasets up to 100 GB in size.
What is the maximum size of training dataset?
Amazon Machine Learning can train models on datasets up to 100 GB in size.
What algorithm does Amazon Machine Learning use to generate models?
Amazon Machine Learning currently uses an industry-standard logistic regression algorithm to generate models.
FAQ Bot
I want to order a spinach and feta pizza.
What size would you like?
I would like a medium pizza.
Do you want any sides with it?
No.
How would you like to pay?
I will pay with cash.
Where will you like it delivered?
O'Reilly Media, 2 Avenue de Lafayette 6th floor Boston, MA 02111 USA
We will get it delivered. Bon Appetit!
Flow-Based Bot
Hello, how are you?
I am fine. What did you do in the last weekend?
I went to a beach.
Great! I really like going to the beach, but I recently read news about how plastics are affecting sea animals. What do you think about plastic overuse?
I think we should control the use of plastic.
I agree with you. Speaking about it, did watch the movie The Revenant?
Yes, it was great.
Leonardo DiCaprio's performance in it was well deserved for an Oscar.
Open-Ended Bot

Topic Modeling



This is not an exhaustive list, of course!

NLP Methods

How does one work with text data?

How do you solve an NLP problem?

- Rule based approach
- Feature engineering + machine learning based
- Deep learning based (no explicit feature engineering)
- LLM/GenAI based (no explicit model/learning phase either!)
- Combinations, and
- Following different approaches at different stages of problem solving

Rule-based NLP

- Encode the NLP task as a set of rules
- No “learning” involved
- No “training data” needed
- Here is a simple rule based system for predicting the sentiment of a text:

```
MySentimentAnalyzer(list_of_positive_words, list_of_negative_words, mytext):
        mywords = getlistofwords(mytext)
        sentpos = 0
        sentneg = 0
        for word in mywords:
                if word in list_of_positive_words:
                        Sentpos = sentpos+1
                elif word in list_of_negative_words:
                        Sentneg = sentneg+1
                else:
                        #do nothing
        If sentpos > sentneg:
                return “positive”
        elif sentpos < sentneg:
                return “negative”
        else:
                return “neutral”
```

Why bother? Is this still relevant?

- Yes! Many production systems use some form of rules somewhere along with machine/deep learning
- They are useful for edge cases, when we have a ton of domain knowledge but no labeled data, etc.
- A 2021 technical report described how Facebook used regular expressions to determine whether a post is about COVID-19

Facebook’s regex text classifier

They built two sets of pattern matching based rules:

(1) for 66 languages, with 99% precision and recall >50%,

(2) for the 11 most common languages, with precision >90% and recall >90%

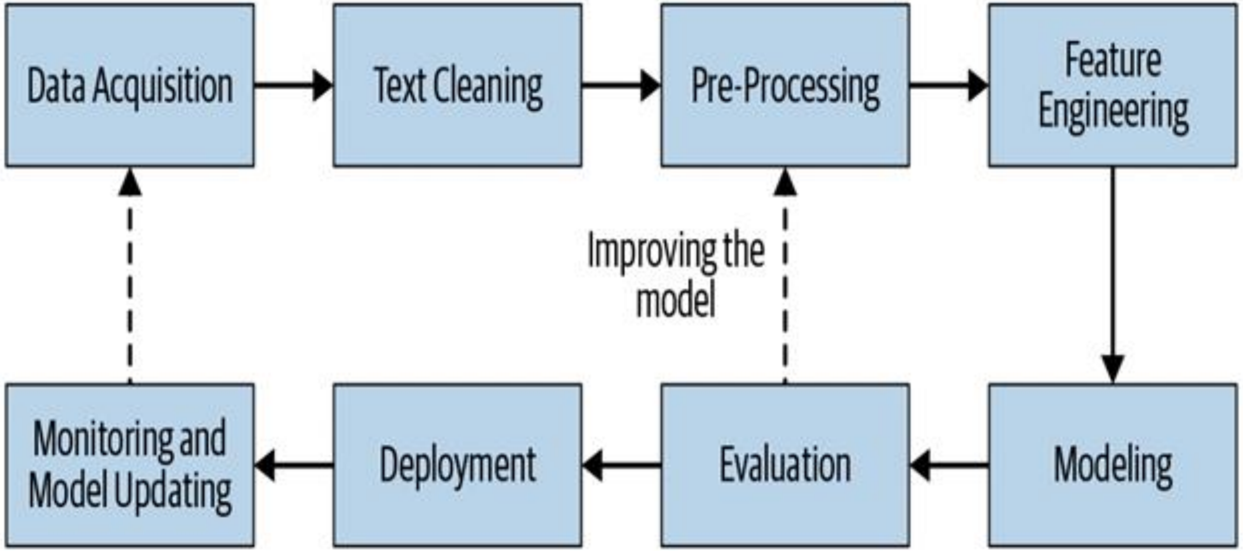
Comparisons to a DNN classifier after collecting manual labeled data showed explainable results, higher precision and recall, and less overfitting for the rule-based approach.

So, this sort of an approach can still give a practical first solution in today’s world!

Traditional Methods are still relevant

- Despite their impressive performance, Large Language Models (LLMs) can face challenges in maintaining reliability and thoroughness, particularly within specialized domains or tasks
- Traditional NLP techniques provide a structured and domain-specific approach that can augment the capabilities of LLMs, addressing their limitations in certain contexts

A typical NLP pipeline



How to get data

- Use existing NLP datasets, if suitable (e.g., huggingface.co/datasets)
- Scrape data from the web, where allowed
- Use available organizational data (logs, internal documents etc)
- Set up data annotation experiments and collect data
- Do synthetic labeling of large amounts of data, with small amount of manually checked/labeled data for quality assurance

Text Extraction and Cleaning

- Often ignored, but crucial step
- What is the issue?: extracting all sorts of data from different formats

  (images, pdfs, invoices, html, docx etc) is non-trivial
- It is not “NLP” per se, but it defines what happens with your NLP system

  later!

Text Pre-processing

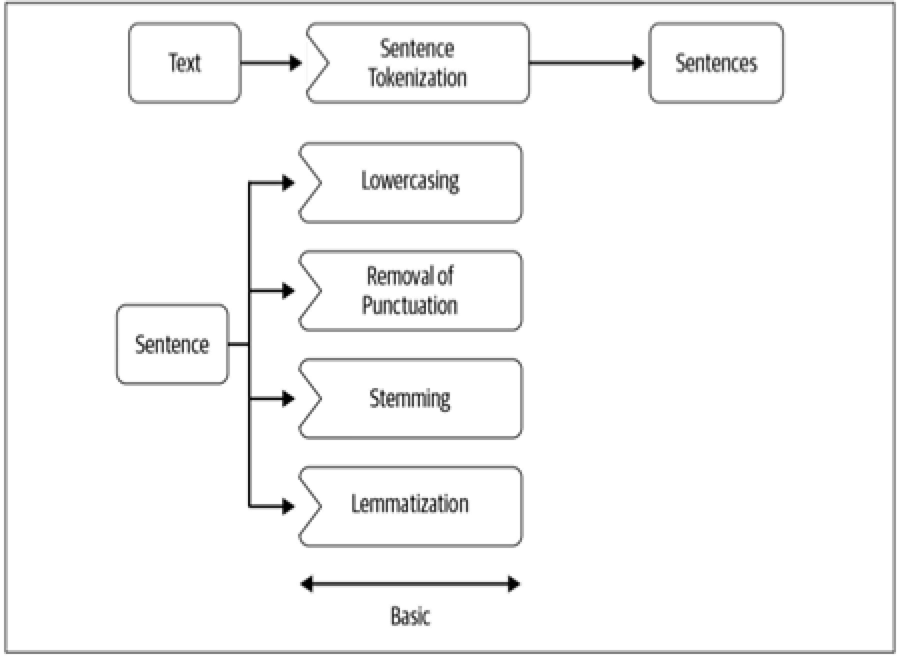


Figure 2-8. Common pre-processing steps on a blob of text

Figure 2-7. Difference between stemming and lemmatization [33]

Feature Engineering: Text Representation

- To build a solution for any kind of NLP task, we first need a way to represent text numerically.

How do we represent text numerically?

Text as a bag of words

TF-IDF

TF*IDF: Text as a bag of words, but encoding some notion of word importance.

TF(t,d)= (Number of occurrences of term t in document d)/(Total number of terms in the document d)

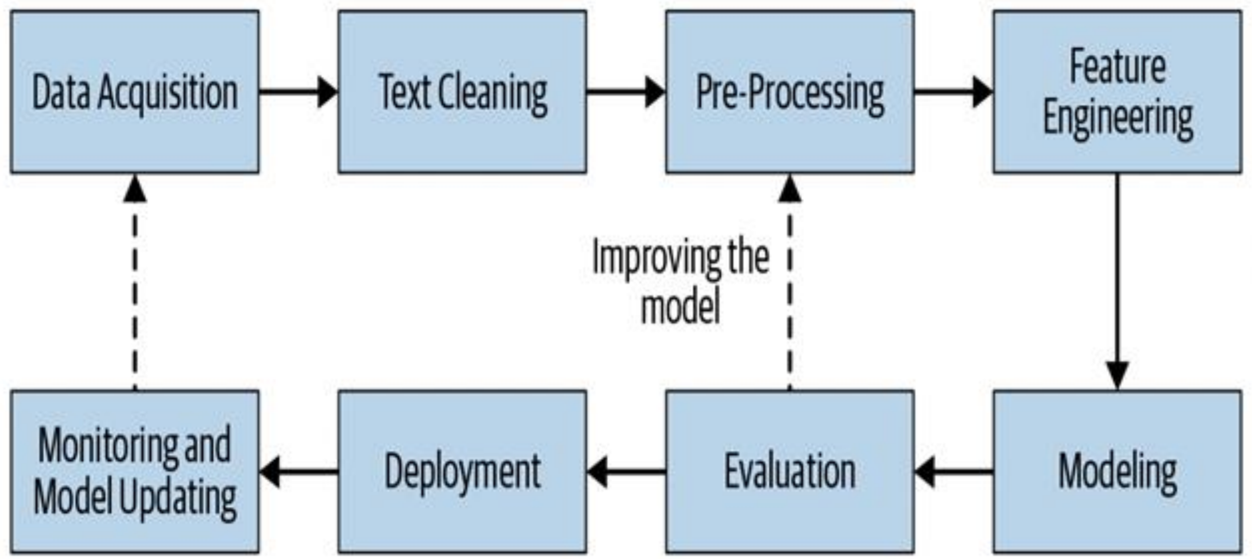
IDF(t)=log (Total number of documents in the dataset)/(Number of documents with term t in them)

The tf-idf weighting based text representation is a great baseline for doing any NLP stuff!

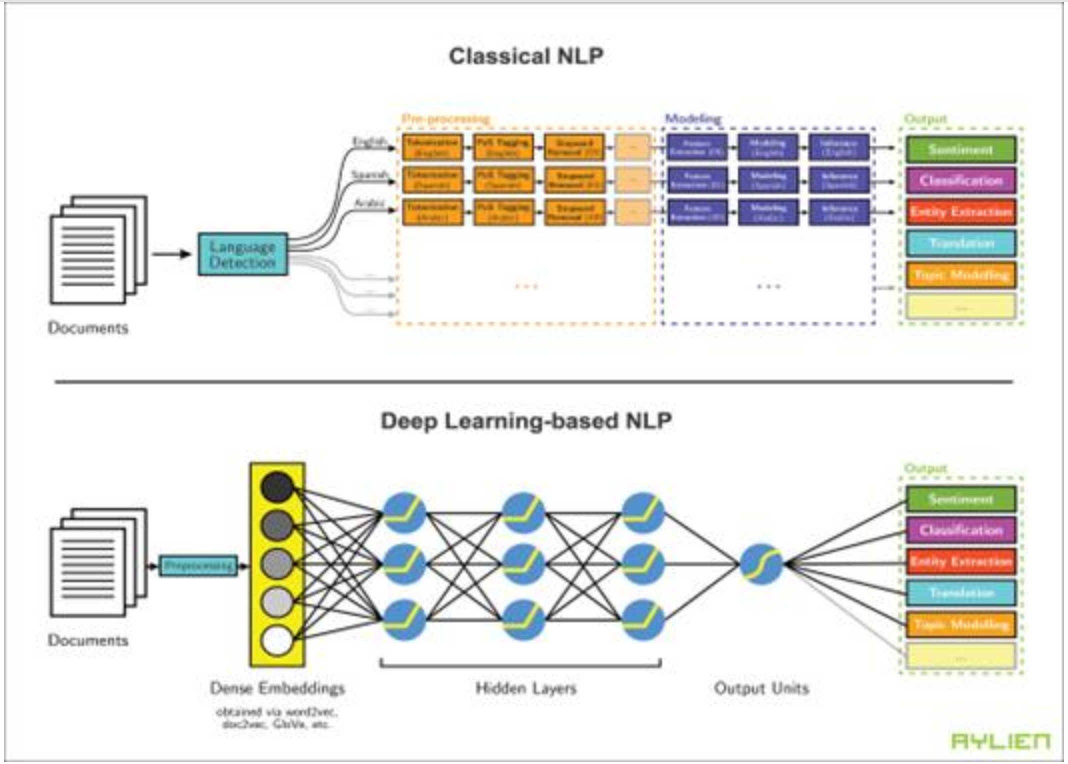
Neural Network model based representation

- Idea: Similar words are likely to occur in similar contexts
- If we're given the word "USA," distributionally similar words could be other countries (e.g., Canada, Germany, India, etc.) or cities in the USA
- If we're given the word "beautiful" words that share some relationship with this word (e.g., synonyms, antonyms) could be considered distributionally similar words
- Modern day NLP is based on text representations which **learn** such semantic relationships in a dense, low dimensional space (compared to sparse, high dimensional space we saw earlier), called "text embeddings"

Let us pause and look at this again



Modeling: learning a model from scratch



Source: Ayilen

Modeling: Fine-Tuning pre-trained models

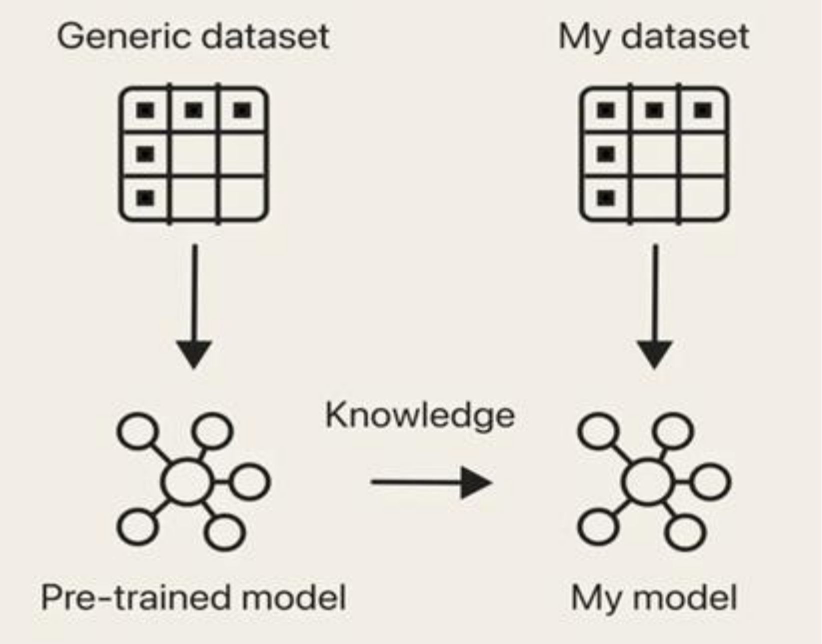
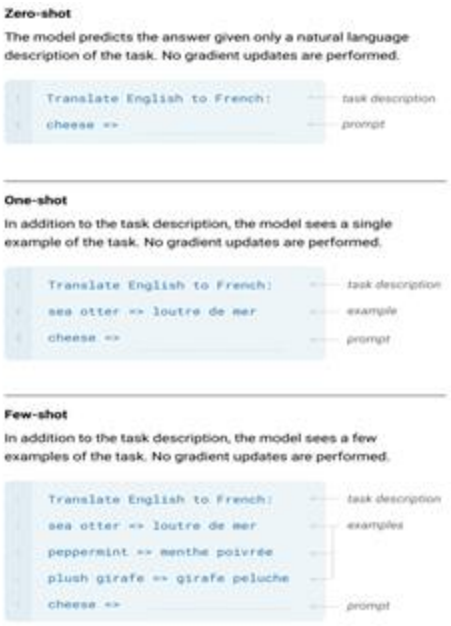


Image source:
Towards Data Science blog post
by Leonie Monigatti, February 2023

A popular "pre-trained" model is "BERT".

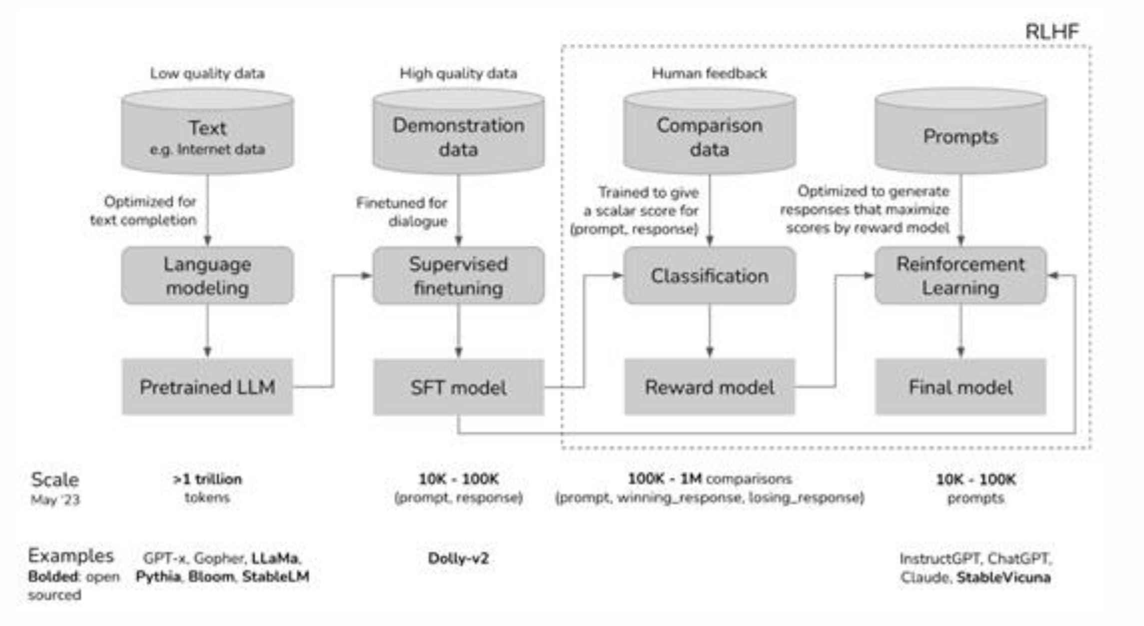
Beyond model building: in-context learning



Today's Large language models may not even need large datasets for fine-tuning. They may just be able to learn a task from a few examples.

Source: GPT-3 paper from OpenAI (2020)

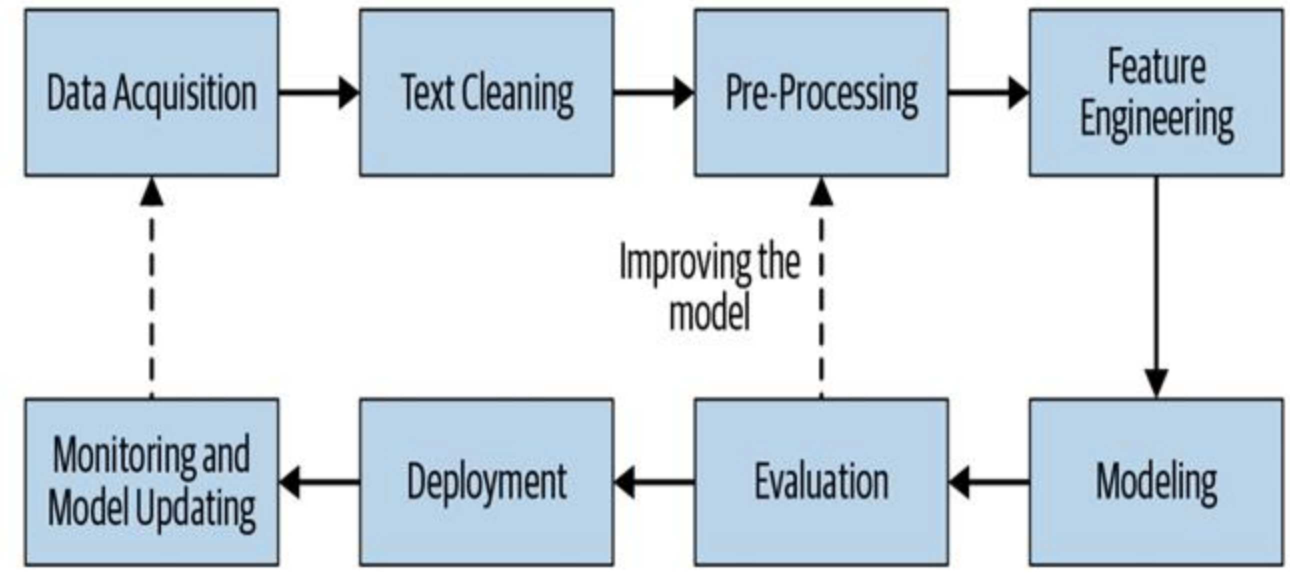
How are LLMs built?



- Some learning from plain text,
- followed by learning to follow human instructions,
- and then, further learning to match human preferences

Source: https://huyenchip.com/2023/05/02/rlhf.html

Let us get back to this figure



Evaluation

- Intrinsic - focuses on intermediary objectives
- Extrinsic - focuses on the final objective

Consider a email spam classification system:

- Intrinsic evaluation: assessing the system performance using measures such as precision and recall on a test set
- Extrinsic evaluation: time a user wasted because a spam email went to their inbox or vice versa

Deploy and Monitor Models

- Where is your solution hosted? (on premise, on cloud, on a device etc.)
- Costs involved
- Speed
- Long term solution
- Updating the model
- Updating the data for train/test etc

Some practical advice

How should you get started?

- Build a small (high-quality, human) labeled data set to test your solution(s).
- Decide how/what to evaluate performance on.
- Try zero-shot/few-shot approaches first using an LLM.
- At a slightly mature stage, collect more data, and fine-tune your NLP model
- Explore more advanced NLP modeling methods as you build necessary custom datasets over time.

What LLMs for off-the-shelf use?

- There are big providers like OpenAI, Cohere, Anthropic, Google etc that host commercial APIs with massive models like GPT4 etc.
- There are many many open source LLMs of various sizes available now, and they may be as good or better for your specific task - explore those too!
- How?: libraries like **ollama** lets you work with some open LLMs locally, on your laptops too (conditions apply!)

Note: Working example on this aspect is provided for text classification with

Using LLMs for data generation

- Beyond using LLMs for zero-shot scenarios, one important way to use LLMs is for synthetic data generation
- What purpose does it serve?: Address data scarce scenarios, augment existing data to create a big enough data to fine-tune our own models.
- Caveats: Don't automate evaluation too. Keep your valuable human labeled data for evaluation.

Note: Working example on this aspect is provided for text classification

Some issues to keep in mind

- Short and long term costs of using a commercial LLM
- Time taken to run a model and get output and how this works out in practical use cases.
- Format of the output, and consistency of the generated output. (despite instructions, LLMs can sometimes generate output in a different format)
- Hallucinations in LLM output (unexpected text in output)

Beyond proprietary LLMs

- There is a lot of interest now on hosting solutions locally, using smaller models, building fine-tuned and maintainable models that do the needed task well, rather than being generally good on all sorts of things.

If situation demands, be ready to explore rules and pattern-matching too!
Remember: GenAI, LLMs, ML, DL etc are all tools - not solutions.

So,

- Know the data requirements and plan for building the datasets first
- Build an evaluation strategy before building the model
- Understand options and costs to build/maintain an NLP system
- Acknowledge potential limitations of the current state of the art
- Gather information on the cutting edge, but don’t expect that to always do better than established practice by default (**Important**!)

Remember …

- We don’t always need a large language model or the most advanced research for all of our language processing problems.
- All we need is a solution that does the required job at the required degree of performance consistently, and reliably.
- Don’t ignore human judgement and intervention where needed.

use NLP/LLMs/GenAI mindfully!

Let us see a few examples of how to use LLMs at different stages of building an NLP model

Example 1: Problem Description

- Classify tweets into one of the 6 categories: "arts_&_culture", "business_&_entrepreneurs", "pop_culture", "daily_life", "sports_&_gaming", "science_&_technology”
- I will use an available dataset (Cardiff Twitter Topic Classification) with some labeled train/test data, and use it in the following scenarios:
  - Use LLMs as text classifiers (via prompting) and evaluate how good they are at it, using the test partition
  - Use the training data and build a regular classifier, but use LLMs to generate some synthetic data which can help classification performance.

How the data looks like

Split (13)

train_coling2022_random · 3.6k rows

Search this dataset

text (string · lengths)	date (string · lengths)	label (class label)	label_name (string · classes)
Jalen hurts first drive low key lookin really good but MALCOM COMIN UP W THE BIG STOP! {{USERNAME}}	2020-12-13	4 sports_&_gaming	sports_&_gaming
Sometimes The Bad Things That Happen In Our Lives Put Us Directly On The Best Things That Will Ever Happen To Us. IWANTASAP WITH ELLA #ASAPFeelGood | {{USERNAME}}	2021-05-16	3 daily_life	daily_life
I request all the country men to give holidays to your servant s who are working at your house, residence, bangalows. By this we can stop spreading of covid19. A work yourself in your houses…	2020-03-22	3 daily_life	daily_life
Quicker end than expected. Had a blast {@USA Softball@} with these girls and coaches. Teammates for life! #softball #memories	2020-08-16	4 sports_&_gaming	sports_&_gaming
The first #musicofthefuturevideooftheweek is now live featuring {{USERNAME}} and you can watch the full video of #LoneWolf on {{URL}} Watch it,subscribe and follow Also #intuition is droppin…	2020-01-05	2 pop_culture	pop_culture
Breaking Bad: El Camino s Rotten Tomatoes Score Is Far From Bad {{URL}} #BreakingBad #RottenTomatoes {@Netflix@}	2019-10-13	2 pop_culture	pop_culture
Watching Harry Potter and The Prisoner of Azkaban and I want {@Alfonso Cuaron@} to go back and remake all of the other Harry Potter films please.	2020-04-05	2 pop_culture	pop_culture
Appreciation tweet for Love In The Air by {@GFRIEND@} . One of their best choices :3 {{URL}}	2021-01-17	2 pop_culture	pop_culture
Leonard Fournette carves up the Chiefs D to make it 28-9 #EDFsports {{USERNAME}} {{USERNAME}} {{USERNAME}} {{USERNAME}} {{USERNAME}}	2021-02-08	4 sports_&_gaming	sports_&_gaming
I made sourdough bread! {{USERNAME}} supervised. Also had a little Easter dinner. #baking #easter {{URL}}	2020-04-12	3 daily_life	daily_life
Aye james harden was hardly on twitter during his rockets stint, went to Nets and was tweeting "Scary Hours" after every win…will he tweet again or will he go back to not being on…	2021-06-20	4 sports_&_gaming	sports_&_gaming
Chelsea loosing to Leicester at the FA cup final wasn t the saddest event yesterdays. But loosing my jovial uncle, who was a staunch Chelsea fan the same night after the game remains a…	2021-05-16	4 sports_&_gaming	sports_&_gaming
BTW the spots by {@FOX Sports@} are pushing hard for the Dodgers.... Come on {@Tampa Bay Rays@} get the W and prove you belong!!!!	2020-10-25	4 sports_&_gaming	sports_&_gaming

Using LLMs as text classifiers - 1

```
from transformers import T5Tokenizer, T5ForConditionalGeneration

from datasets import load_dataset

from datetime import datetime


tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-base")

model = T5ForConditionalGeneration.from_pretrained("google/flan-t5-base")


dataset = load_dataset("cardiffnlp/tweet_topic_single")["test_coling2022"]

cats = ",".join(["arts_&_culture", "business_&_entrepreneurs", "pop_culture",
"daily_life", "sports_&_gaming",

       "science_&_technology"])

prompt = "You are a topic classifier that classifies the given input as one of the
following 6 categories:" + cats + "Just return the output, without any explanation. Here
is the input:  "
```

FlanT5 as a text classifier

```
num_correct = 0
start = datetime.now()
for i in range(0,dataset.num_rows):
   input_ids = tokenizer(prompt+dataset[i]["text"], return_tensors="pt").input_ids
   outputs = model.generate(input_ids)
   pred = tokenizer.decode(outputs[0]).split("> ")[1].split("<")[0]
   print(pred, dataset[i]["label_name"])
   if pred == dataset[i]["label_name"]:
       num_correct += 1
end = datetime.now()
print(end-start) #time taken for ~3500 samples
print(num_correct/dataset.num_rows) #cases where the model assigned the correct label
```

Took ~10 min 30 seconds; got 76% agreement between human labels and model labels, and I could run locally on my laptop.

Using LLMs as Text Classifiers - 2 (OpenAI)

```
#import the required libraries
from openai import OpenAI
from datasets import load_dataset
from datetime import datetime
import os
#initialize the OpenAI client
client = OpenAI(api_key=os.getenv("MY_OPENAI_KEY"))
#load the dataset from huggingface
dataset = load_dataset("cardiffnlp/tweet_topic_single")["test_coling2022"]
numcorrect = 0 #numcorrect collects the number of cases where the model assigned the
correct label
start = datetime.now()#start is used to measure the time taken for the model to run
```

OpenAI Models as Text Classifiers

```
#iterate over the dataset sending requests to openai and collecting the output
for i in range(0,dataset.num_rows):
  response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[
      {
        "role": "system",
        "content": "You are a topic classifier that classifies the given input as one of the following 6 categories: "
                   "\"arts_&_culture\", \"business_&_entrepreneurs\", \"pop_culture\", \"daily_life\", "
                   "\"sports_&_gaming\", \"science_&_technology\""
                   "Just return the output, without any explanation. Here is the input:  \""
      },
      {
        "role": "user",
        "content": dataset[i]["text"]
      },
    ],
    temperature=1,
    max_tokens=256,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
  )
```

```
    print(response)
    print(response.choices[0].message.content, dataset[i]["label_name"])
     if response.choices[0].message.content == dataset[i]["label_name"]:
         numcorrect += 1
    #the above lines were a part of the for loop from previous slide.

print(numcorrect)
end = datetime.now()
print(end-start) #time taken for ~3500 samples
#cases where the model assigned the correct label
print(numcorrect/dataset.num_rows)
```

Took ~30 min to run; got 72% agreement between human labels and model labels, and OpenAI labeling cost was around 1 $.

A summary showing other models

Model	Size (parameters)	Accuracy	Run time	Added cost
Flan-T5-Base	248M	77%	~10min	-
Flan-T5-Large	783M	70%	~33min	-
Flan-T5-XL	2.85B	73%	~1.75 hours	-
GPT-3.5-turbo	??	72%	~30 min	0.7 USD
GPT-4	??	83%	~ 1 hr	14 USD

Learnings

- Using LLMs for data annotation can potentially reduce labeling costs for some kind of problems.
- Using more than one LLM and looking at their agreements and disagreements can be a strategy for deciding on what to send to human labelers.
- Note: There are many LLMs that can potentially run on our laptops (and even mobile devices) now. Feel free to explore!

Caveats

- I did not do elaborate experiments.
- Only looked at classification accuracy, but the label distribution is not exactly balanced.
- No error analysis done, so I don’t know what labels are confused the most etc.
- I only tried with one prompt.
- Most importantly: I have a means of evaluating this experiment (i.e., I know the human labels too!)

How is this useful, actually?

- We can use LLMs as is and be happy.
- We can bootstrap with LLM generated predictions and move towards building larger models that can be finetuned and adapted further.

LLMs can help human labelers

One process that can work in real-world scenarios in terms of time and cost savings is:

- Use LLMs as the first step for annotation.
- Go through manual annotation on the full data or a sample, using LLM annotation as the input (instead of raw text).
- If LLMs are bad annotators for a given task, there are humans to chip in [and your cost/time goes up accordingly, but so does the data quality]
- If LLMs are doing a good job, they reduce human effort on that task!

Let us look at another scenario

- Let us say we either have substantial labeled data to start with, or we reached a point where we built a dataset (going from Solution 0 to Solution 1) and we now want to get better (Solution 2).
- In the case of our dataset, the original training data looks like this:

```
Counter({'sports_&_gaming': 1217,
         'pop_culture': 1378,
         'arts_&_culture': 57,
         'daily_life': 605,
         'business_&_entrepreneurs': 151,
         'science_&_technology': 190})
```

What is one issue with this dataset?

Let me quickly build a text classifier

```
from datasets import load_dataset
#download the dataset from huggingface
train_dataset = load_dataset("cardiffnlp/tweet_topic_single")["train_coling2022"]
test_dataset = load_dataset("cardiffnlp/tweet_topic_single")["test_coling2022"]
train_texts, train_labels = train_dataset["text"], train_dataset["label"]
test_texts, test_labels = test_dataset["text"], test_dataset["label"]

#Feature extraction
from sentence_transformers import SentenceTransformer
transformer = SentenceTransformer('all-MiniLM-L6-v2') -> this model occupies only 80MB on hard disk!
train_vectors = transformer.encode(train_texts)
test_vectors = transformer.encode(test_texts)
```

The Classifier

```python
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.linear_model import LogisticRegression

baseline_model = LogisticRegression(max_iter=100)  # note we first re-instantiate the model
baseline_model.fit(X=train_vectors, y=train_labels)

preds = baseline_model.predict(test_vectors)
acc_og = accuracy_score(test_labels, preds)
print(f"\n Test accuracy of original model: {acc_og}")
print(classification_report(test_labels, preds, target_names=just_labels))
```

Test accuracy of original model: 0.8281847602235952

	precision	recall	f1-score	support
arts_&_culture	1.00	0.03	0.06	95
business_&_entrepreneurs	0.74	0.55	0.63	166
pop_culture	0.86	0.90	0.88	1360
daily_life	0.52	0.71	0.60	356
sports_&_gaming	0.94	0.94	0.94	1266
science_&_technology	0.69	0.40	0.50	156
accuracy			0.83	3399
macro avg	0.79	0.59	0.60	3399
weighted avg	0.84	0.83	0.82	3399

We have a problem

Idea: Data Augmentation

- We can use LLMs to create synthetic data for categories with less amount of data.
- Why?: Boost the performance on arts_and_culture
- How do we do it?

Step 1: Get the indices of texts with this label (indicated by the integer 0 in the dataset)

```
# We are doing pretty badly with Arts and Culture. How about trying to improve it a bit?
indices_needed = [i for i in range(0, len(train_labels)) if train_labels[i] == 0]
# These need to be augmented, and the model needs to be re-trained.
```

Use an LLM and prompt it

```
import ollama #-> my favorite python library to run LLMs locally.

model = "mistral"
# build a prompt:
prompt = """You are a text data augmenter that can generate variations of the given input text
            without loss of meaning. Variations can be generated by replacing words with their
synonyms,
            or by replacing words with words that are similar in meaning or paraphrasing.
            Just return the output, without any explanation. Generate only one variation.
            Here is the input:
        """
```

Perform the data augmentation

```
train_texts_aug = train_texts[:]
train_labels_aug = train_labels[:]
for myindex in indices_needed:
    mytext = train_texts[myindex]
    response = ollama.chat(model='mistral', messages=[{'role': 'system', 'content': prompt},
                                                      {'role': 'user', 'content' : mytext}])
    myparaphrases = response['message']['content'].split("\n")
    #print(myparaphrases[0])
    train_texts_aug.append(myparaphrases[0])
    train_labels_aug.append(train_labels[myindex])
```

Retrain with the augmented data

```
#Train another classifier with the new augmented data
 transformer = SentenceTransformer('all-MiniLM-L6-v2')

 train_vectors = transformer.encode(train_texts_aug)
 test_vectors = transformer.encode(test_texts)

 new_model = LogisticRegression(max_iter=100)  # note we first re-instantiate the model
 new_model.fit(X=train_vectors, y=train_labels_aug)

 new_preds = new_model.predict(test_vectors)
 acc_og = accuracy_score(test_labels, new_preds)
 print(f"\n Test accuracy of new model: {acc_og}")
```

```
Test accuracy of new model: 0.8320094145336864
```

Compare old and new models

```
Test accuracy of original model: 0.8281847682235952
                          precision    recall  f1-score   support

           arts_&_culture       1.00      0.83      0.86        95
business_&_entrepreneurs       0.74      0.55      0.63       166
             pop_culture       0.86      0.90      0.88      1360
              daily_life       0.52      0.71      0.60       356
          sports_&_gaming       0.94      0.94      0.94      1266
    science_&_technology       0.69      0.40      0.50       156

                accuracy                           0.83      3399
               macro avg       0.79      0.59      0.60      3399
            weighted avg       0.84      0.83      0.82      3399
```

```
print(classification_report(test_labels, new_preds, labels=[0,1,2,3,4,5]))
```

```
              precision    recall  f1-score   support

           0       0.65      0.21      0.32        95
           1       0.73      0.55      0.63       166
           2       0.86      0.89      0.88      1360
           3       0.53      0.71      0.60       356
           4       0.94      0.94      0.94      1266
           5       0.69      0.39      0.50       156

    accuracy                           0.83      3399
   macro avg       0.73      0.62      0.64      3399
weighted avg       0.84      0.83      0.83      3399
```

Summary

- adding a few augmented examples (I added 57 examples in this process) resulted in a much better performance for that category, without losing performance on others.
- I tried only one method, of course, and with one LLM.
- You can try this approach changing LLMs, generating more examples, trying other prompts or using other augmentation methods (e.g., back translation)

What next?

- We will stop here, for now.
- There are a couple of Python notebooks associated with the lecture slides
- … and two more (one doing data augmentation with “back translation”, and another on using an open LLM for RAG)
- Explore those based on need and interest
- Contact us if you need more information or have questions!

Some free online resources

- Short, introductory courses from deeplearning.ai
- Huggingface courses
- Minaee et.al. (2024) - LLMs: A survey
- Numbers every LLM developer should know
- LLM Course by Maxime Labonne
- Advanced NLP - a free online course from SpaCy

Other Resources

Traditional NLP: Practical Natural Language Processing, O'Reilly Media, by Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, Harshit Surana (2020)

Deep Learning based NLP:

1. Natural Language Processing with Transformers, O'Reilly Media, by Lewis Tunstall, Leandro von Werra, Thomas Wolf (2022)
2. Speech and Language Processing, online edition, by Jurafsky and Martin (2023)

LLMs and NLP:

1. Upcoming book: Building a Large Language Model (from scratch) by Sebastian Raschka, Manning Publications
2. Upcoming book: Designing Large Language Model Applications by Suhas Pai, O'Reilly Media
3. Upcoming book: Hands-on Large Language Models by Jay Alammar and Marten Grootendorst, O'Reilly Media

NRC·CNRC

NRC.CANADA.CA •

THANK YOU

Sowmya.vajjala
Isar.nejadgholi
Khadija.osman
@nrc-cnrc.gc.ca