

Question Answering and Dialogue Systems

1121AITA09

MBA, IM, NTPU (M5265) (Fall 2023)

Tue 2, 3, 4 (9:10-12:00) (B3F17)



<https://meet.google.com/miy-fbif-max>



Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>



Syllabus

Week Date Subject/Topics

- 1 2023/09/13 Introduction to Artificial Intelligence for Text Analytics
- 2 2023/09/20 Foundations of Text Analytics:
Natural Language Processing (NLP)
- 3 2023/09/27 Python for Natural Language Processing
- 4 2023/10/04 Natural Language Processing with Transformers
- 5 2023/10/11 Case Study on Artificial Intelligence for Text Analytics I
- 6 2023/10/18 Text Classification and Sentiment Analysis

Syllabus

Week Date Subject/Topics

7 2023/10/25 Multilingual Named Entity Recognition (NER)

8 2023/11/01 Midterm Project Report

9 2023/11/08 Text Similarity and Clustering

10 2023/11/15 Text Summarization and Topic Models

11 2023/11/22 Text Generation with Large Language Models (LLMs)

12 2023/11/29 Case Study on Artificial Intelligence for Text Analytics II

Syllabus

Week Date Subject/Topics

13 2023/12/06 Question Answering and Dialogue Systems

14 2023/12/13 Deep Learning, Generative AI, Transfer Learning,
Zero-Shot, and Few-Shot Learning for Text Analytics

15 2023/12/20 Final Project Report I

16 2023/12/27 Final Project Report II

17 2024/01/03 Self-learning

18 2024/01/10 Self-learning

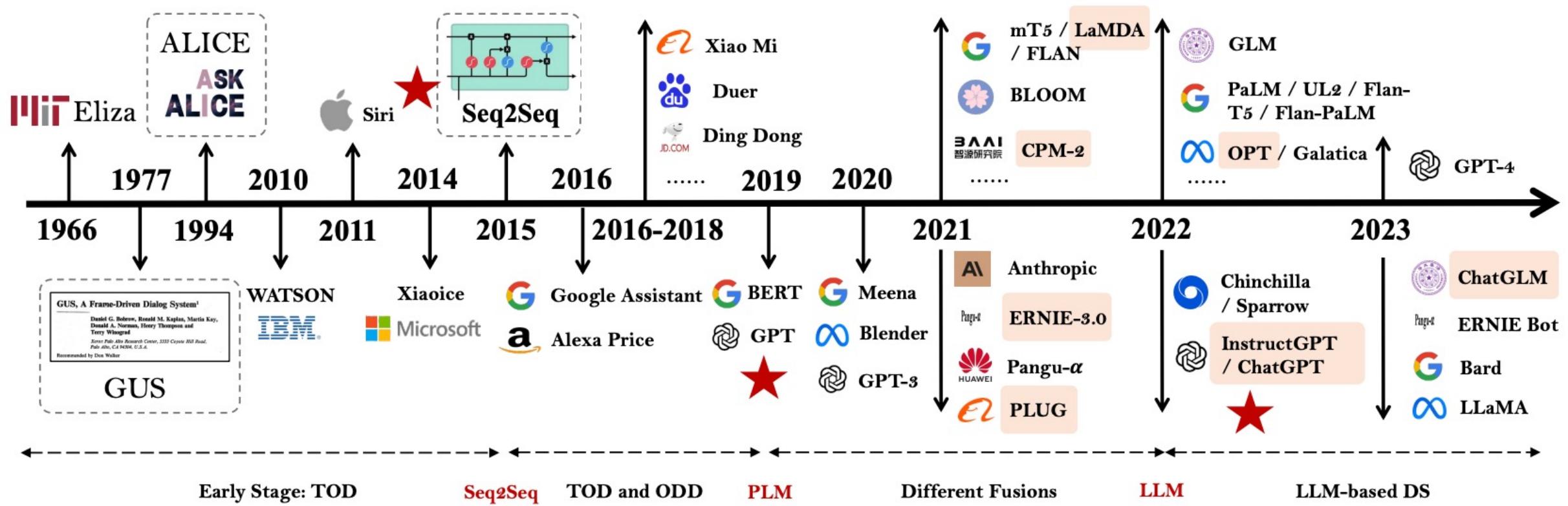
Question Answering and Dialogue Systems

Outline

- Question Answering
- Dialogue Systems
 - Task Oriented Dialogue (TOD) System
- Task-Oriented Dialogue System (TDS)
- Conversational Dialogue System (CDS)
- Question Answering Dialogue System (QADS)

The Development of LM-based Dialogue Systems

- 1) Early Stage (1966 - 2015)
- 2) The Independent Development of TOD and ODD (2015 - 2019)
- 3) Fusions of Dialogue Systems (2019 - 2022)
- 4) LLM-based DS (2022 - Now)



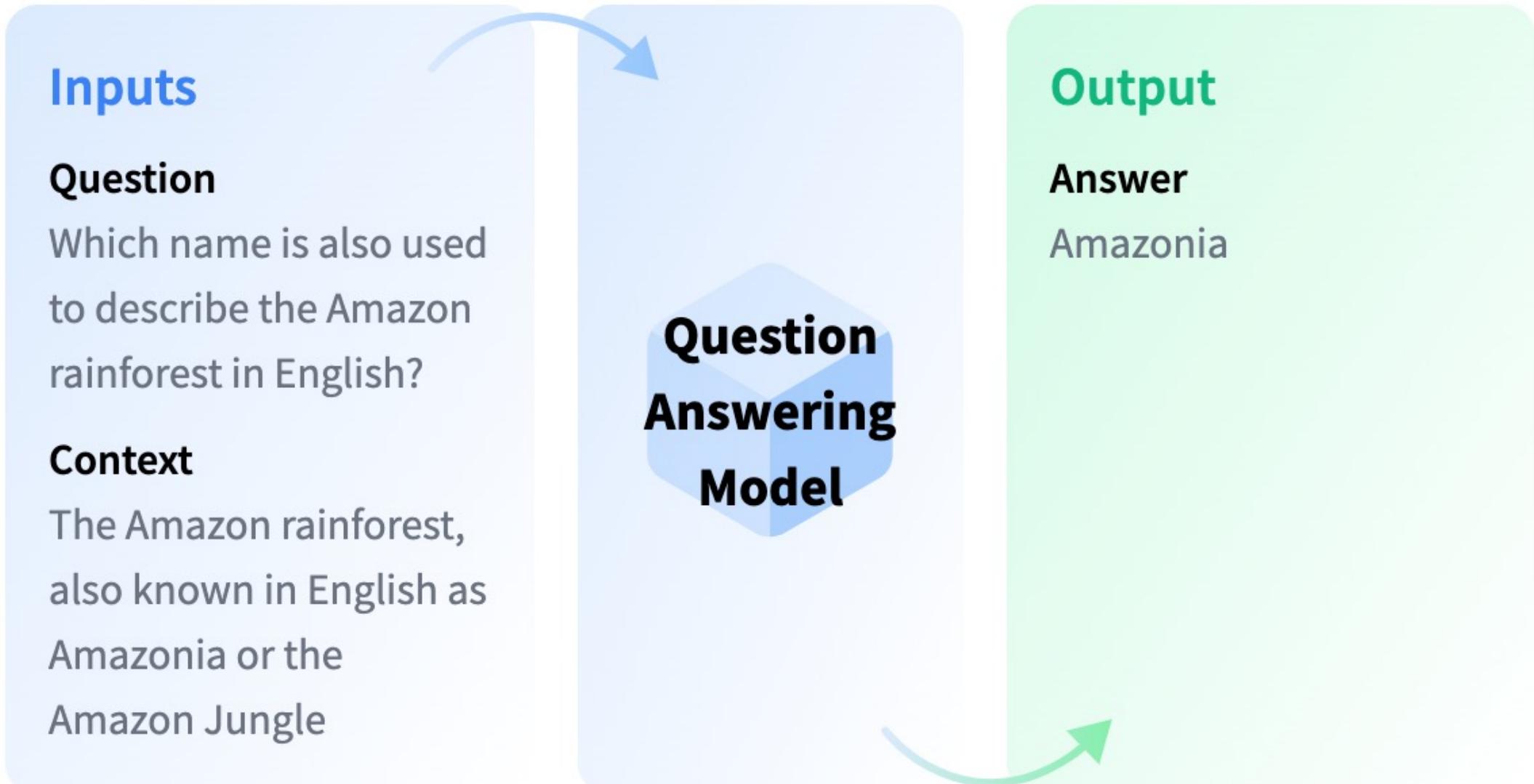
Task-oriented DS (TOD), Open-domain DS (ODD)

Dialogue System

Different types of dialogues in Task-oriented DS (TOD) and Open-domain DS (ODD)

Types of DS	Types of Dialogues	User message (U)	System response (S)	External knowledge (K)
Task-oriented DS (TOD)	Task-oriented	I need to find a nice restaurant in Madrid that serves expensive Thai food	There is a restaurant called Bangkok City located at 9 Red Ave.	Restaurant database
Open-domain DS (ODD)	Social Chit-Chat KG Chit-Chat	How are you going ? I like watching action movies	I am good. And you? Here are some action movies from various eras and styles that you might enjoy: <i>Inception</i> , <i>The Avengers</i>	Movie KG
	Question Answering	Do you know which team wins the 2022 World Cup?	The Argentina team	Wikipedia

Question Answering



Question Answering

⚡ **Question Answering demo**
using deepset/roberta-base-squad2

Question Answering Example 2 ▾

Where do I live? Compute

Context

My name is Michael and I live in Taipei.

Computation time on cpu: 0.0492 s

Taipei 0.920

JSON Output Maximize

The screenshot shows a user interface for a Question Answering demo. At the top, it says "⚡ Question Answering demo using deepset/roberta-base-squad2". Below this, there are two tabs: "Question Answering" and "Example 2" (which is selected). A dropdown arrow is next to "Example 2". In the main area, a question "Where do I live?" is input into a text field, and a "Compute" button is to its right. Below the input field, the word "Context" is followed by a box containing the sentence "My name is Michael and I live in Taipei.". Underneath the context box, it says "Computation time on cpu: 0.0492 s". At the bottom, the answer "Taipei" is displayed with a confidence score of "0.920" in a green box. At the very bottom, there are links for "JSON Output" and "Maximize".

<https://huggingface.co/tasks/question-answering>

Question Answering

```
!pip install transformers
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)
```

```
{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

Question Answering

```
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
output = qamodel(question = question, context = context)
print(output['answer'])
```

Taipei

Question Answering

Question Answering

When did Marie Curie win her first Nobel Prize?
1903

Google search results for "when did marie curie win her first nobel prize?"

Search bar: when did marie curie win her first nobel prize?

Navigation: All, Images, News, Videos, Maps, More, Settings, Tools

About 319'000 results (1.41 seconds)

Images displayed:

- A woman in a lab coat working at a bench.
- Two men and a woman in a laboratory setting.
- Portrait of a young Marie Curie.
- Portrait of a younger Marie Curie.
- A woman sitting on a chair next to a child.
- A close-up portrait of an older Marie Curie.

1903

With Henri Becquerel and **her** husband, Pierre **Curie**, **Marie Curie was** awarded the 1903 **Nobel Prize** for Physics. She **was** the sole winner of the 1911 **Nobel Prize** for Chemistry. She **was** the **first** woman to **win** a **Nobel Prize** and the only woman to **win** the award in two different fields.

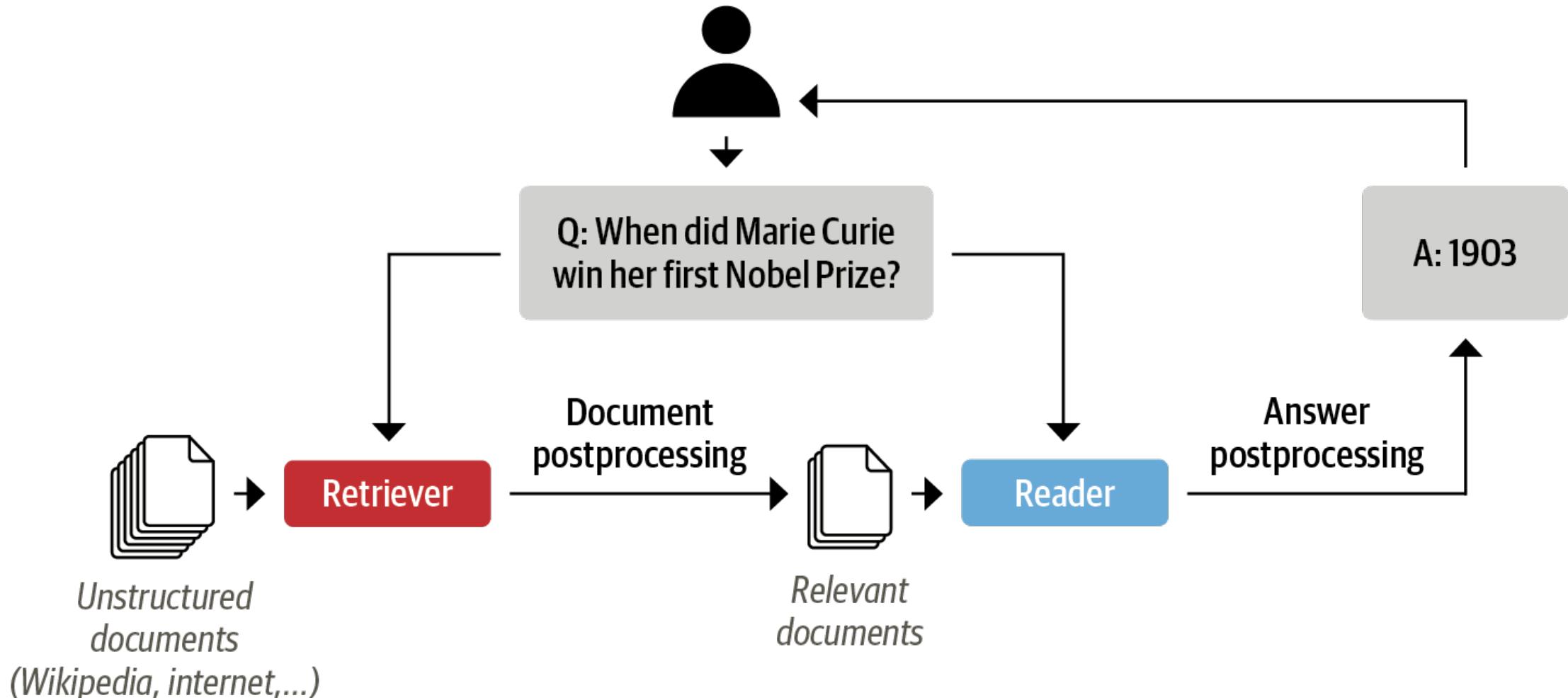
<https://www.britannica.com> › Science › Physics › Physicists
[Marie Curie | Biography & Facts | Britannica](#)

About featured snippets • Feedback

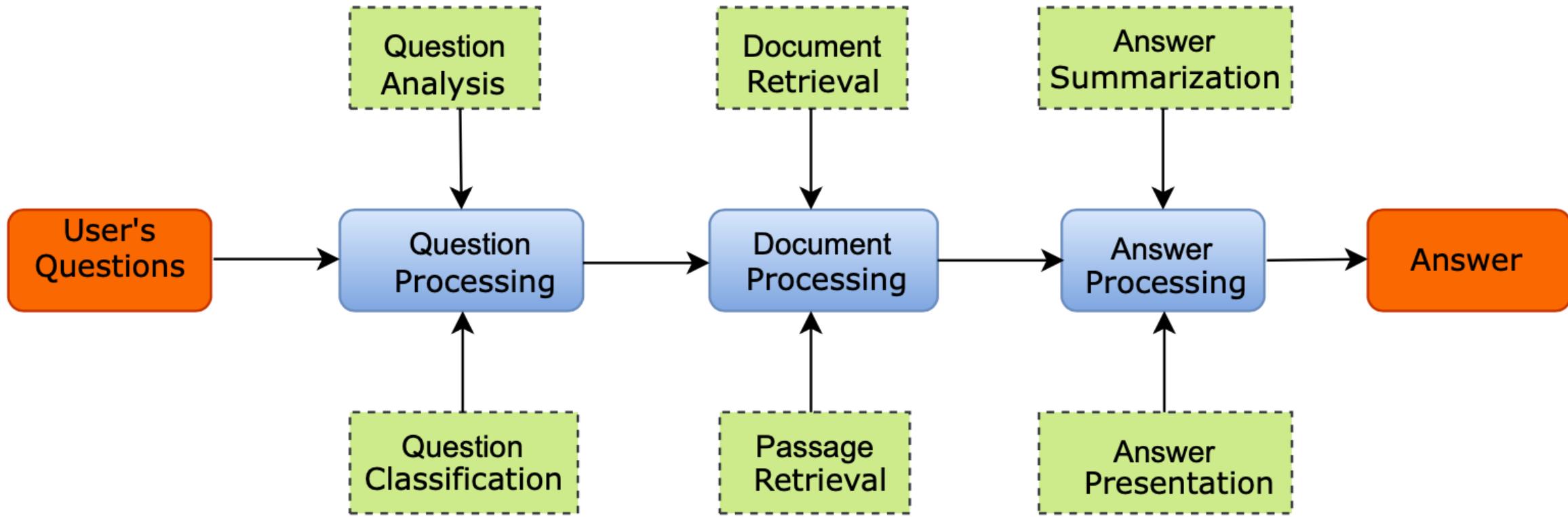
The Retriever-Reader Architecture for Modern QA Systems

When did Marie Curie win her first Nobel Prize?

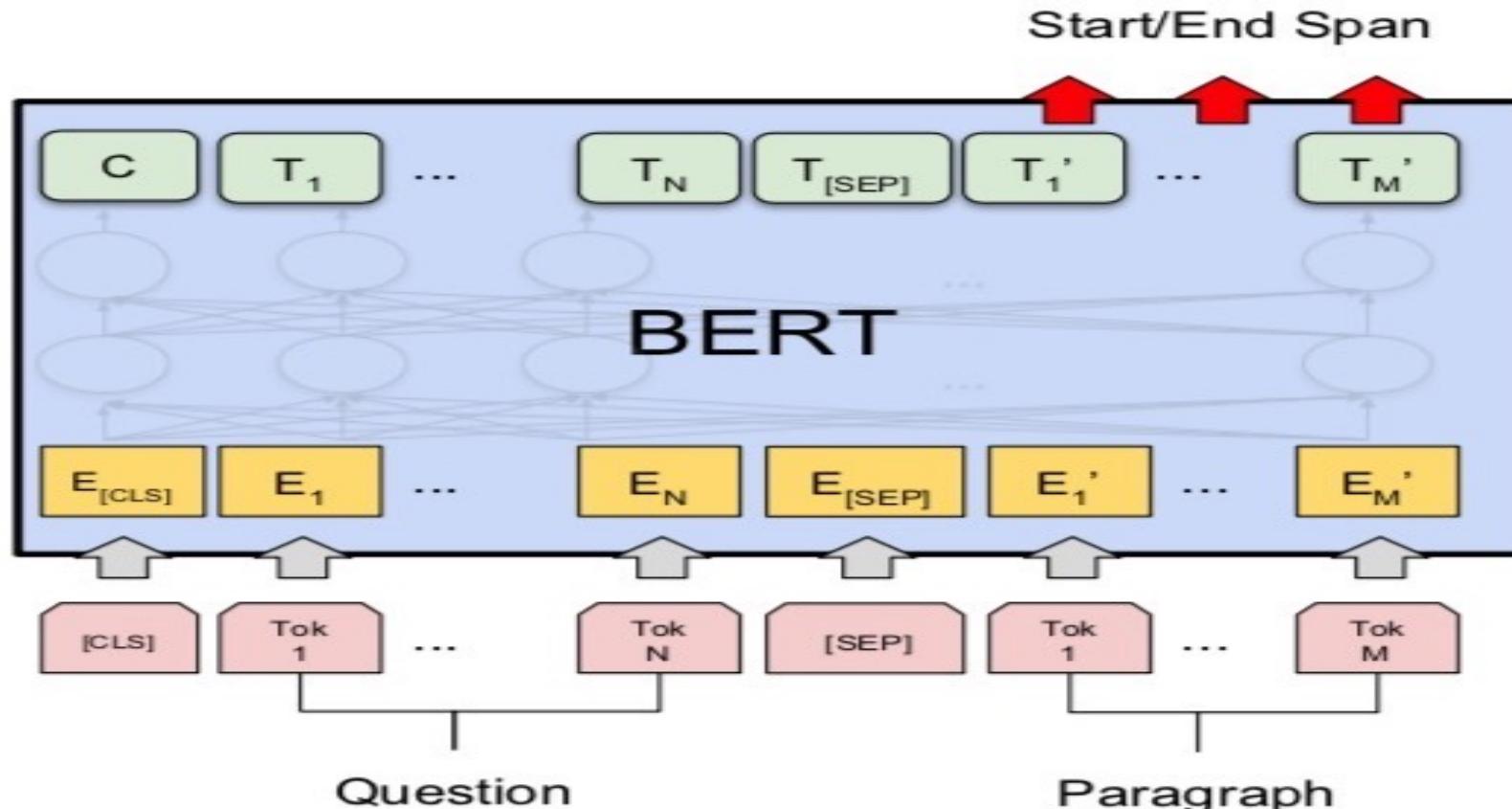
1903



Question Answering System (QAS)



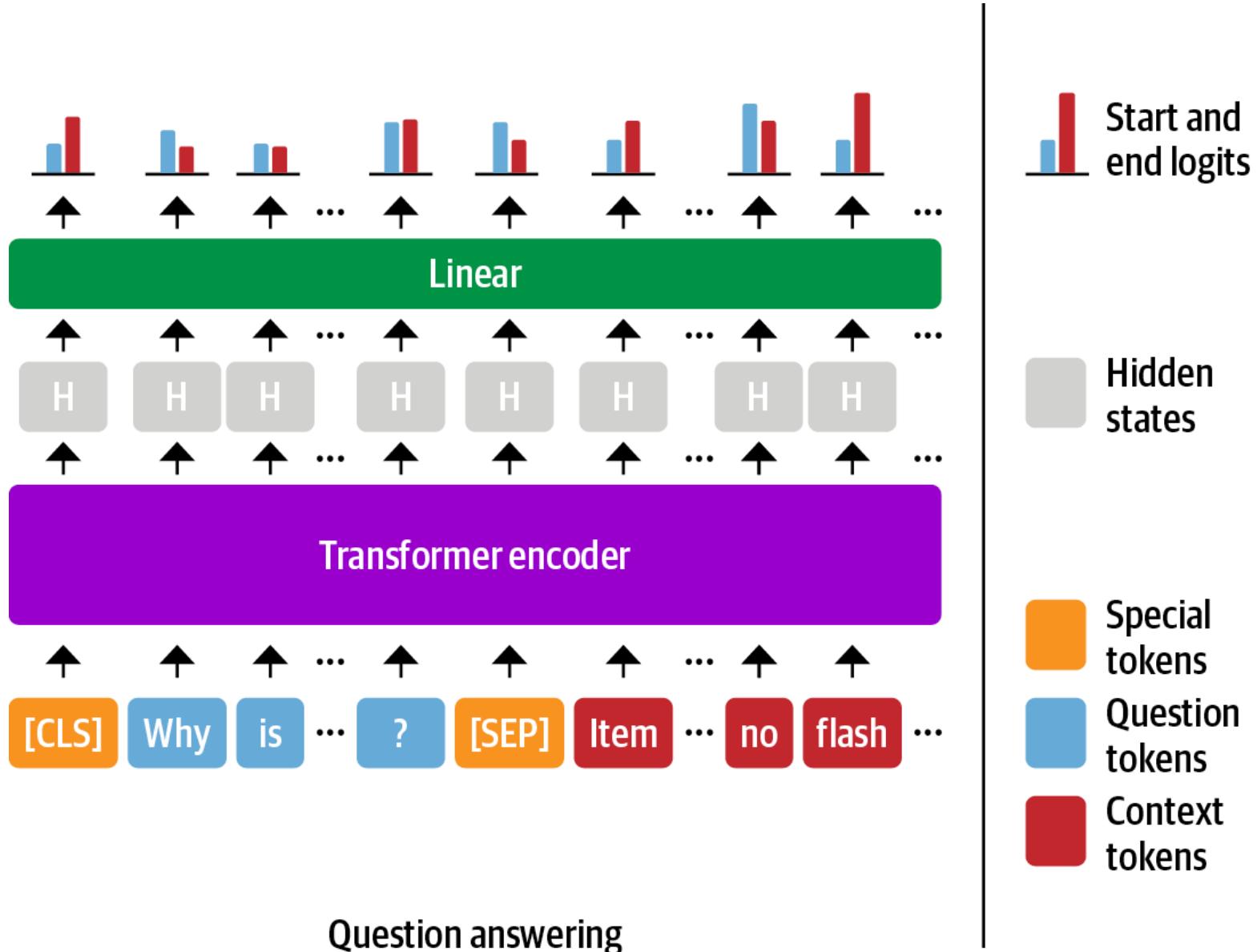
Fine-tuning BERT on Question Answering (QA)



(c) Question Answering Tasks:
SQuAD v1.1

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

The span classification head for QA tasks



Multiple question-context pairs

Why is the camera of poor quality? Item like the picture, fast deliver 3 days well packed, good quality for the price. The camera is decent (as phone cameras go). There is no flash though...



[CLS] [blue bar] [SEP] [red bar] [SEP]

Stride

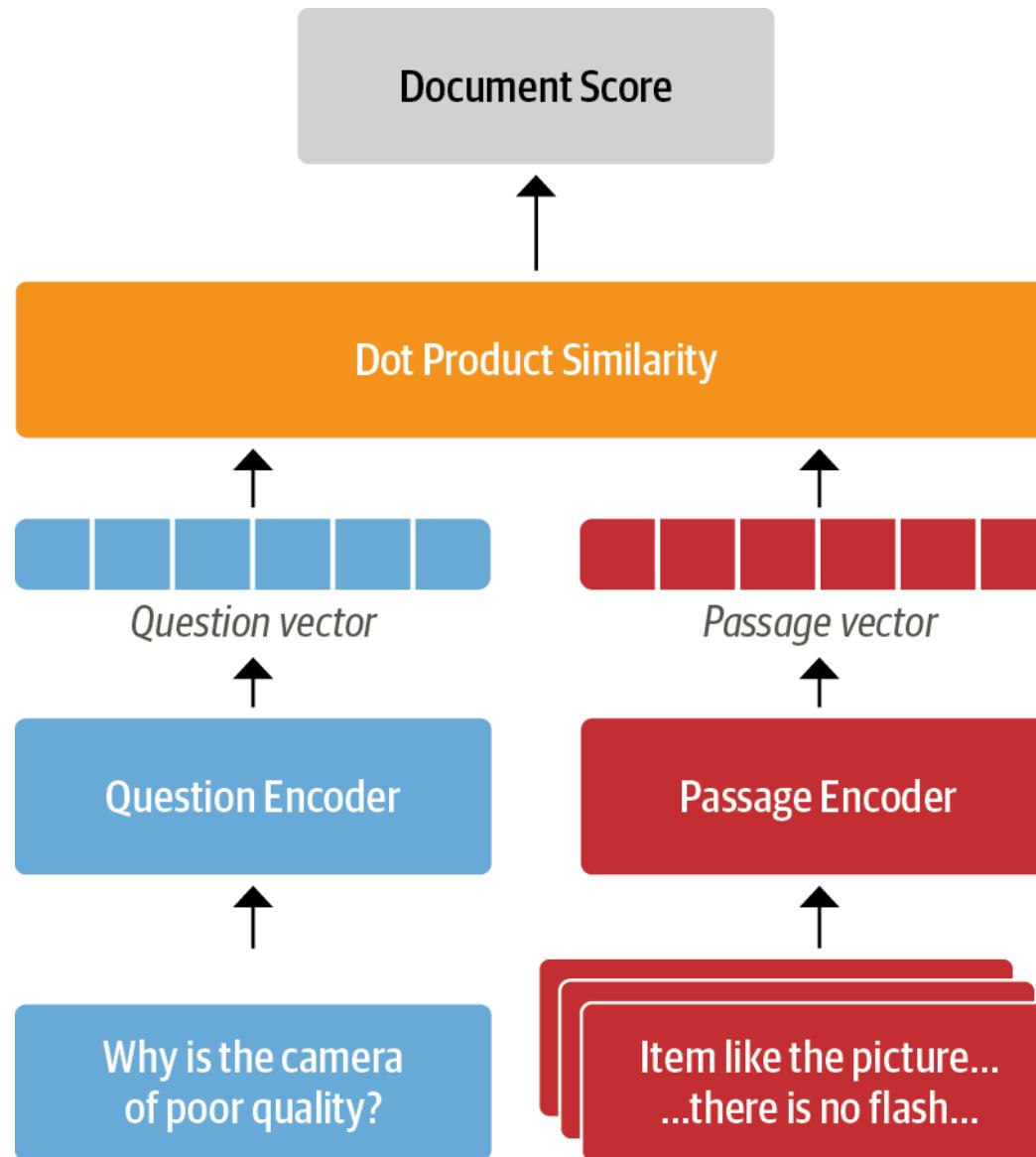


Why is the camera of poor quality? Item like the picture, fast deliver 3 days well packed, good quality for the price. The camera is decent (as phone cameras go). There is no flash though...



[CLS] [blue bar] [SEP] [red bar] [SEP]

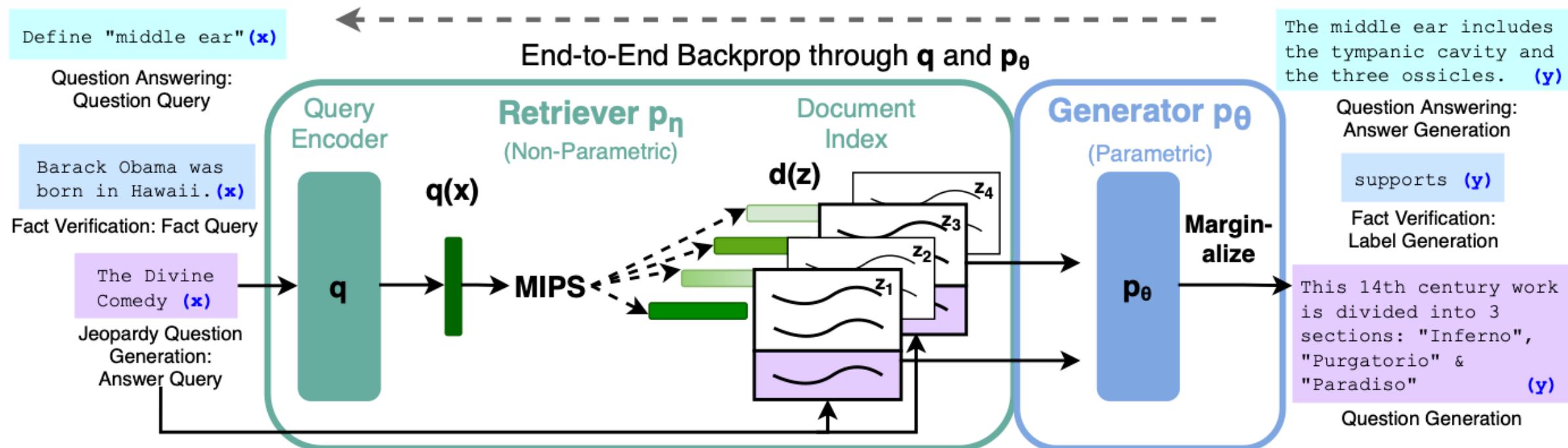
Dense Passage Retrieval (DPR)



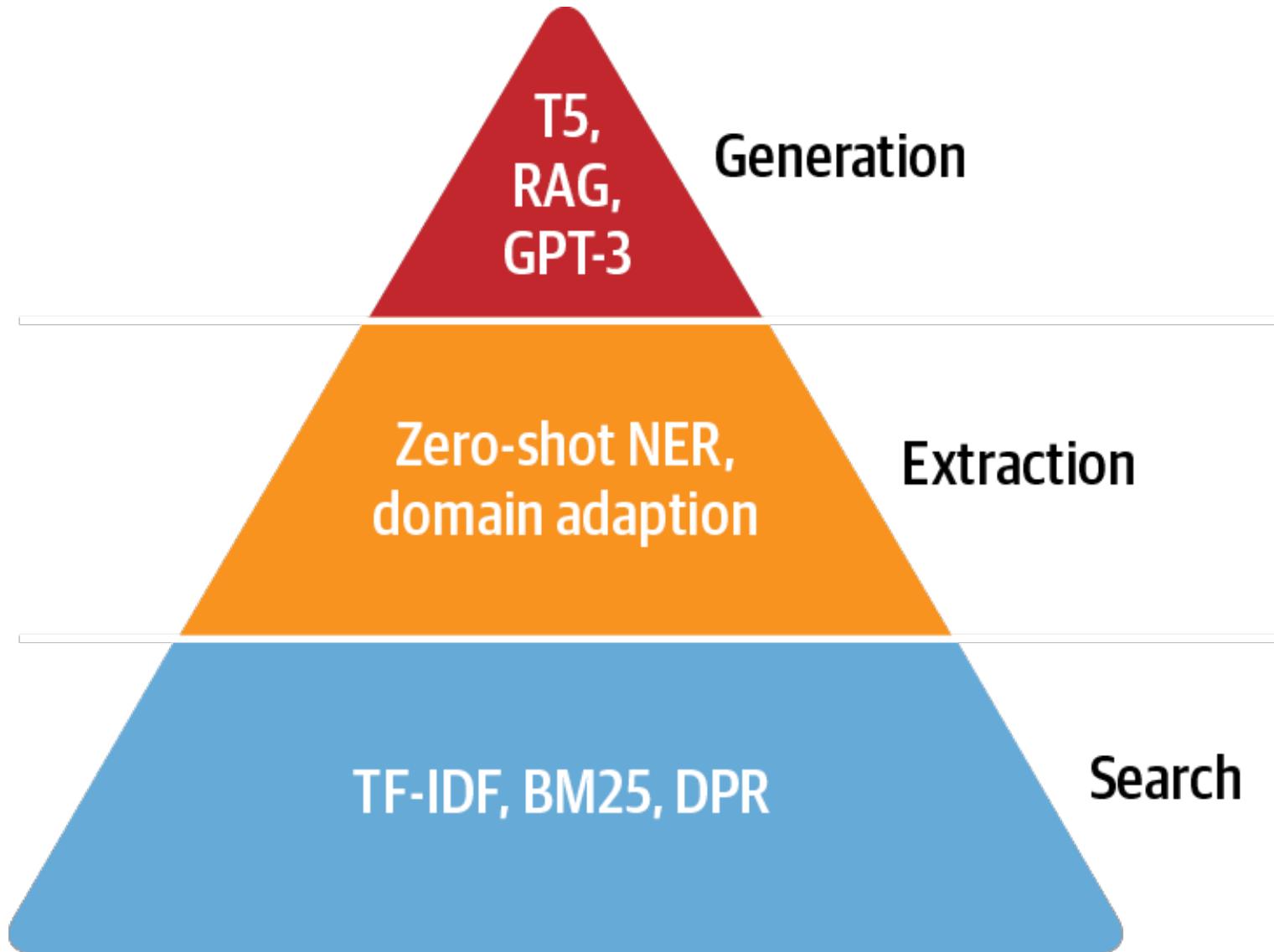
Going Beyond Extractive QA

Retrieval-Augmented Generation (RAG)

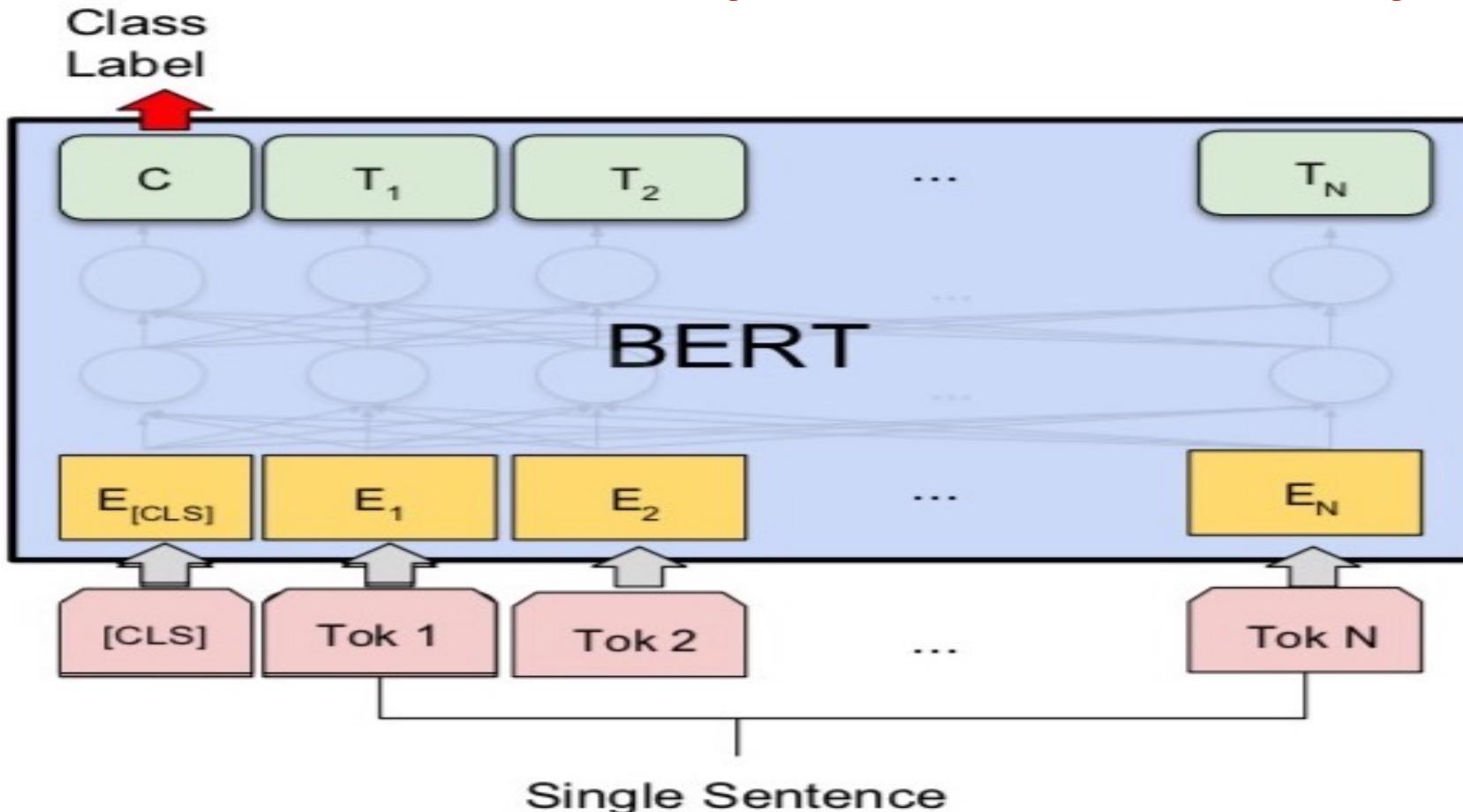
The RAG architecture for fine-tuning a retriever and generator end-to-end



The QA Hierarchy of Needs



Fine-tuning BERT on Dialogue Intent Detection (ID; Classification)

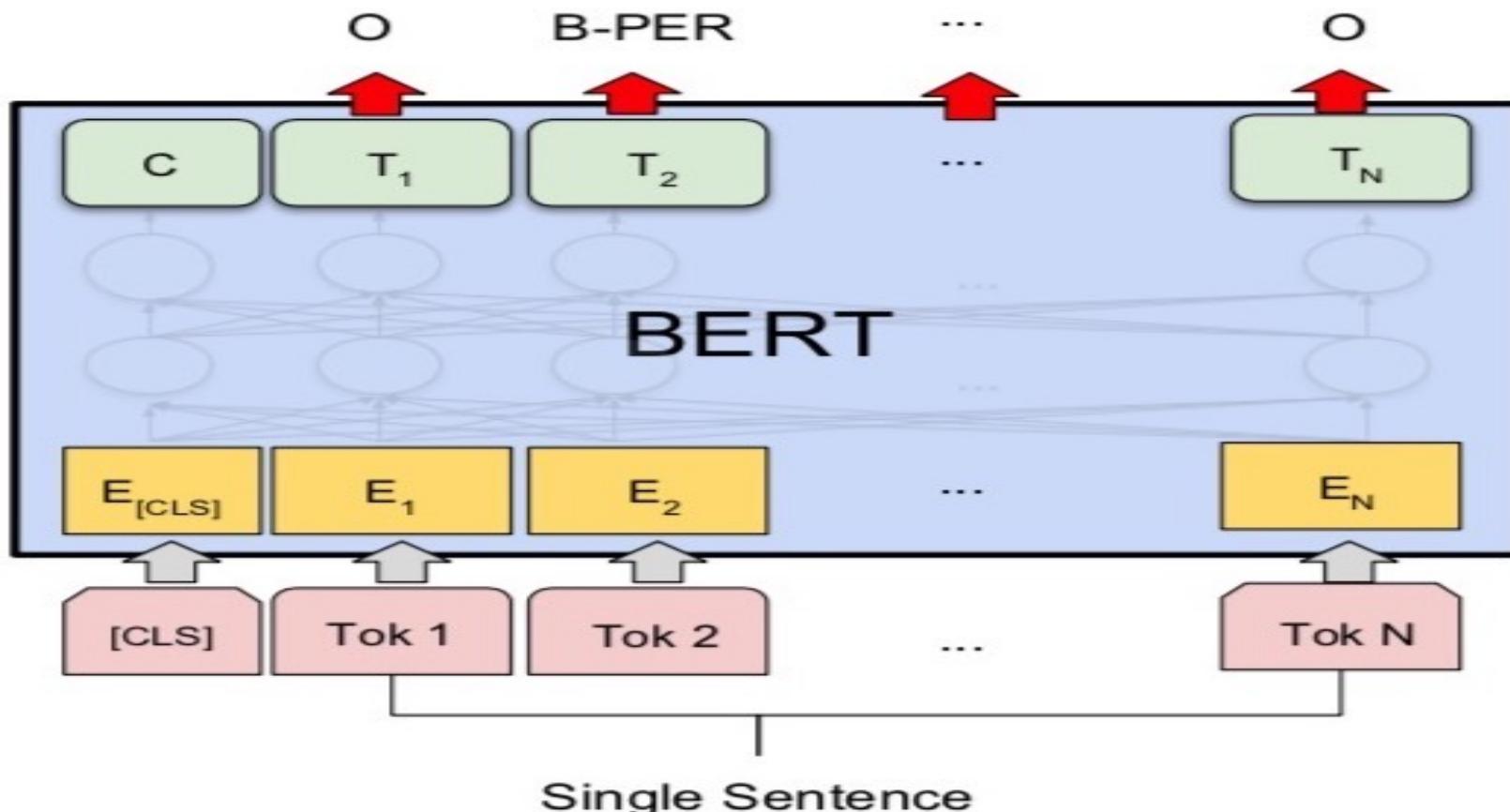


(b) Single Sentence Classification Tasks:
SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on Dialogue Slot Filling (SF)



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Question Answering

(QA)

SQuAD

Stanford Question Answering Dataset

SQuAD

SQuAD

Home

Explore 2.0

Explore 1.1

SQuAD2.0

The Stanford Question Answering Dataset

What is SQuAD?

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or *span*, from the corresponding reading passage, or the question might be unanswerable.

SQuAD2.0 combines the 100,000 questions in SQuAD1.1 with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. To do well on SQuAD2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

Rank	Model	EM	F1
	Human Performance Stanford University (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) QIANXIN	90.724	93.011
2 May 05, 2020	SA-Net-V2 (ensemble) QIANXIN	90.679	92.948
2	Retro-Reader (ensemble)	90.578	92.978

<https://rajpurkar.github.io/SQuAD-explorer/>

SQuAD

SQuAD: 100,000+ Questions for Machine Comprehension of Text

Pranav Rajpurkar and Jian Zhang and Konstantin Lopyrev and Percy Liang

{pranavsr,zjian,klopyrev,pliang}@cs.stanford.edu

Computer Science Department
Stanford University

Abstract

We present the Stanford Question Answering Dataset (SQuAD), a new reading comprehension dataset consisting of 100,000+ questions posed by crowdworkers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage. We analyze the dataset to understand the types of reasoning required to answer the questions, leaning heavily on dependency and constituency trees. We build a strong logistic regression model, which achieves an F1 score of 51.0%, a significant improvement over a simple baseline (20%). However, human performance (86.8%) is much higher, indicating that the dataset presents a good challenge problem for future research. The dataset is freely available at <https://stanford-qa.com>.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure 1: Question-answer pairs for a sample passage in the

Source: Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang.

"Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).

SQuAD (Question Answering)

Q: What causes precipitation to fall?

Precipitation

From Wikipedia, the free encyclopedia

For other uses, see [Precipitation \(disambiguation\)](#).

In meteorology, **precipitation** is any product of the condensation of atmospheric water vapor that falls under gravity from clouds.^[2] The main forms of precipitation include drizzle, rain, sleet, snow, ice pellets, graupel and hail. Precipitation occurs when a portion of the atmosphere becomes saturated with water vapor (reaching 100% [relative humidity](#)), so that the water condenses and "precipitates". Thus, fog and mist are not precipitation but suspensions, because the water vapor does not condense sufficiently to precipitate. Two processes, possibly acting together, can lead to air becoming saturated: cooling the air or adding water vapor to the air. Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers."^[3]

SQuAD (Question Answering)

Paragraph

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?

A: gravity

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

A: graupel

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: Where do water droplets collide with ice crystals to form precipitation?

A: within a cloud

SQuAD (Question Answering)

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

Q: What causes precipitation to fall?

A: **gravity**

Q: What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

A: graupel

Q: Where do water droplets collide with ice crystals to form precipitation?

A: within a cloud

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows the Google Colab interface with three code cells. The first cell contains code to answer a question about precipitation. The second cell contains code to answer a question about another main form of precipitation. The third cell contains code to answer a question about where water droplets collide with ice crystals to form precipitation. The output for each cell is shown below the code.

```
[12]: 1 from transformers import pipeline
2 qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "What causes precipitation to fall?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

gravity

```
[13]: 1 from transformers import pipeline
2 qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

graupel

```
1 #from transformers import pipeline
2 #qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "Where do water droplets collide with ice crystals to form precipitation?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

within a cloud

<https://tinyurl.com/aintpuppython101>

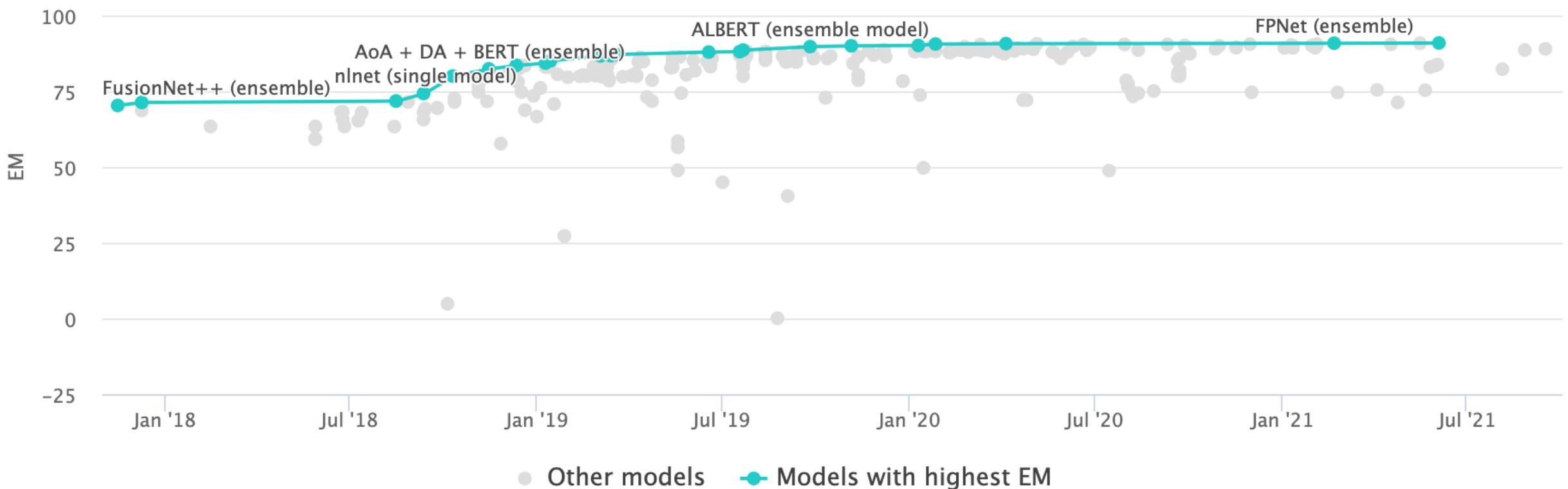
Question Answering

```
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "What causes precipitation to fall?"
context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers"."""
output = qamodel(question = question, context = context)
print(output['answer'])
```

gravity

Question Answering on SQuAD2.0

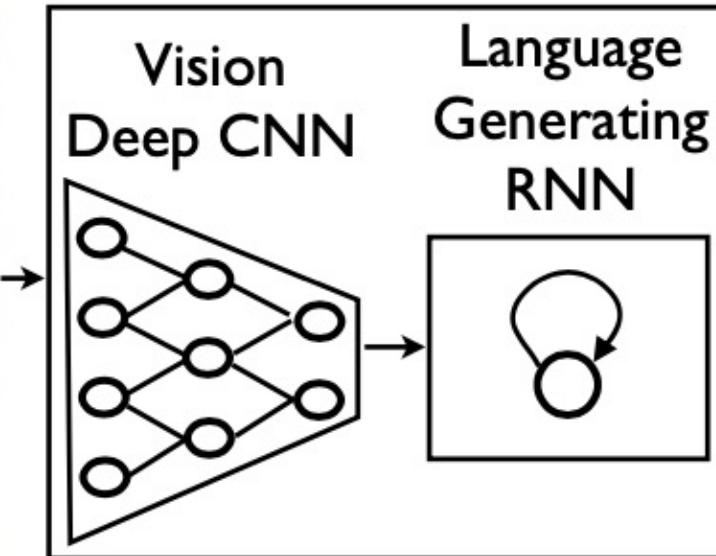
SQuAD 2.0 benchmark (Papers with Code)



<https://paperswithcode.com/sota/question-answering-on-squad20>

Neural Image Captioning (NIC)

image-to-text description generation

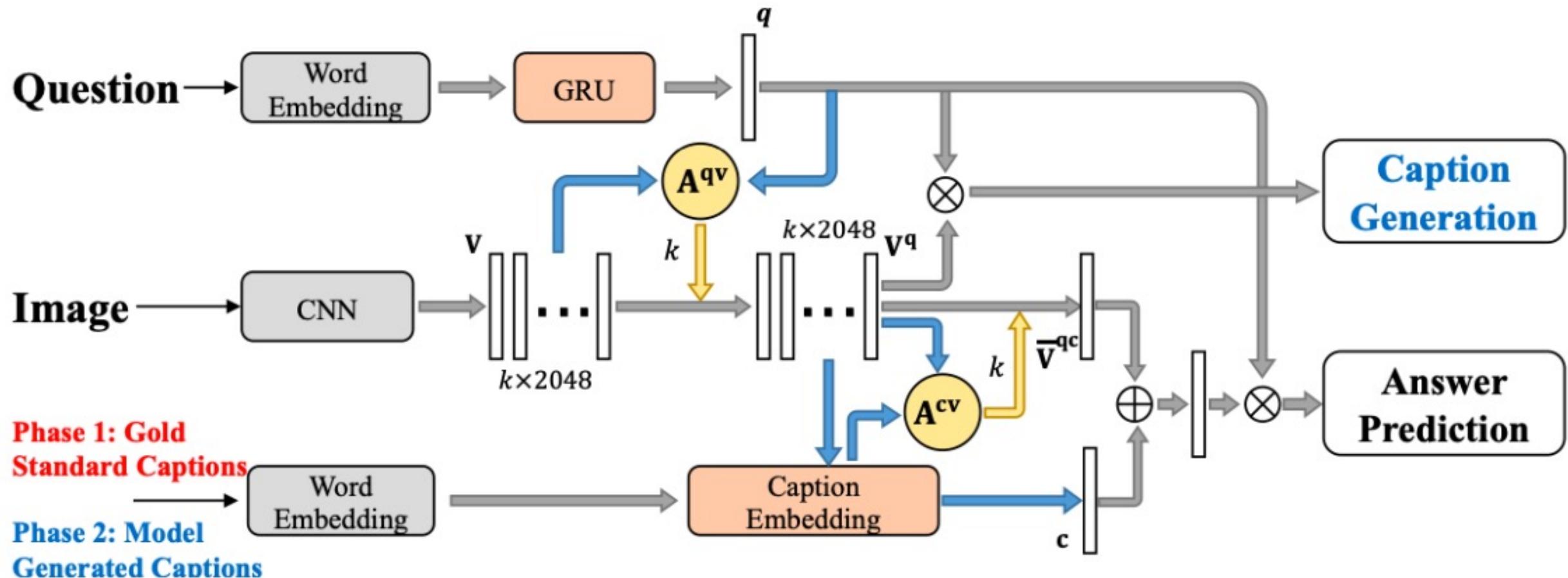


A group of people shopping at an outdoor market.

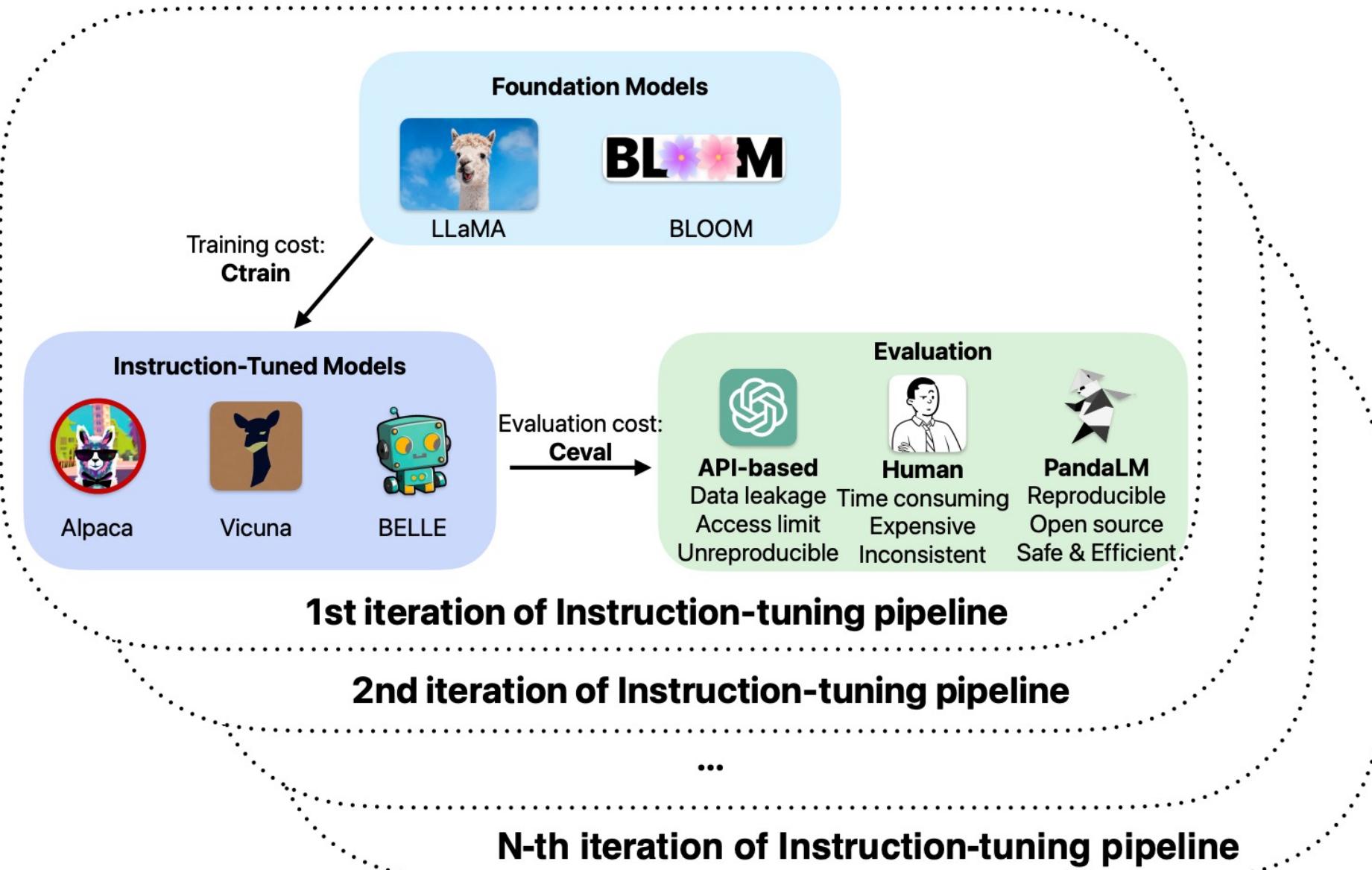
There are many vegetables at the fruit stand.

Visual Question Answering

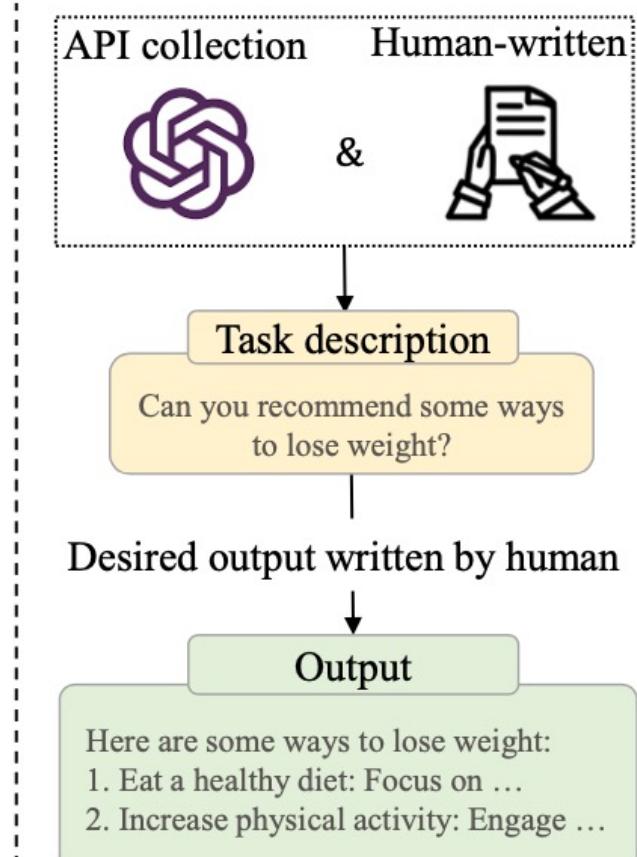
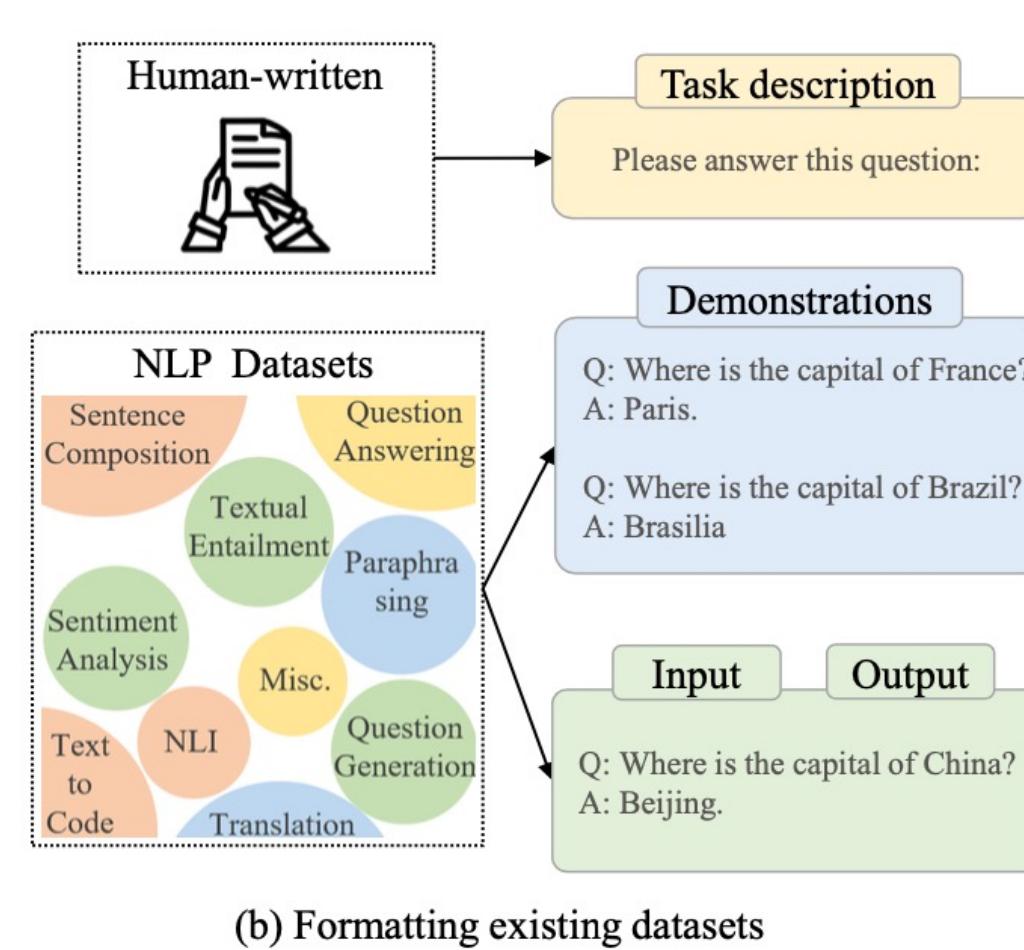
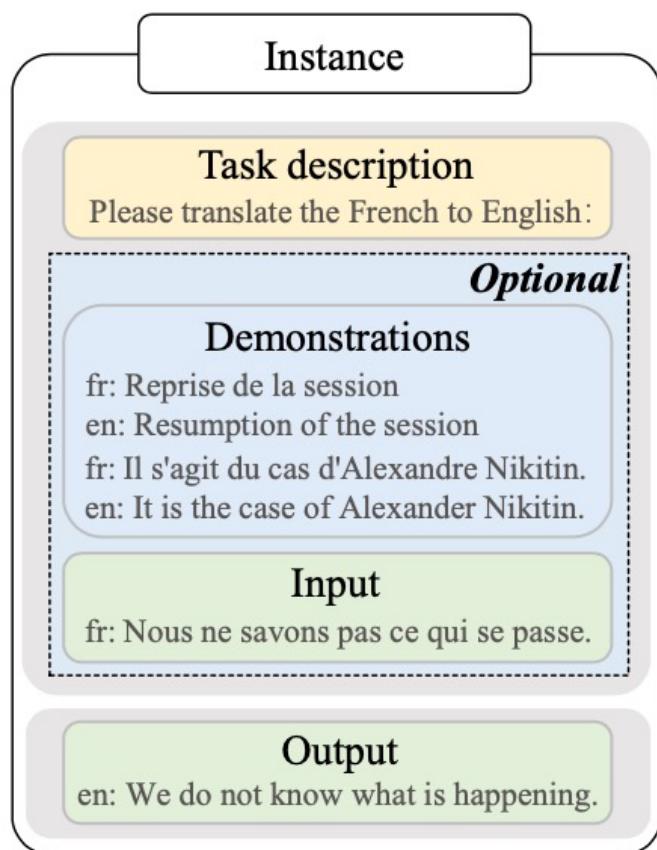
Neural caption generation is employed to aid answer prediction



Pipeline of Instruction Tuning LLMs



Instance Formatting and Two Different Methods for Constructing the Instruction-formatted Instances



(a) Instance format

(b) Formatting existing datasets

(c) Formatting human needs

In-context Learning (ICL) and Chain-of-thought (CoT) Prompting

In-Context Learning

Answer the following mathematical reasoning questions:

Q: If you have 12 candies and you give 4 candies to your friend, how many candies do you have left?

A: The answer is 8.

Q: If a rectangle has a length of 6 cm and a width of 3 cm, what is the perimeter of the rectangle?

A: The answer is 18 cm.

Q: Sam has 12 marbles. He gives 1/4 of them to his sister. How many marbles does Sam have left?

A: The answer is 9.

Chain-of-Thought Prompting

Answer the following mathematical reasoning questions:

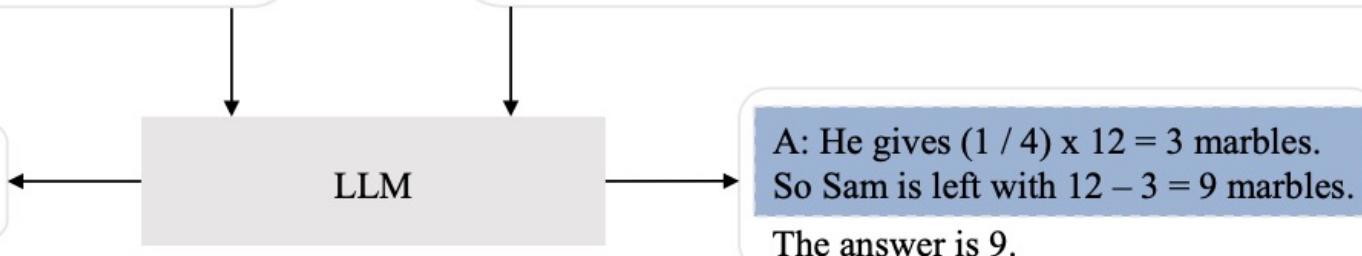
Q: If a rectangle has a length of 6 cm and a width of 3 cm, what is the perimeter of the rectangle?

A: For a rectangle, add up the length and width and double it. So, the perimeter of this rectangle is $(6 + 3) \times 2 = 18$ cm.

The answer is 18 cm.

Q: Sam has 12 marbles. He gives 1/4 of them to his sister. How many marbles does Sam have left?

A: He gives $(1 / 4) \times 12 = 3$ marbles. So Sam is left with $12 - 3 = 9$ marbles.
The answer is 9.



: Task description

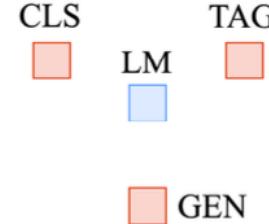
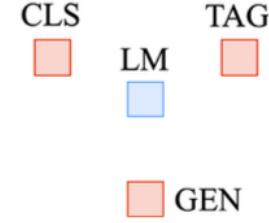
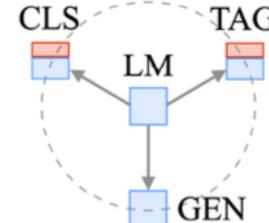
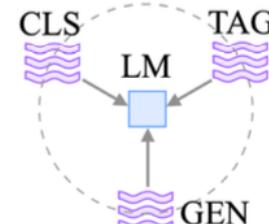
: Demonstration

: Chain-of-Thought

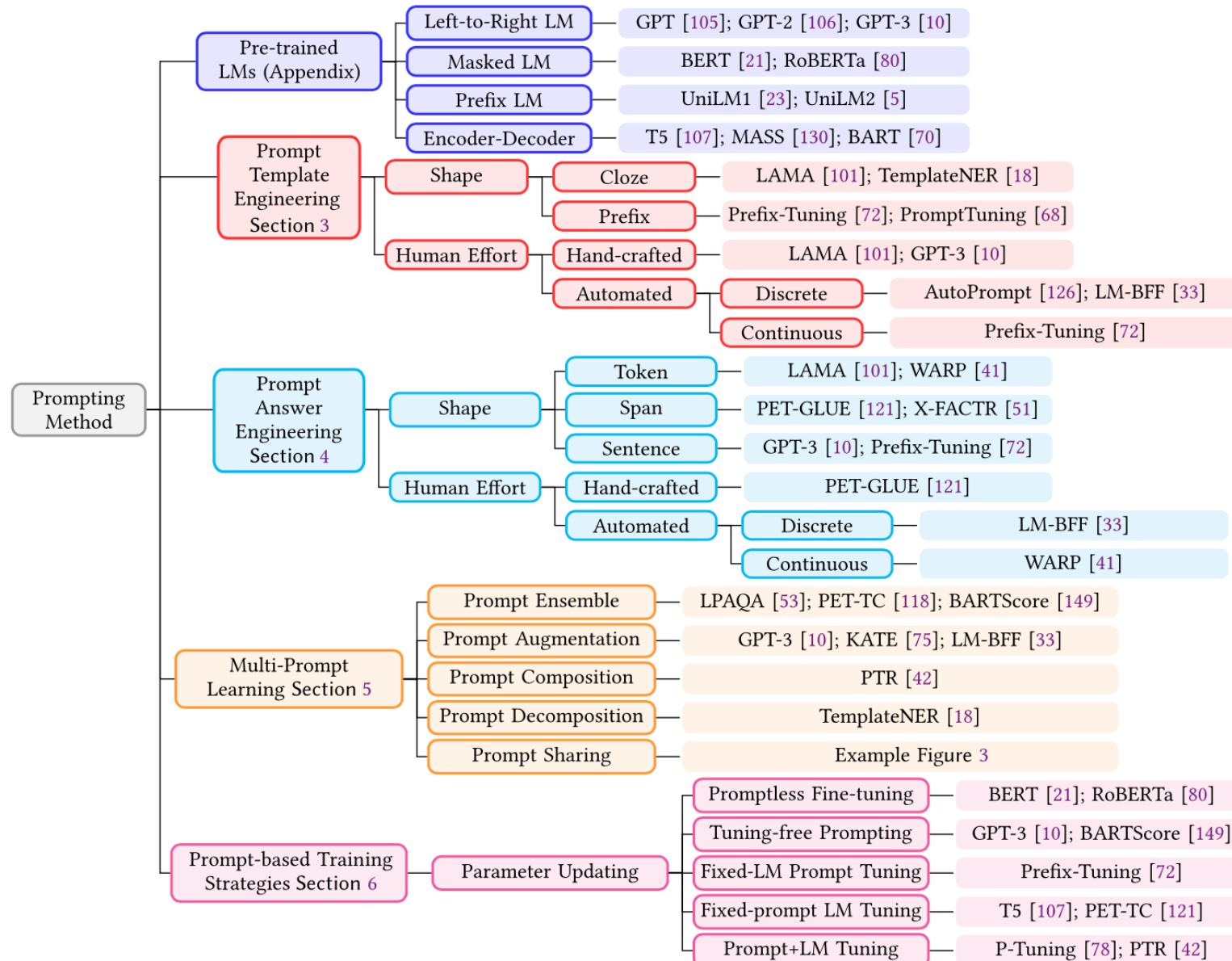
: Query

Pre-train, Prompt, and Predict: Prompting Methods in Natural Language Processing (LLMs)

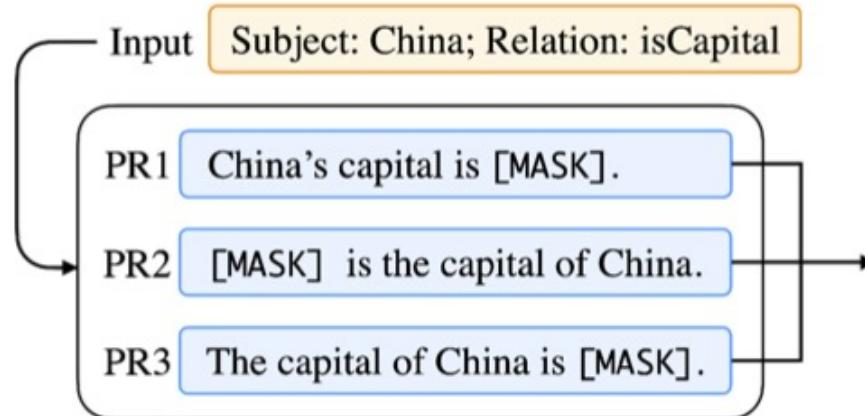
Four Paradigms in NLP

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Feature (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

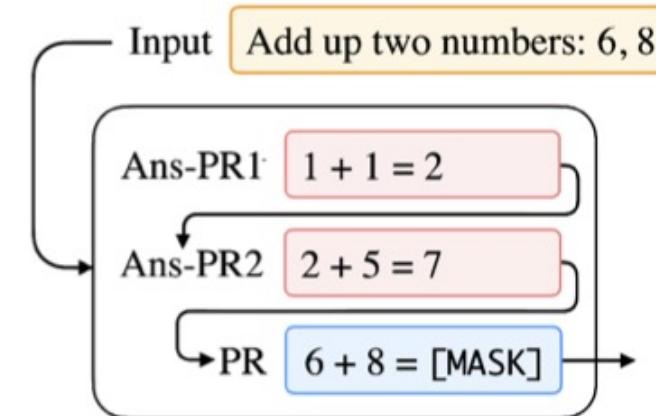
Typology of Prompting Methods



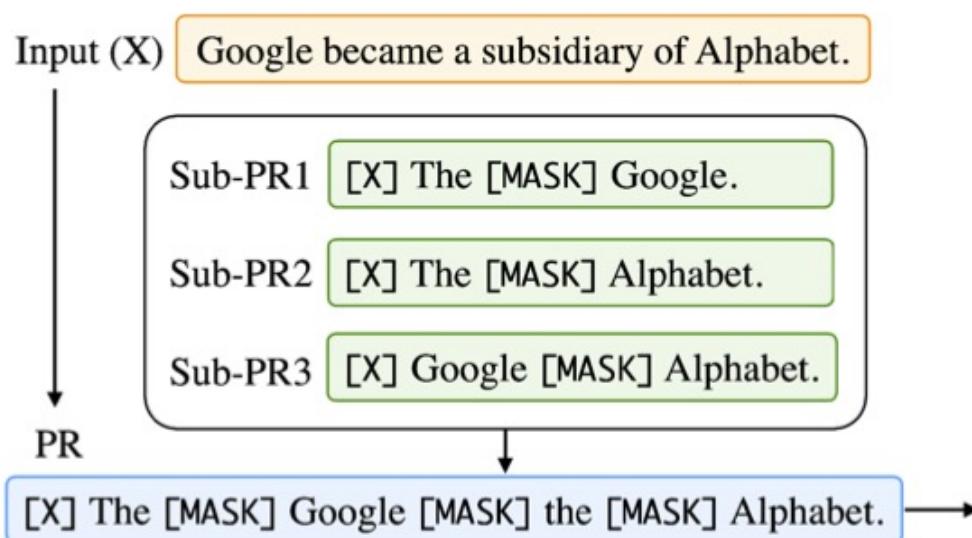
Different Multi-Prompt Learning Strategies



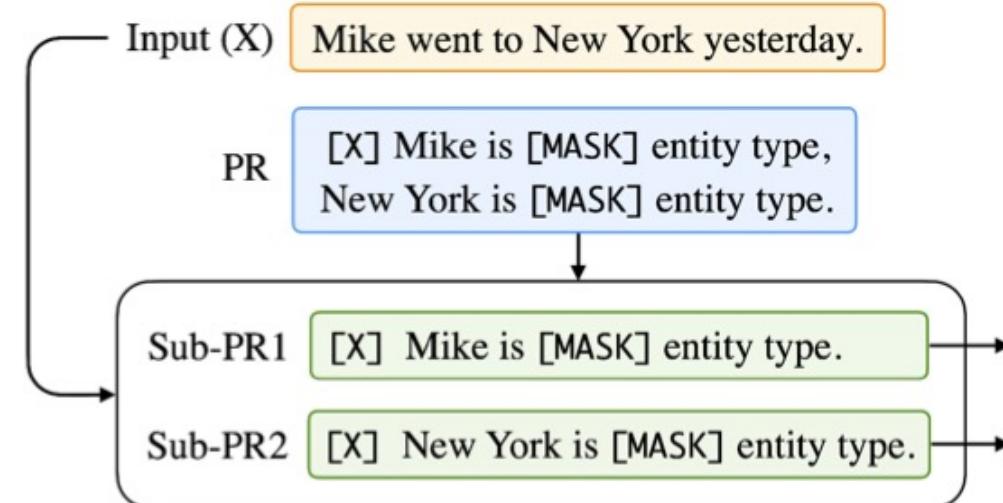
(a) Prompt Ensembling.



(b) Prompt Augmentation.



(c) Prompt Composition.



(d) Prompt Decomposition.

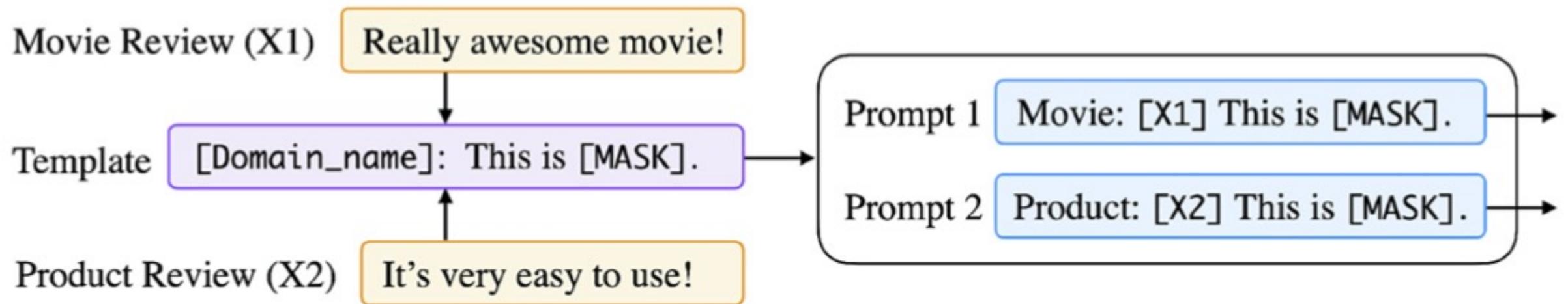
Examples of Input, Template, and Answer for Different Tasks

Type	Task Example	Input ([X])	Template	Answer ([Z])
Text Classification	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span Classification	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
				Yes No ...
				...
Text-pair Classification	Natural Language Inference	[X1]: An old man with ...	[X1]? [Z], [X2]	Yes No ...
		[X2]: A man walks
		[X1]: Mike went to Paris.		organization location ...
Tagging	Named Entity Recognition	[X2]: Paris	[X1][X2] is a [Z] entity.	...
		[X1]: Mike went to Paris.		The victim ... A woman
		[X2]: Paris		...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman
				...
				I love you. I fancy you. ...
Regression	Textual Similarity	[X1]: A man is smoking.	[X1] [Z], [X2]	Yes No ...
		[X2]: A man is skating.		...
		[X1]: A man is smoking.		...

Characteristics of Different Tuning Strategies

Strategy	LM Params	Prompt Params		Example
		Additional	Tuned	
Promptless Fine-tuning	Tuned	—	—	ELMo [97], BERT [20], BART [69]
Tuning-free Prompting	Frozen	✗	✗	GPT-3 [9], AutoPrompt [125], LAMA [100]
Fixed-LM Prompt Tuning	Frozen	✓	Tuned	Prefix-Tuning [71], Prompt-Tuning [67]
Fixed-prompt LM Tuning	Tuned	✗	✗	PET-TC [117], PET-Gen [118], LM-BFF [32]
Prompt+LM Fine-tuning	Tuned	✓	Tuned	PADA [5], P-Tuning [77], PTR [41]

Multi-prompt Learning for Multi-task, Multi-domain, or Multi-lingual Learning



GPT Prompt Engineering for Question Answering

- Find the answer to the question from the given context.
- When the question cannot be answered with the given context, say "unanswerable".
- Just say the answer without repeating the question.
- Context: {context}
- Question:{question}
- Answer:

Prompts and QA Inference For FLAN T5

Question Answering

- Prompts and QA Inference For FLAN T5, we follow [41] and use the following prompt:
- Context: {context}\nQuestion: {question}\nAnswer:
- Context: {context}
- Question: {question}
- Answer:

Prompt Engineering

- **Prompts For FLAN models**
 - Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., ... & Roberts, A. (2023). The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.
- **MNLI, NLI-FEVER, VitaminC:**
 - "Premise: {premise}\n\nHypothesis: {hypothesis}\n\nDoes the premise entail the hypothesis?\n\nA yes\nB it is not possible to tell\nC no"
- **ANLI:**
 - "{context}\n\nBased on the paragraph above can we conclude that\"{hypothesis}\"?\n\nA Yes\nB It's impossible to say\nC No"
- **SNLI:**
 - "If \"{premise}\", does this mean that \"{hypothesis}\"?\n\nA yes\nB it is not possible to tell\nC no"

Dialogue Systems

Fine-tuning LLM for Dialogue System

**Reinforcement Learning from
Human Feedback (RLHF)**

ChatGPT:
Optimizing Language Models for Dialogue

Chatbot

Dialogue System

Intelligent Agent

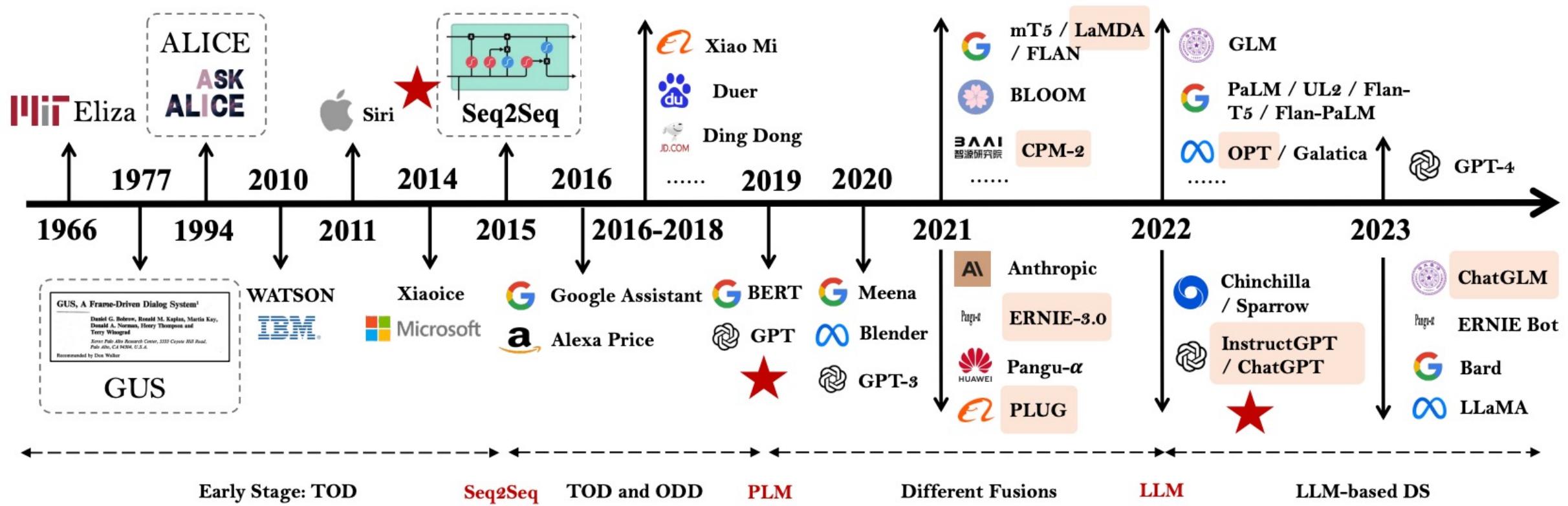
The Development of LM-based Dialogue Systems

1) Early Stage (1966 - 2015)

2) The Independent Development of TOD and ODD (2015 - 2019)

3) Fusions of Dialogue Systems (2019 - 2022)

4) LLM-based DS (2022 - Now)

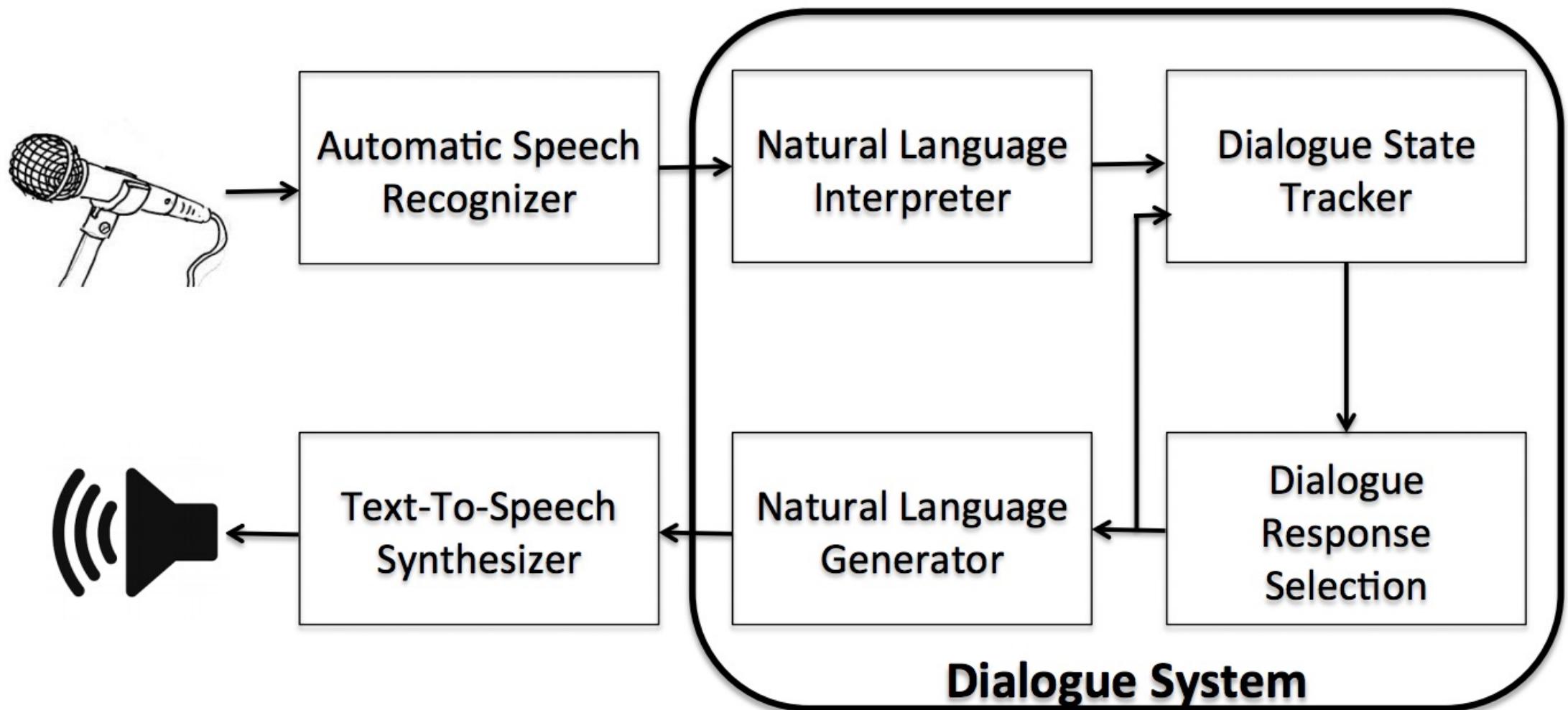


Dialogue System

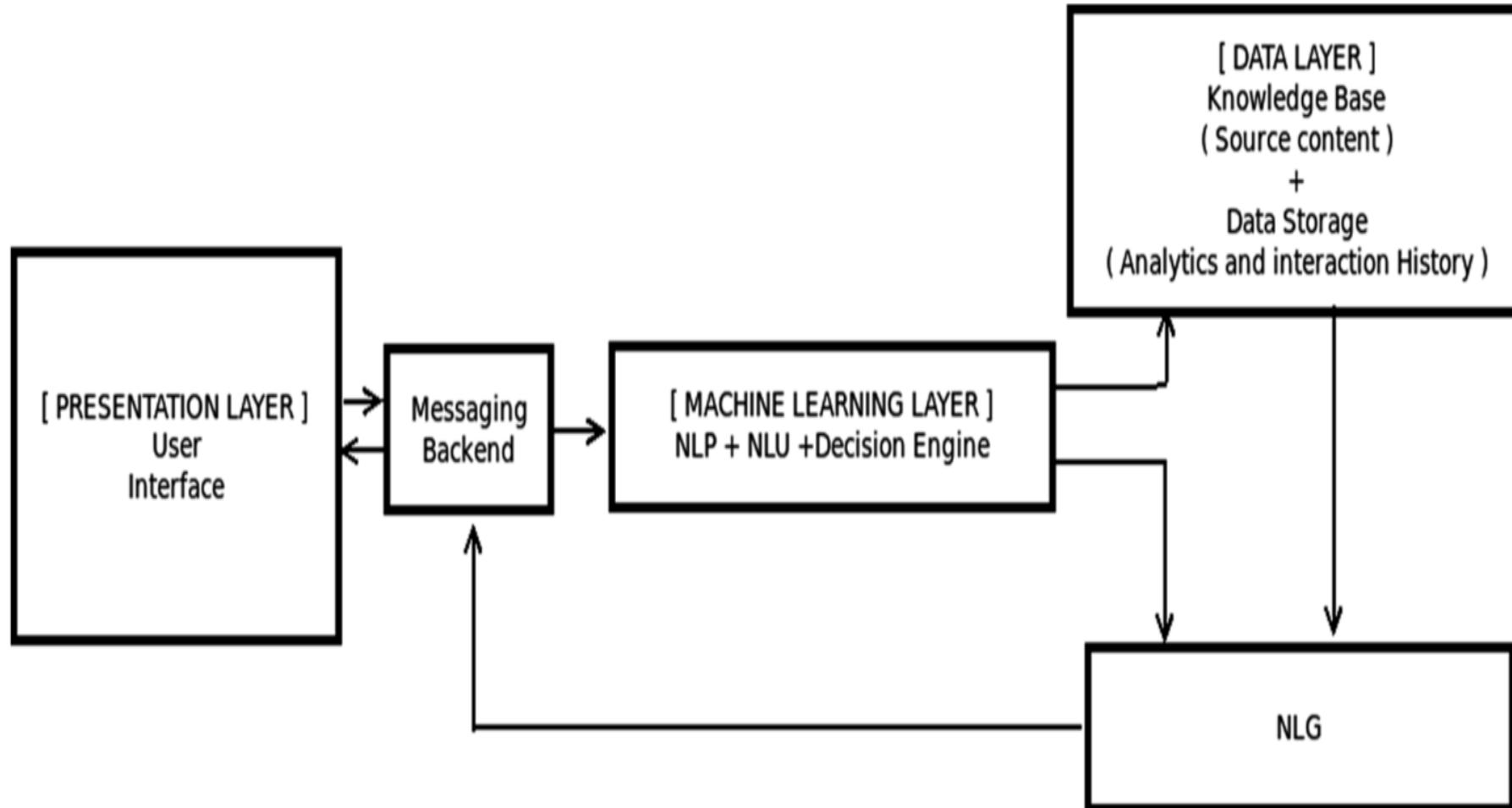
Different types of dialogues in Task-oriented DS (TOD) and Open-domain DS (ODD)

Types of DS	Types of Dialogues	User message (U)	System response (S)	External knowledge (K)
Task-oriented DS (TOD)	Task-oriented	I need to find a nice restaurant in Madrid that serves expensive Thai food	There is a restaurant called Bangkok City located at 9 Red Ave.	Restaurant database
Open-domain DS (ODD)	Social Chit-Chat KG Chit-Chat	How are you going ? I like watching action movies	I am good. And you? Here are some action movies from various eras and styles that you might enjoy: <i>Inception</i> , <i>The Avengers</i>	Movie KG
	Question Answering	Do you know which team wins the 2022 World Cup?	The Argentina team	Wikipedia

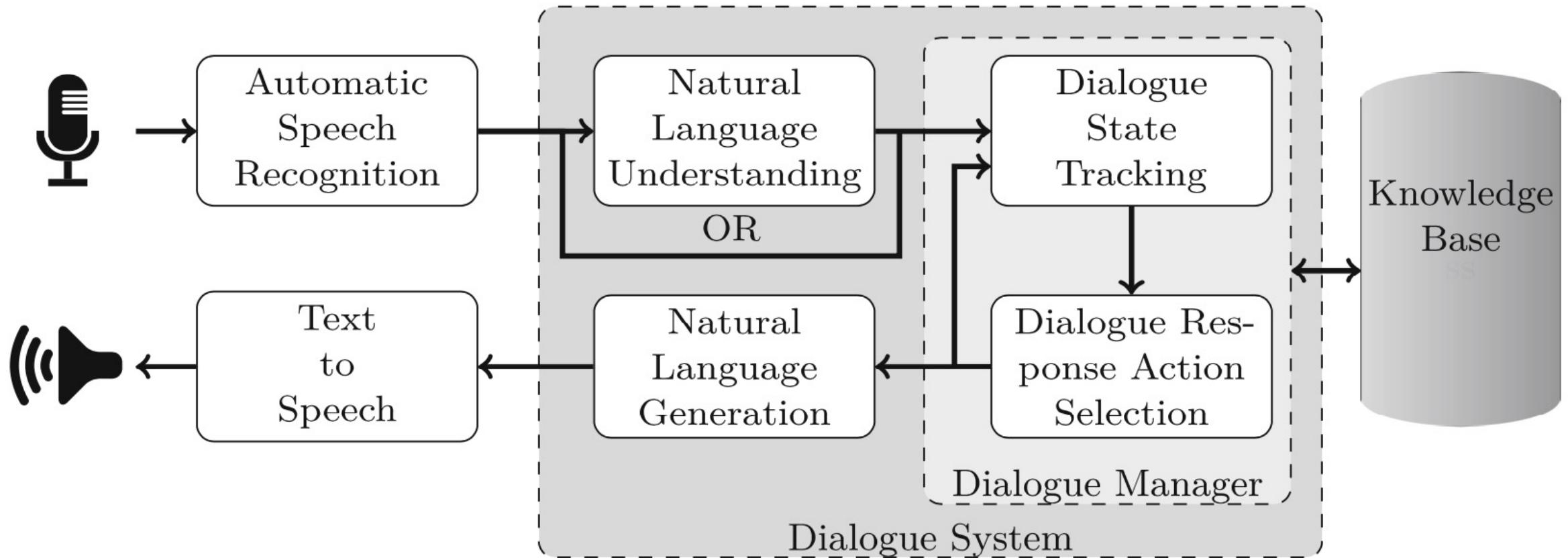
Dialogue System



Overall Architecture of Intelligent Chatbot



Modular Speech-based Dialogue System



Categories of Dialogue Systems:

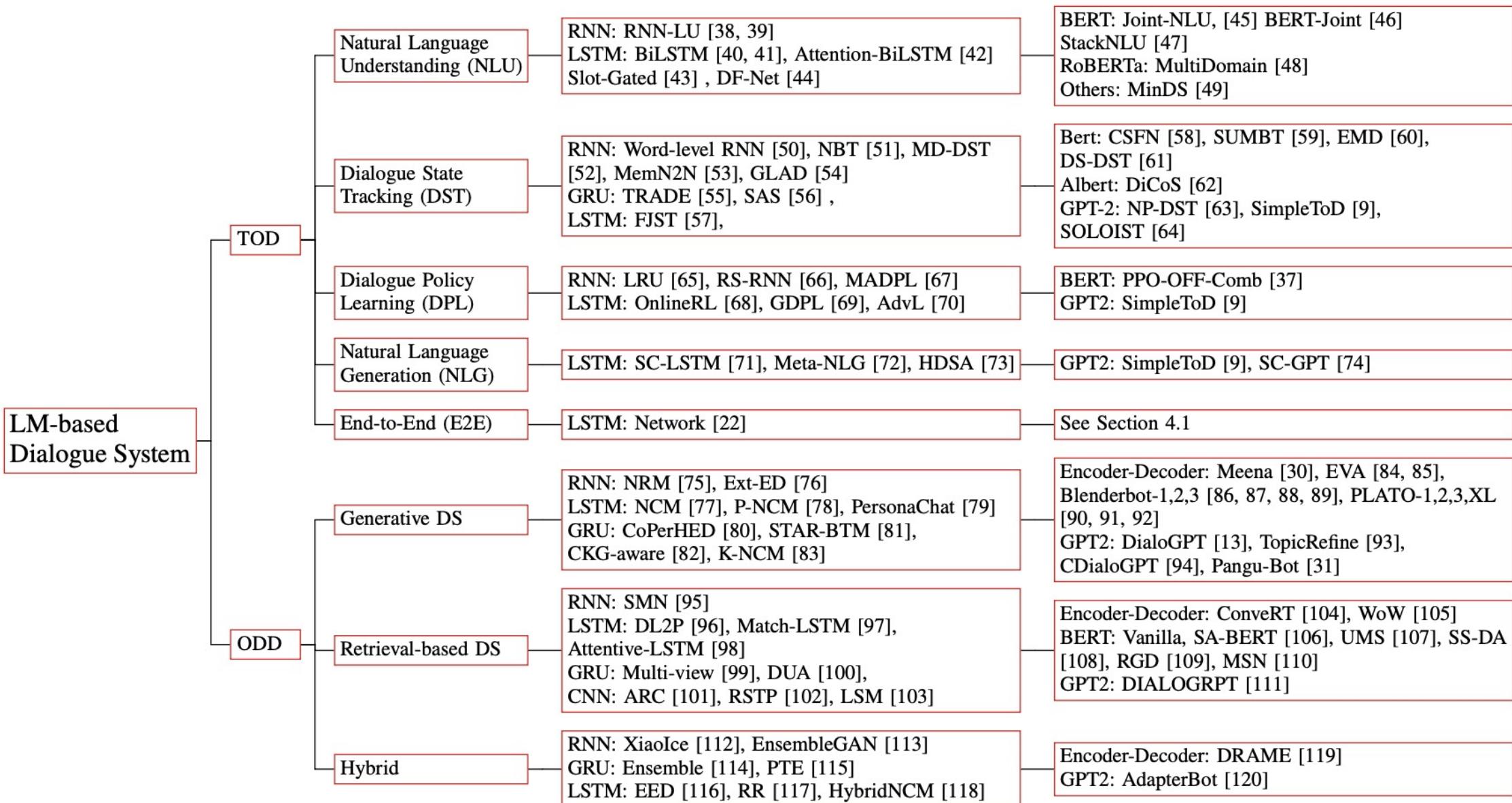
Task-Oriented Dialogue System (TDS)

Conversational Dialogue System (CDS)

Question Answering Dialogue System (QADS)

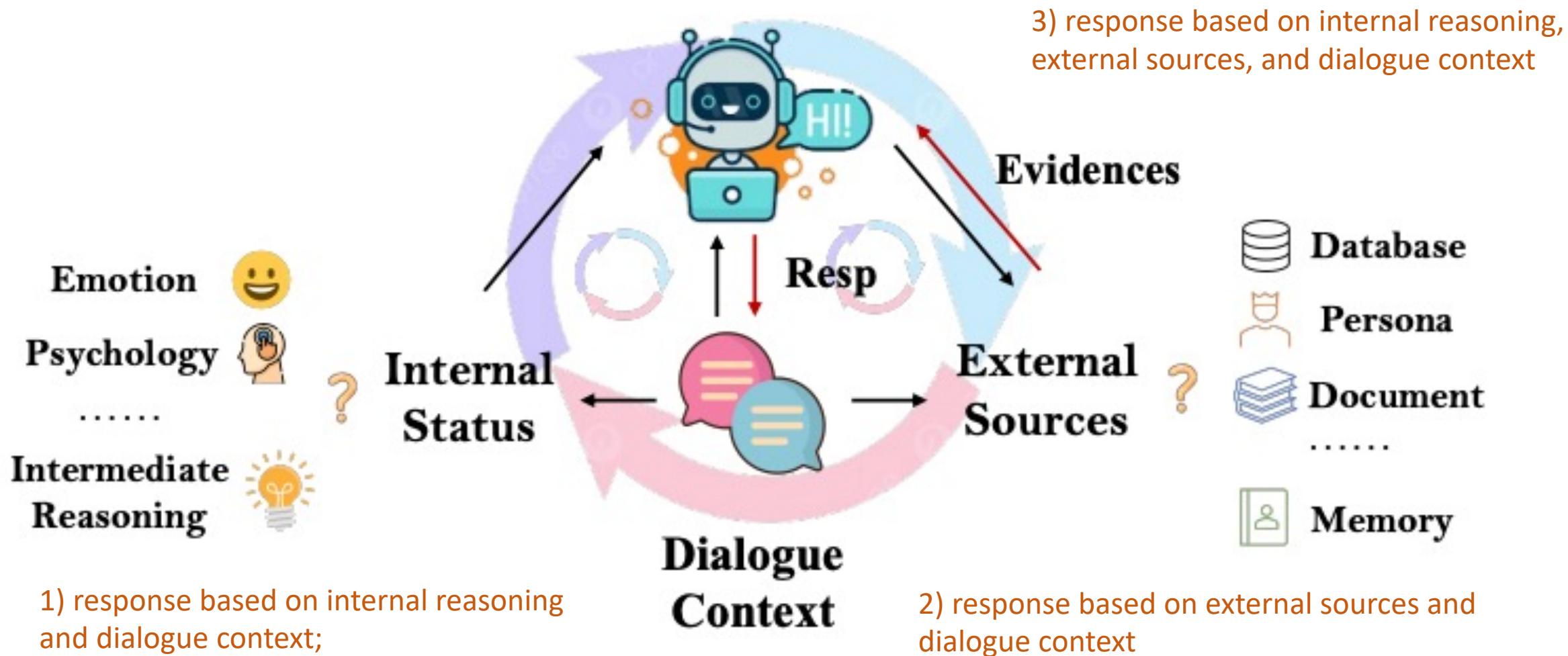
Objective	Modality	Domain	Architecture	Model
TDS	text/speech/multi	single/multi	modular	rule/statistical/neural
CDS	text/speech/multi	open	end-to-end	neural
QADS	text	multi/open	modular/end-to-end	rule/statistical/neural

Taxonomy of LM-based Dialogue System



LLM-based Dialogue System

LLM-based Dialogue System



Dialogue Subtasks

Browse SoTA > Natural Language Processing > Dialogue

Dialogue subtasks

Dialogue Generation

Dialogue Generation

9 benchmarks

78 papers with code



Task-Oriented Dialogue Systems

Task-Oriented Dialogue Systems

2 benchmarks

48 papers with code



Visual Dialog

8 benchmarks

37 papers with code



Goal-Oriented Dialog

1 benchmark

20 papers with code

Dialogue Management

12 papers with code



Dialogue Understanding

11 benchmarks

8 papers with code



Dialogue Act Classification

2 benchmarks

8 papers with code

Short-Text Conversation

Short-Text Conversation

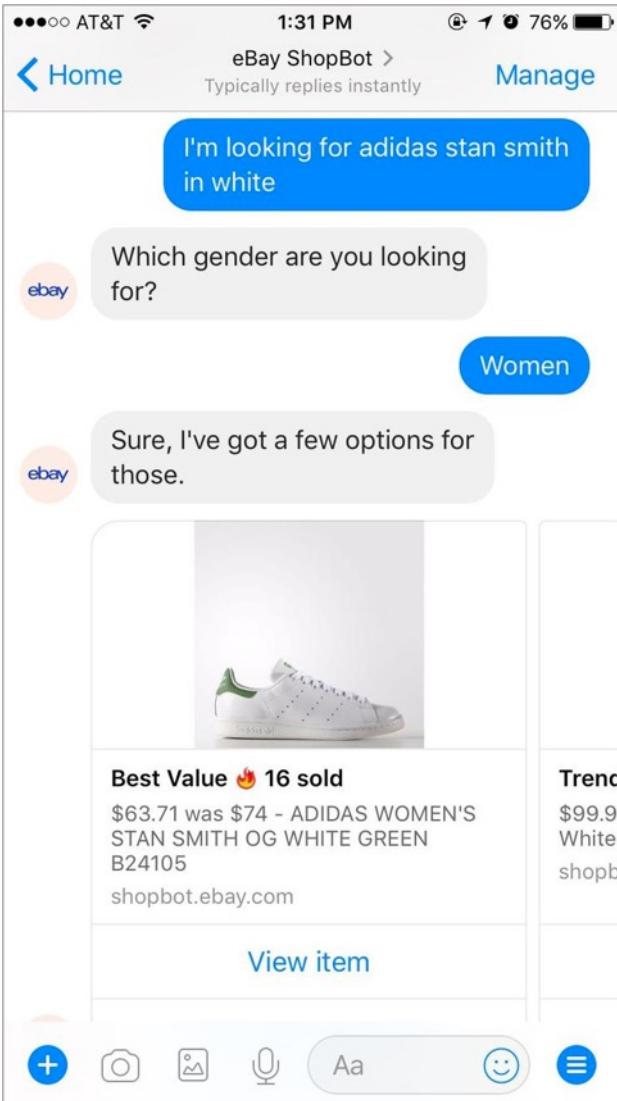
7 papers with code



Goal-Oriented Dialogue Systems

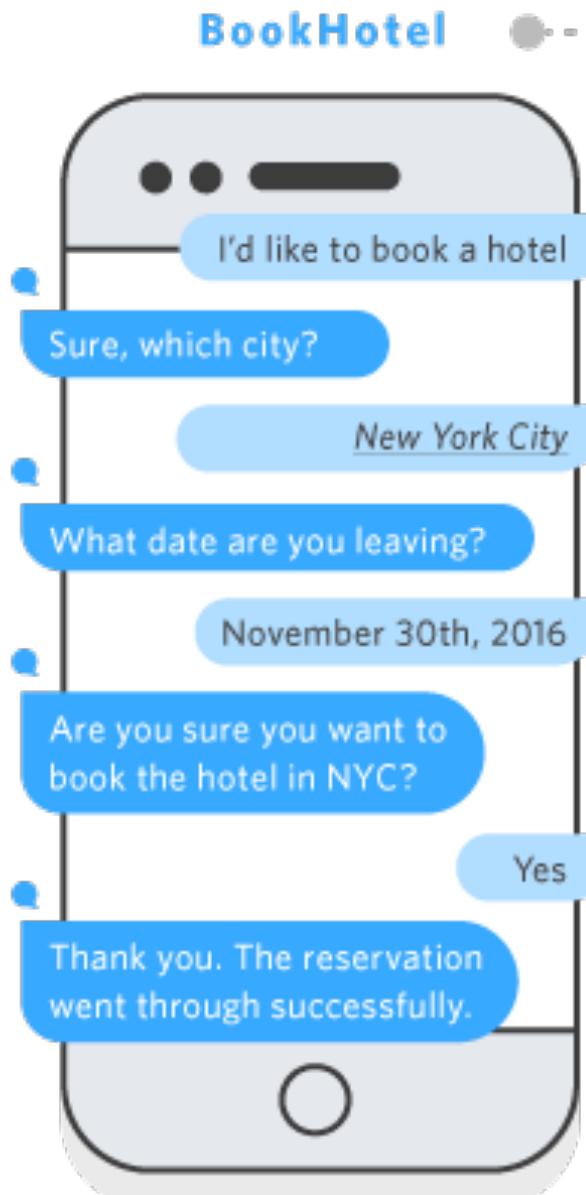
7 papers with code

Conversational Commerce: eBay AI Chatbots



Source: <https://www.forbes.com/sites/rachelarthur/2017/07/19/conversational-commerce-ebay-ai-chatbot/>

Hotel Chatbot



Intents

An intent performs an action in response to natural language user input

**Intent
Detection**

Utterances

Spoken or typed phrases that invoke your intent

Slot Filling

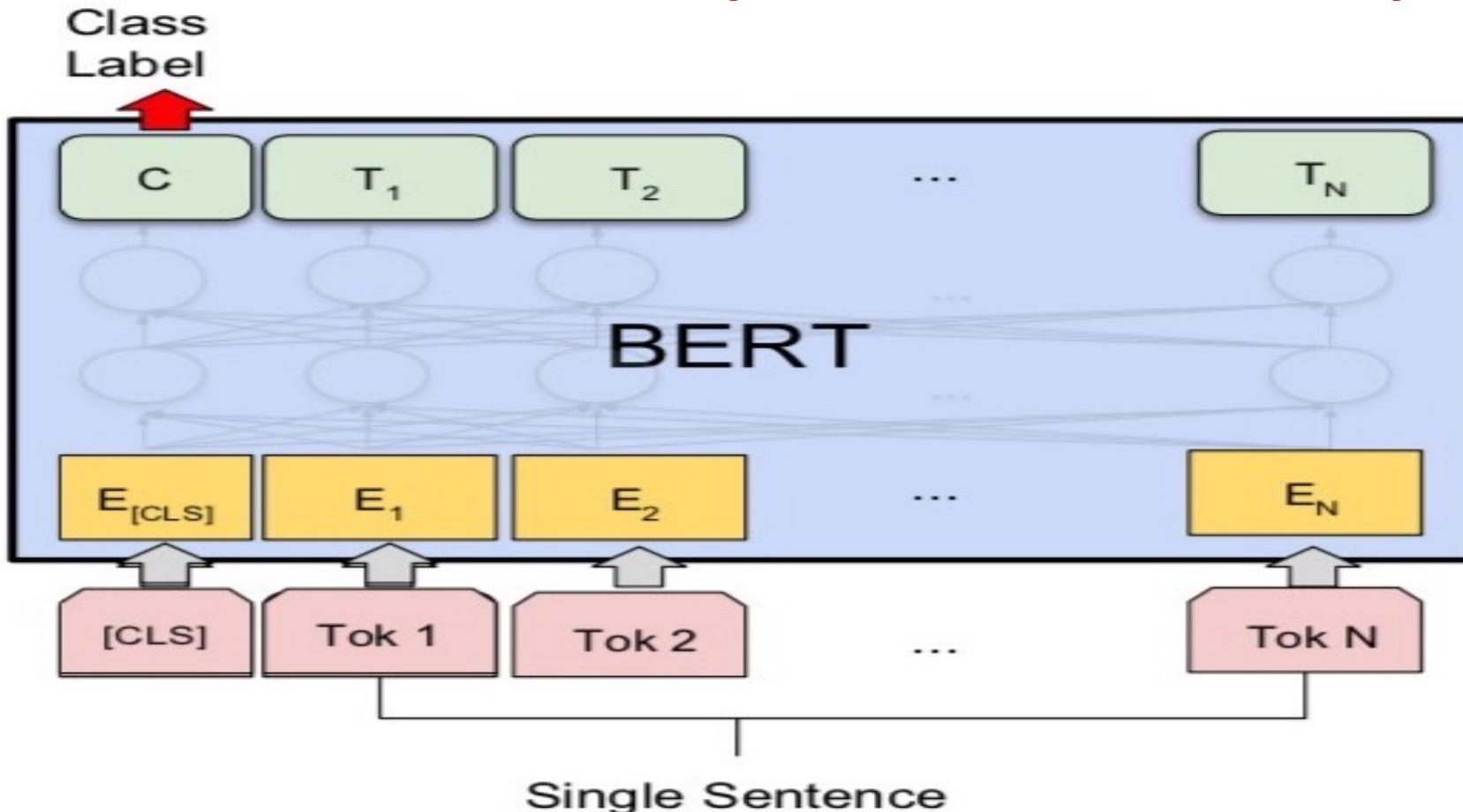
Slots

Slots are input data required to fulfill the intent

Fulfillment

Fulfillment mechanism for your intent

Fine-tuning BERT on Dialogue Intent Detection (ID; Classification)

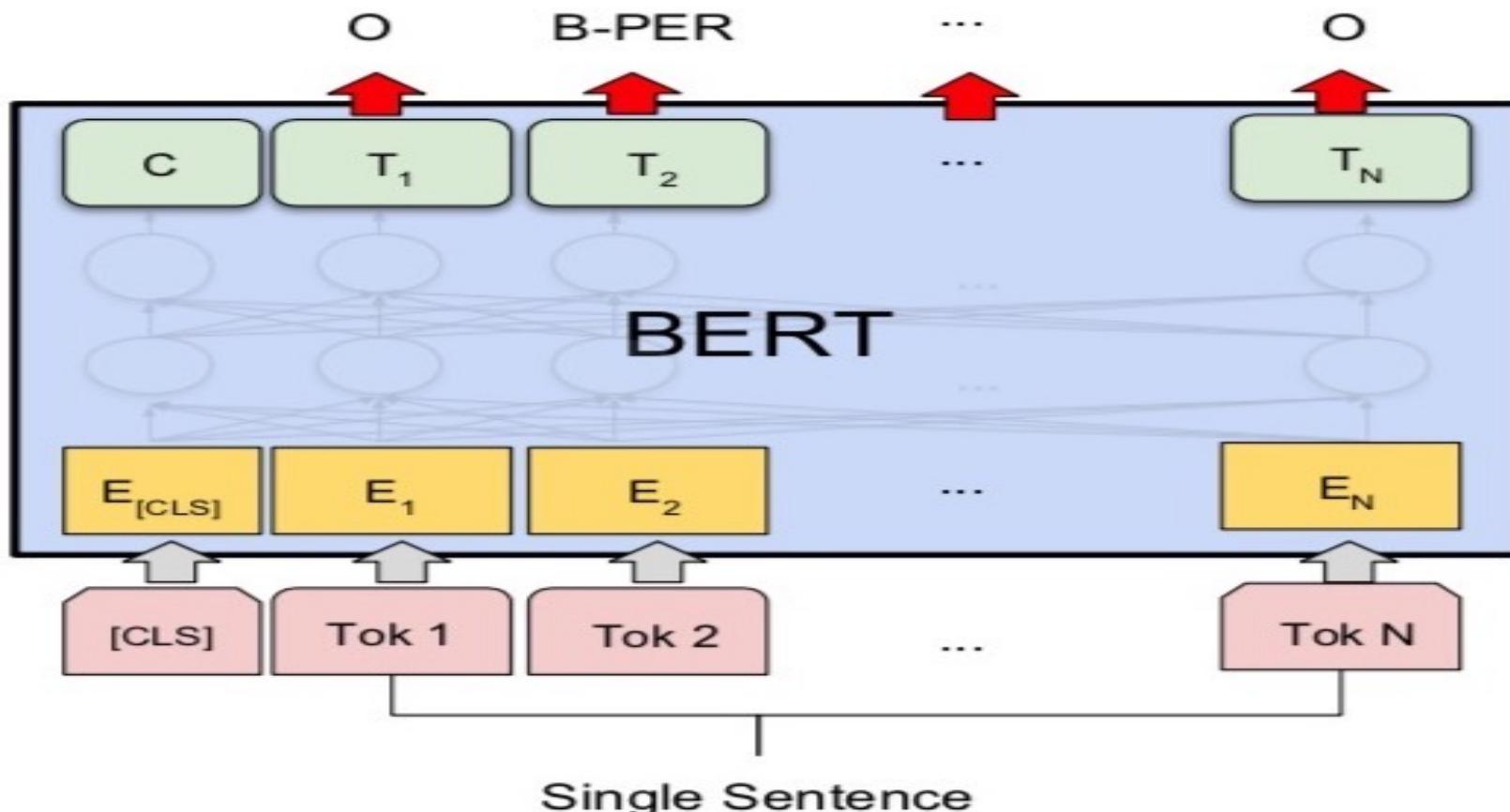


(b) Single Sentence Classification Tasks:
SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on Dialogue Slot Filling (SF)

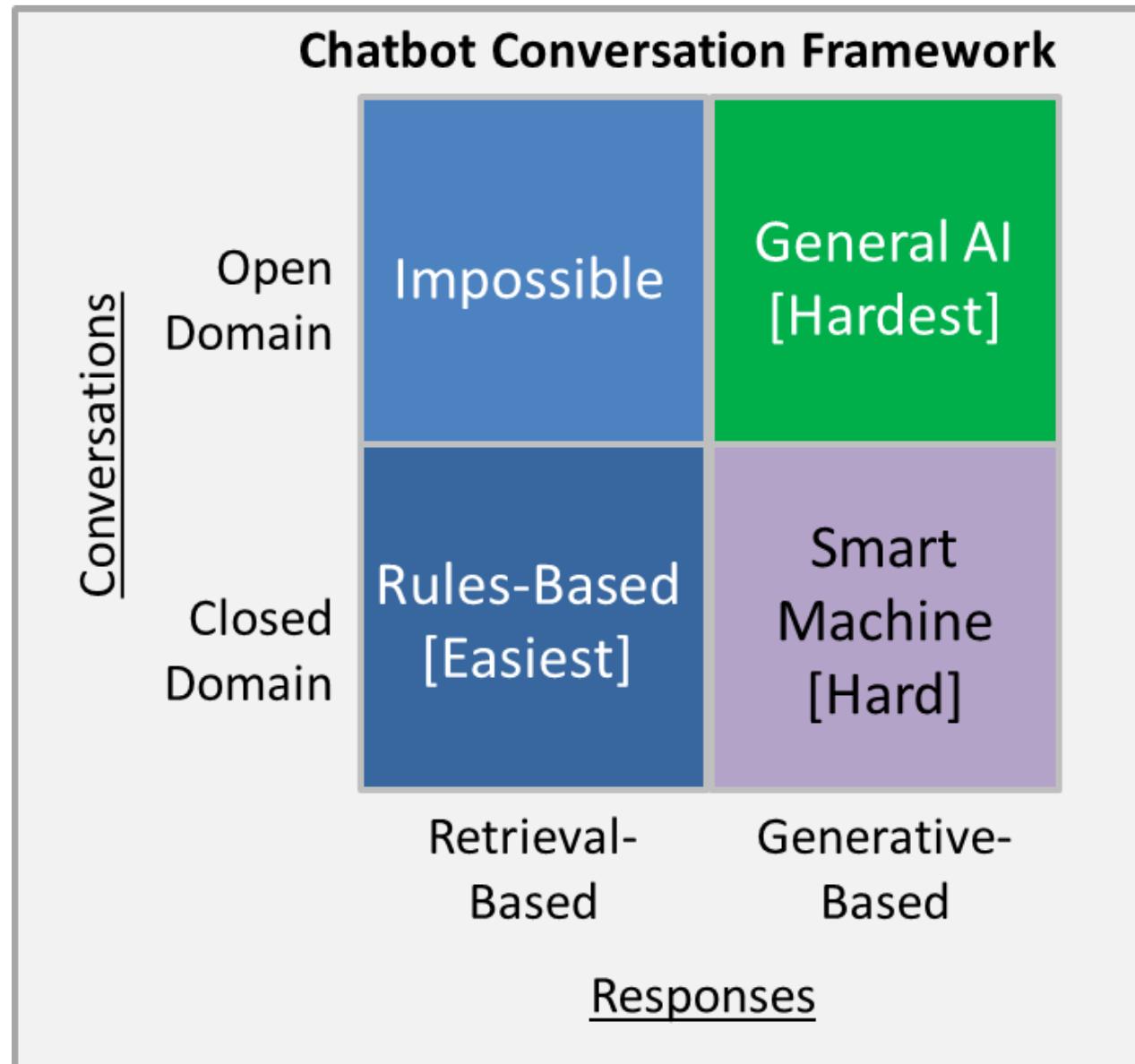


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

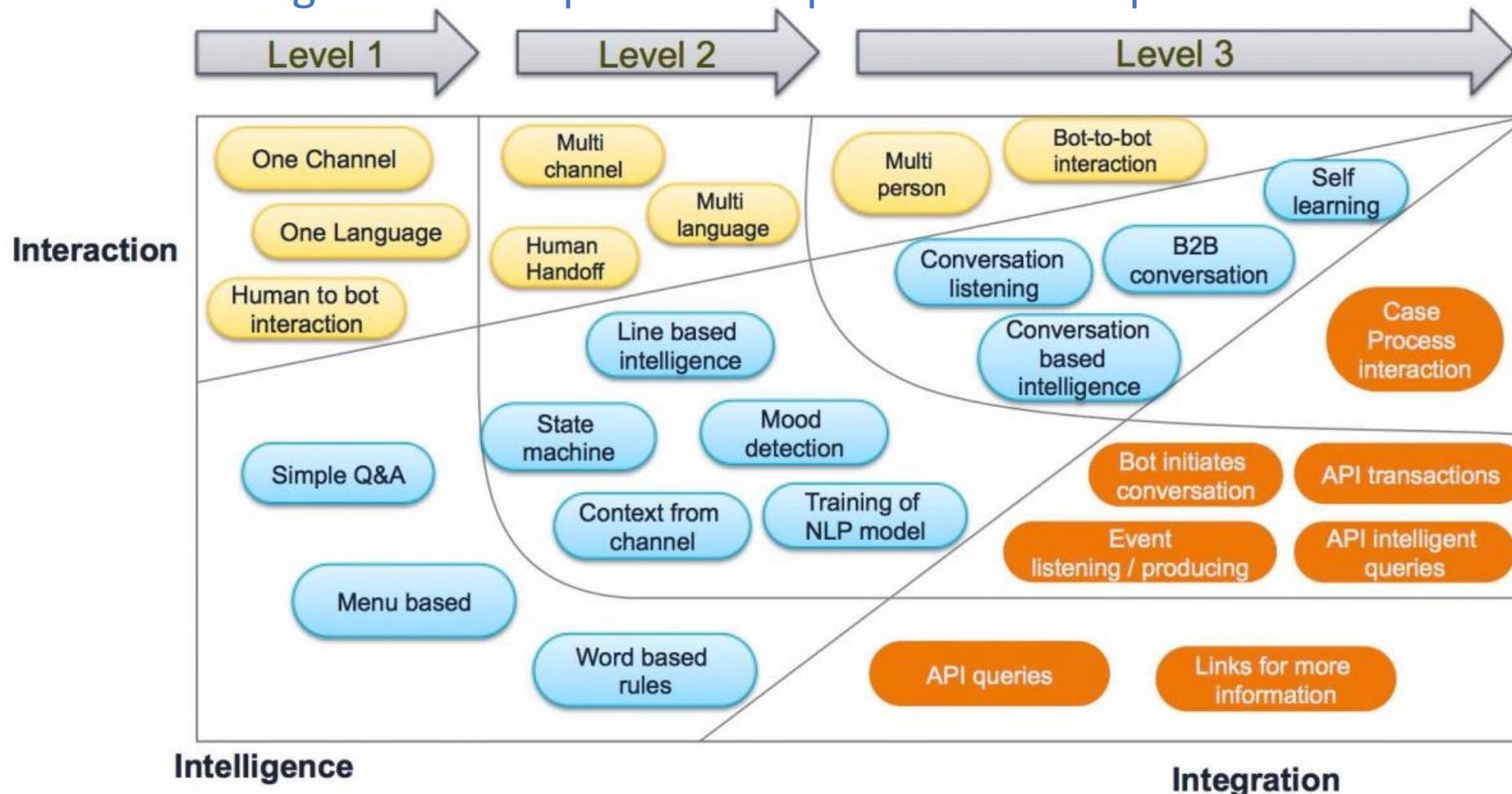
Chatbot Conversation Framework



Chatbots

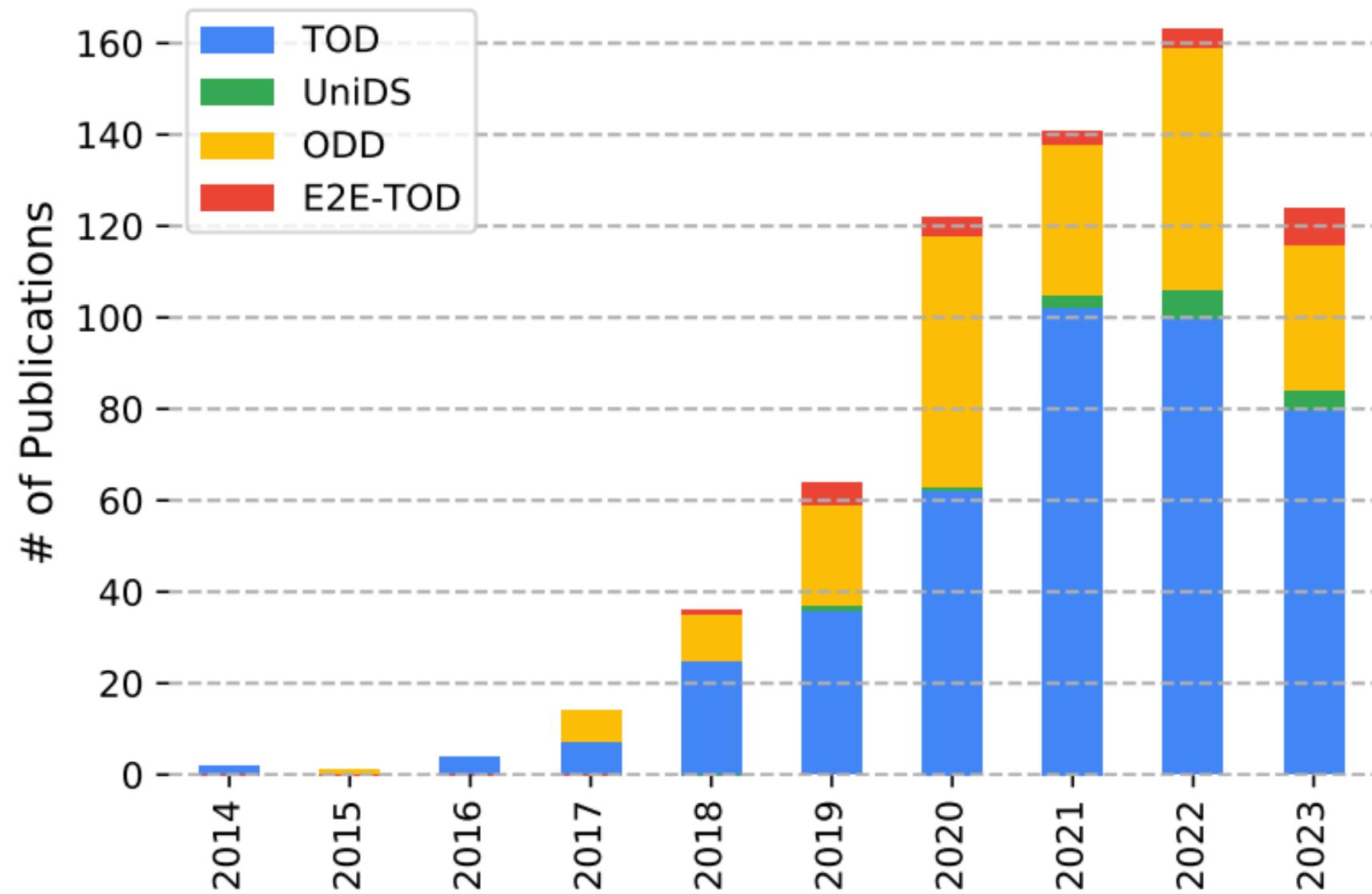
Bot Maturity Model

Customers want to have simpler means to interact with businesses and get faster response to a question or complaint.



LM-based Dialogue Systems

The number of publication at ACL Anthology conferences of different research directions in LM-based dialogue systems



End-to-End (E2E) Task-oriented Dialogue System

Model	Year	Base Model	NLU	DST	DPL	NLG	Add Tasks*	Add Data	Evaluation Dataset
NN [22]	2017	LSTM, RNN, CNN	✓	✓	✓	✓	✗	✗	Wizard-of-Oz Data
MOSS [154]	2019	GRU	✓	✓	✓	✓	✗	✗	CamRest676, LaptopNetwork
MinTL [161]	2020	T5, BART		✓		✓	✗	✗	MultiWOZ 2.0
SimpleTOD [162]	2020	GPT2		✓	✓	✓	✗	✗	MultiWOZ 2.0
SOLOIST [64]	2021	GPT2		✓		✓	Cont	✓	CamRest676, MultiWOZ 2.0
UBAR [163]	2021	DistilGPT2		✓	✓	✓	✗	✗	MultiWOZ 2.0, 2.1
MTTOD [164]	2021	T5		✓		✓	SP	✗	MultiWOZ 2.0, 2.1
AuGPT [159]	2021	GPT2		✓		✓	BC, SAC, IC	✓	MultiWOZ 2.0, 2.1
GALAXY [156]	2022	UniLM			✓	✓	RS, CR	✓	In-Car, MultiWOZ 2.0, 2.1
PPTOD [155]	2022	T5	✓	✓	✓	✓	✗	✓	MultiWOZ 2.0 2.1
OPERA [160]	2022	T5				✓	QA	✗	OB-MultiWOZ
SPACE-3 [157]	2022	UniLM	✓		✓	✓	MLM, SRM	✓	MultiWOZ, CamRest676, In-Car

High-level of fusion between TOD and ODD

Task-oriented DS (TOD) Open-domain DS (ODD)

Model	Task	Type of ODD	Knowledge sources			Training method
			Database	Web	PLMs	
Q-TOD [135]	TOD	X	✓	X	X	End-to-end
Internet-Augmented [88]	ODD	KG Chit-Chat	X	✓	X	End-to-end
UniDS [134], FusedChat [170] SalesBot [168]	TOD + ODD	Chit-Chat	✓	X	X	End-to-end
HyKnow [173], KETOD [169]	TOD + ODD	KG Chit-Chat	✓	✓	X	End-to-end
GODEL [153]	TOD + ODD	Question Answering, KG Chit-Chat	✓	✓	X	End-to-end
DuClarifyDial [8]	TOD + ODD	Chit-Chat, Question Answering, KG Chit-Chat	✓	✓	X	End-to-end
MDS [174]	TOD + ODD	Question Answering	✓	✓	X	End-to-end
OPERA [160]	TOD + ODD	Question Answering	✓	✓	✓	End-to-end
DialogStudio [133]	TOD + ODD	Dial-Sum, NLU, Conv-Rec, KG Chit-Chat	✓	✓	X	End-to-end

Dialogue Models based on Large Language Models

Dialogue Models	Release Date	Base Model	# of Params	Window size	Trainig Tokens	Language Preference	Source Type
BlenderBot3[89]	Aug, 2022	OPT	3B/30B/175B	X	180B	EN	Open
ChatGPT	Dec, 2022	GPT3.5 [176]	175B	4k/16k	X	EN	Closed
Bing Chat	Feb, 2023	GPT4 [177]	X	8k/32k	X	EN	Closed
LLaMA-Chat [35]	Feb, 2023	LLaMA	7B/13B/33B/65B	2k	1.4T	EN	Open
Alpaca [178]	Mar, 2023	LLaMA	7B	2k	1.4T	EN	Open
Vicuna [179]	Mar, 2023	LLaMA	7B/13B	2k	1.4T	EN	Open
ChatGLM [180]	Mar, 2023	GLM	6B/12B	2k	1T	CN	Open
Slack	Mar, 2023	Claude	X	200k	2.0T	EN	Closed
Bard	Mar, 2023	PALM	540B	X	X	EN	Closed
Falcon-Chat [181]	Jun, 2023	Falcon	7.5B/40B/180B	2k	1.5T/1T/3.5T	EN	Open
MPT-Chat [182]	Jun, 2023	MPT	7B/30B	8k	1T	EN	Closed
InternLM-Chat [183]	Jun, 2023	InternLM	7B/20B	8k	2.3T	CN	Open
Baichuan-Chat [184],	Jun, 2023	Baichuan	7B/13B/53B	4k	1.2T	CN	Open (7B/13B)
ChatGLM2 [180]	Jun, 2023	GLM	6B/12B	32k	1T	CN	
LLaMA2-Chat [185]	Jul, 2023	LLaMA2	7B/13B/34B/70B	4k	2.0T	EN	Open
Atom-Chat	Aug, 2023	LLaMA2	7B/13B	4k	1.0T	CN	Open
Qwen-Chat [186]	Aug, 2023	QWen	7B/14B	8k	2.4T	CN	Open
Baichuan2-Chat [184]	Sep, 2023	Baichuan2	7B/13B	4k	2.6T	CN	Open
ChatGLM3	Oct, 2023	GLM	6B	32k	/	CN	Open

Dialogue System

Objective	Modality	Domain	Architecture	Model
TDS	text/speech/multi	single/multi	modular	rule/statistical/neural
CDS	text/speech/multi	open	end-to-end	neural
QADS	text	multi/open	modular/end-to-end	rule/statistical/neural

Dialogue System

Extracted information for the example sentence
show restaurant at New York tomorrow
by the module natural language understanding

Sentence	show	restaurant	at	New	York	tomorrow
Slots	O	O	O	B-desti	I-desti	B-date
Intent	Find Restaurant					
Category	Order					

Dialogue System

Selected intelligent virtual assistants, commercial frameworks and research dialogue systems categorised by objective, modality, domain, architecture and model

	System name	Objective	Modality	Domain	Architecture	Model	Specifications
Intelligent Virtual Assistants	Task-oriented Conversational Question Answering	-	-	-	-	-	Open source
	Text Speech Multi-modal	Text Speech Multi-modal	Single Multi Open	Modular End-to-end	Rule-based Statistical Neural		
Amazon Alexa ^a	• - -	• • -	- • -	- • -	•	• • •	-
Bixby ^b	• - -	• • -	- • -	x x	• • •	-	-
Cortana [48]	• - -	• • -	- • -	• -	• • •	-	-
Google Assistant ^c	• - -	• • -	- • -	x x	• • •	-	-
Siri ^d	• - -	• • -	- • -	x x	• • •	-	-
XiaoIce [79]	- • -	• • -	- - •	• -	• - •	-	Emotional module, Chinese
Commercial Frameworks	Cerence Studio ^e	• - -	• • -	- • -	• -	- • •	-
	Conversational AI ^f	• - -	• • -	- • -	- •	x x x	-
	Dialogflow ^g	• - -	• • -	- • -	- •	• - •	-
	Dragon ^h	- - -	- • -	- • •	x x	- - •	ASR (dictation software)
	LUIS ⁱ	• - -	• • -	- • -	x x	- - •	-
	Nuance Mix ^j	• - -	• • -	- • •	• -	• - •	-
	Rasa ^k	• - -	• - -	- • -	• -	• • •	NLU&DM, also research system
	SemVox [3]	• - -	- • •	- • -	• -	x x x	also IVA
	Watson Assistant ^l	• - -	• • -	- • -	- - -	• -	-

Dialogue System

Selected intelligent virtual assistants, commercial frameworks and research dialogue systems categorised by objective, modality, domain, architecture and model

	System name	Objective	Modality	Domain	Architecture	Model	Specifications
Research Dialogue Systems		Task-oriented Conversational Question Answering	Text Speech Multi-modal	Single Multi Open	Modular End-to-end	Rule-based Statistical Neural	Open source
	DenSPI [52]	- - - •	• - - -	- - - •	- - •	- - - •	•
	DrQA [4]	- - - •	• - - -	- - - •	• -	• - - •	•
	R ³ [65]	- - - •	• - - -	- - - •	• -	- - - •	•
	SmartWeb [56]	- - - •	• - • •	- - •	• -	• - - -	•
	ELIZA [67]	- - • -	• - - -	- - •	• -	• - - -	•
	ConvLab [27]	• - - -	• - - -	- - •	• - •	- - - •	•
	DialogOS [23]	• - - -	- - • -	- - •	• -	• - - -	•
	Nemo [25]	• • • •	- - • -	- - •	• -	- - - •	•
	ParlAI [40]	• • • •	• - - •	- - •	• -	- - - •	•
	Plato [47]	• - - -	• - - -	- - •	• • •	• - - -	•
	PyDial [61]	• - - -	• - • -	- - •	• -	- - - •	•
	ReTiCo [38,39]	• - - -	• - • -	- - •	• -	- - • •	•
	Siam-dp [42]	• - - •	- - • •	- - •	• -	- - - •	•

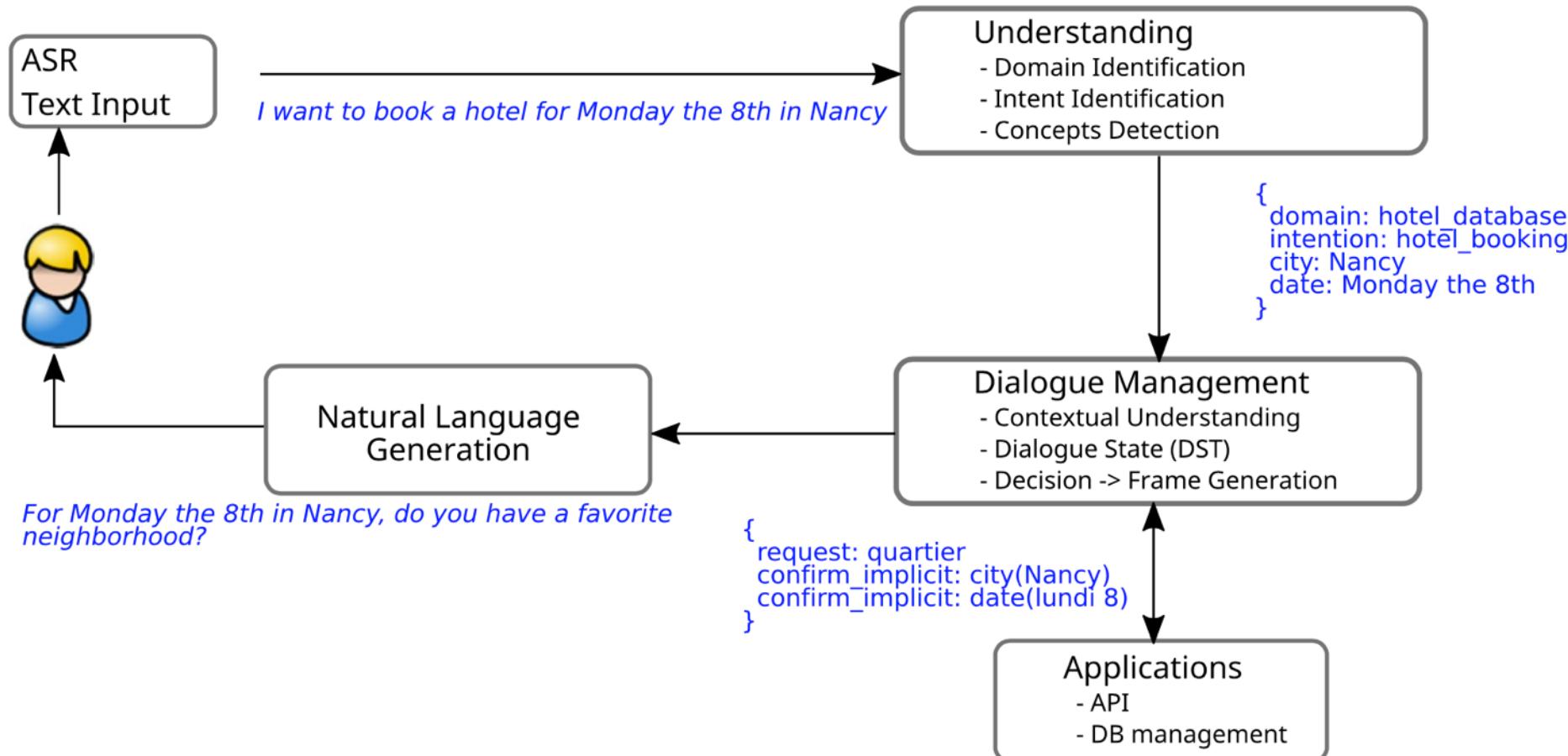
Properties relevant for implementation of research dialogue systems

System	Application Framework	Code Availability	Programming Language	Windows	MacOS	Linux	Tutorial	Demonstration	Last Activity (Releases, Commits, Issues)	Purpose
DenSPI	• -	•	Python	•	•	•	-	•	06/2022	Real-time, Wikipedia-based
DrQA	• -	•	Python	-	•	•	-	•	11/2022	Machine reading at scale
R ³	• -	•	Lua, Python	•	•	•	-	-	04/2018	Reinforcement learning
SmartWeb	• -	-	Java	•	•	•	-	-	2014	Multi-modal access to Web
ELIZA	• •	•	Python	•	•	•	-	•	1966	Simulate interlocutors
ConvLab	- •	•	Python	-	•	•	•	-	04/2023	Reusable experimental setup
DialogOS	- •	•	Java	•	•	•	-	-	12/2022	Student projects
Nemo	- •	•	Python	•	•	•	•	-	04/2023	Reuse code and models
ParlAI	- •	•	Python	-	•	•	•	•	04/2023	Share, train and test systems
Plato	- •	•	Python	•	•	•	•	-	09/2020	Multi-agent setting
PyDial	- •	•	HTML	•	•	•	•	•	04/2022	Research on modules
ReTiCo	- •	•	Python	•	•	•	-	-	04/2023	Real time, incremental
Siam-dp	- •	•	Java	•	•	•	-	-	02/2017	Multi-modal development

Task-Oriented Dialogue System

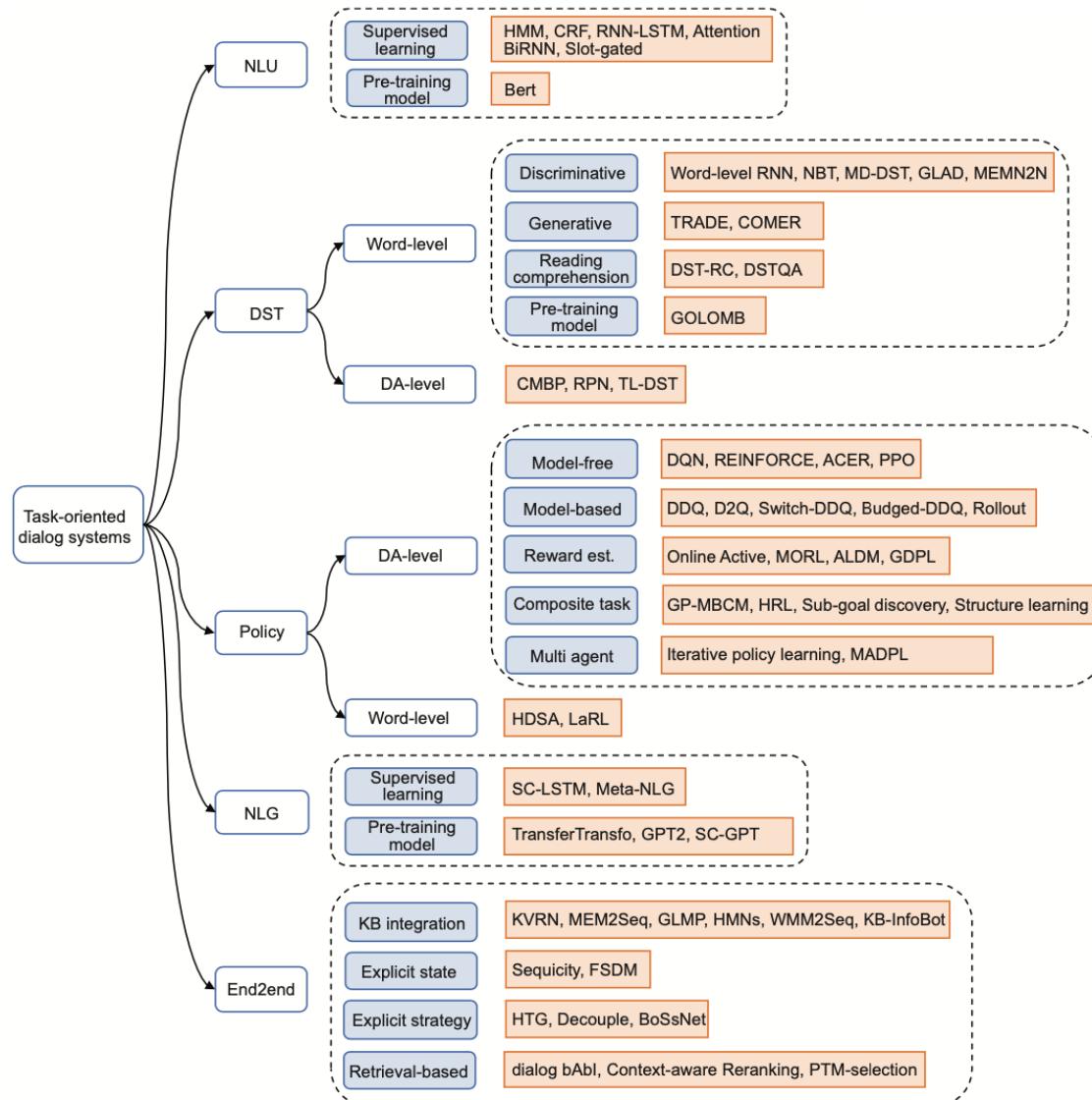
Task-Oriented Dialogue System

(Deriu et al., 2021)



Task-Oriented Dialogue Systems

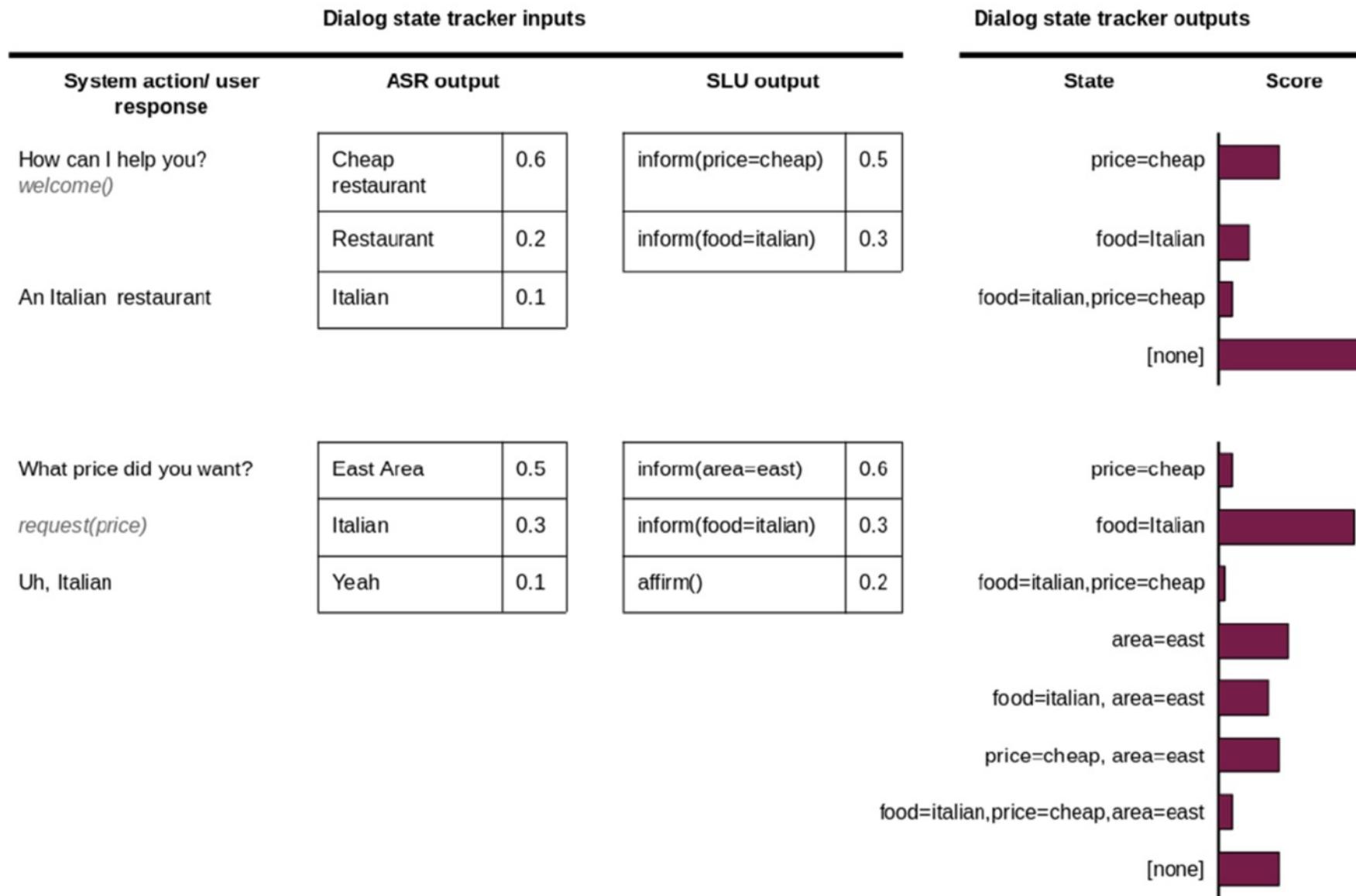
(Zhang et al., 2020)



Source: Zhang, Zheng, Ryuichi Takanobu, Qi Zhu, Minlie Huang, and Xiaoyan Zhu (2020).

"Recent advances and challenges in task-oriented dialog systems." Science China Technological Sciences (2020): 1-17.

Dialog State Tracker (DST)



Source: Deriu, Jan, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak (2021). "Survey on evaluation methods for dialogue systems." Artificial Intelligence Review 54, no. 1 (2021): 755-810.

Dialogue Acts

(Young et al., 2010)

Dialogue act	Description
hello($a = x, b = y, \dots$)	Open a dialogue and give info $a = x, b = y, \dots$
inform($a = x, b = y, \dots$)	Give information $a = x, b = y, \dots$
request($a, b = x, \dots$)	Request value for a given $b = x, \dots$
reqalts($a = x, \dots$)	Request alternative with $a = x, \dots$
confirm($a = x, b = y, \dots$)	Explicitly confirm $a = x, b = y, \dots$
confreq($a = x, \dots, d$)	Implicitly confirm $a = x, \dots$ and request value of d
select($a = x, a = y$)	Select either $a = x$ or $a = y$
affirm($a = x, b = y$)	Affirm and give further info $a = x, b = y, \dots$
negate($a = x$)	Negate and give corrected value $a = x$
deny($a = x$)	Deny that $a = x$
bye()	Close a dialogue

Sample Dialogue Acts

Utterance	Dialogue Act
U: Hi, I am looking for somewhere to eat	hello(task = find,type=restaurant)
S: You are looking for a restaurant. What type of food?	confreq(type = restaurant,food)
U: I'd like an Italian somewhere near the museum.	inform(food = Italian,near=museum)
S: Roma is a nice Italian restaurant near the museum.	inform(name = "Roma", type = restaurant, food = Italian, near = museum)
U: Is it reasonably priced?	confirm(pricerange = moderate)
S: Yes, Roma is in the moderate price range.	affirm(name = "Roma", pricerange = moderate)
U: What is the phone number?	request(phone)
S: The number of Roma is 385456.	inform(name = "Roma", phone = "385456")
U: Ok, thank you goodbye.	bye()

**Dialogue
on
Airline Travel
Information System
(ATIS)**

The ATIS (Airline Travel Information System) Dataset

<https://www.kaggle.com/siddhadev/atis-dataset-from-ms-cntk>

Sentence	what	flights	leave	from	phoenix
Slots	O	O	O	O	B-fromloc
Intent	atis_flight				

Training samples: 4978

Testing samples: 893

Vocab size: 943

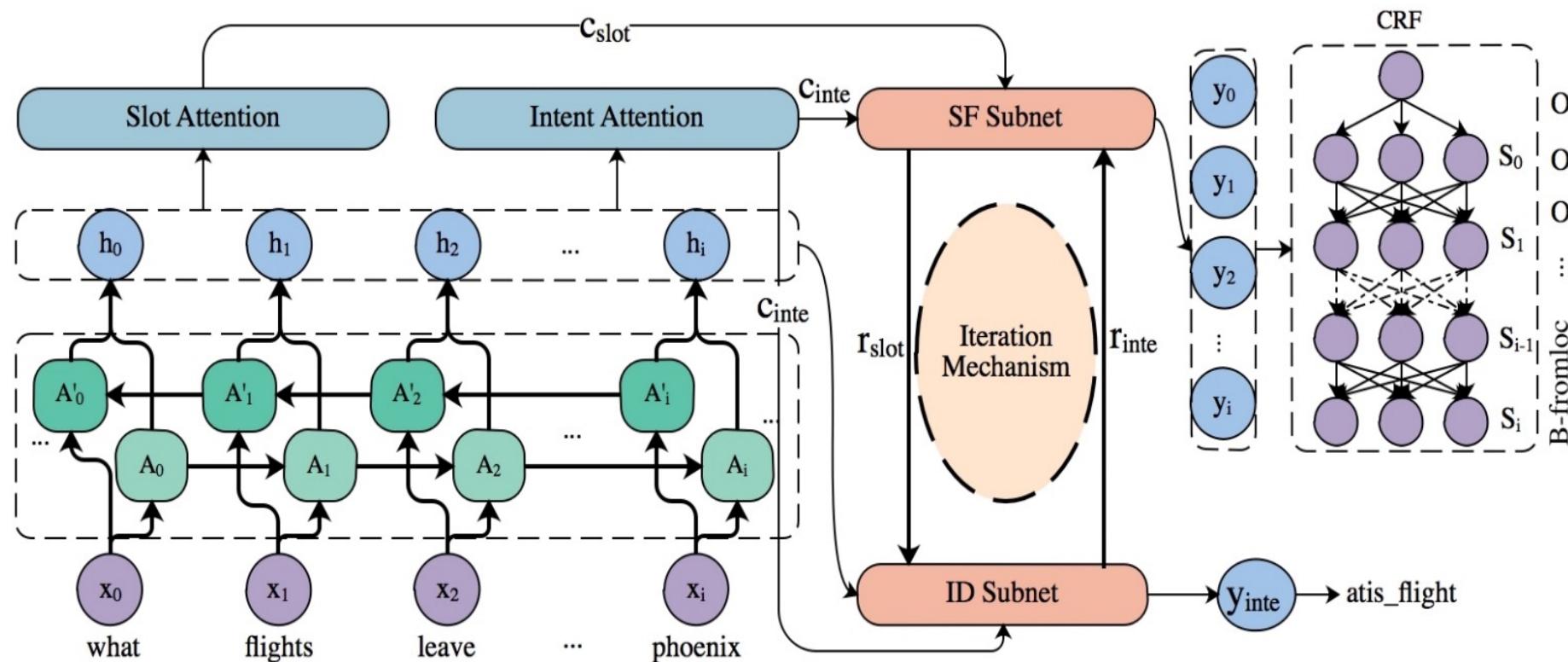
Slot count: 129

Intent count: 26

SF-ID Network (E et al., 2019)

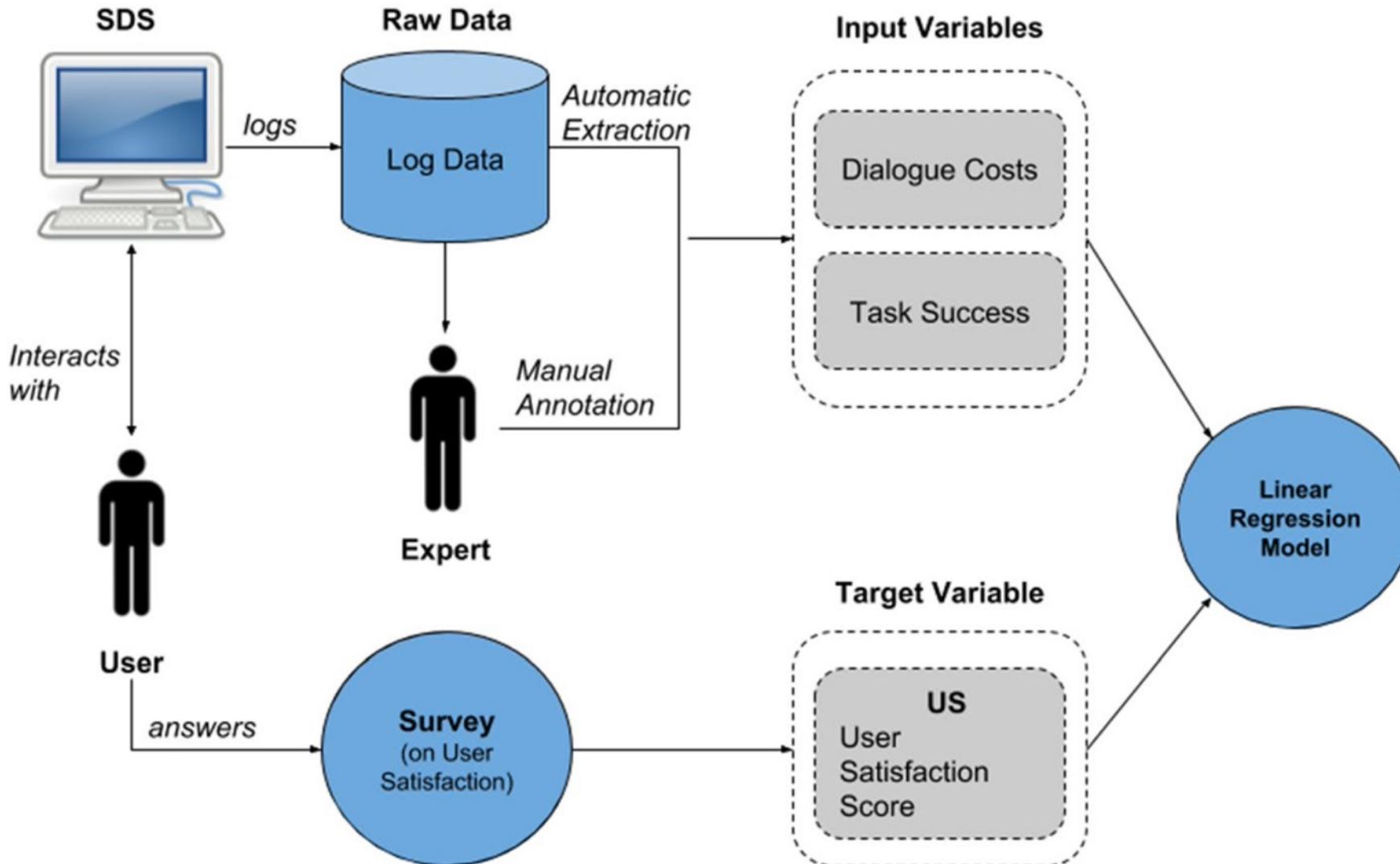
Slot Filling (SF) Intent Detection (ID)

A Novel Bi-directional Interrelated Model for Joint Intent Detection and Slot Filling



PARAdigm for Dialog System Evaluation

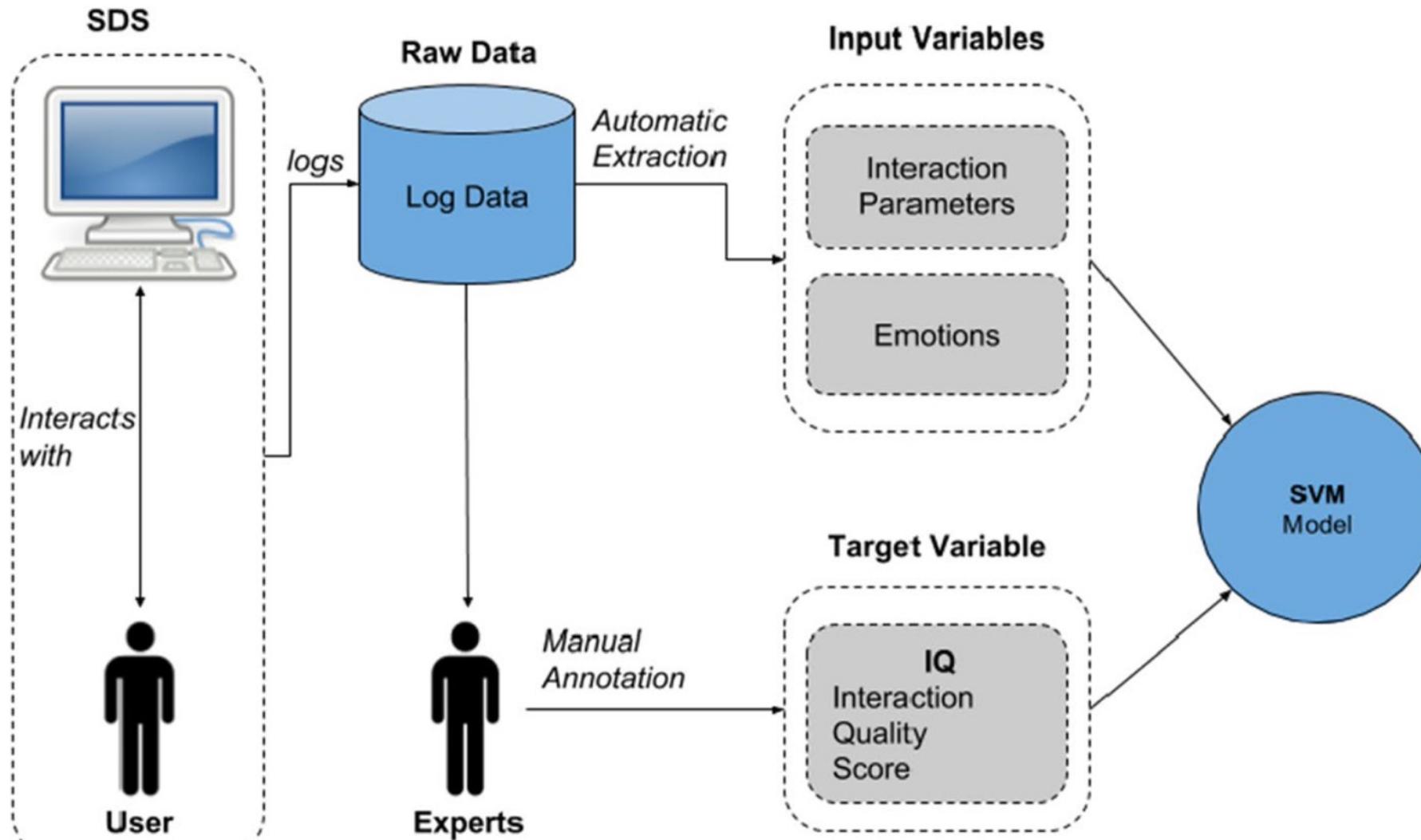
PARADISE Framework (Walker et al. 1997)



Source: Deriu, Jan, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak (2021). "Survey on evaluation methods for dialogue systems." Artificial Intelligence Review 54, no. 1 (2021): 755-810.

Interaction Quality procedure

(Schmitt and Ultes, 2015)



Datasets for task-oriented dialogue systems

Name	Topics	# dialogues	Reference
DSTC1	Bus schedules	15,000	(Williams et al. 2013)
DSTC2	Restaurants	3000	(Henderson et al. 2014)
DSTC3	Tourist information	2265	(Henderson et al. 2013a)
DSTC4 & DSTC5	Tourist information	35	(Kim et al. 2016)
DSTC6	Restaurant reservation	–	(Perez et al. 2017)
DSTC7 (Flex Data)	Student guiding	500	(Gunasekara et al. 2019)
DSTC8 (MetaLWOz)	47 domains	37,884	(Lee et al. 2019)
DSTC8 (Schema-Guided)	20 domains	22,825	(Rastogi et al. 2019)
MultiWOZ	Tourist information	10,438	(Budzianowski et al. 2018)
Taskmaster-1	6 domains	13,215	(Byrne et al. 2019)
MultiDoGo	6 domains	86,698	(Peskov et al. 2019)

Restaurants Dialogue Datasets

- **MIT Restaurant Corpus**
 - <https://groups.csail.mit.edu/sls/downloads/restaurant/>
- **CamRest676**
(Cambridge restaurant dialogue domain dataset)
 - <https://www.repository.cam.ac.uk/handle/1810/260970>
- **DSTC2 (Dialog State Tracking Challenge 2 & 3)**
 - <http://camdial.org/~mh521/dstc/>

CrossWOZ: A Large-Scale Chinese Cross-Domain Task-Oriented Dialogue Dataset

MultiWOZ	CrossWOZ
usr: I'm looking for a college type attraction. ... usr: I would like to visit in town centre please. ... usr: Can you find an Indian restaurant for me that is also in the town centre?	usr: 你好，可以帮我推荐一个评分是4.5分以上的景点吗? Hello, could you recommend an attraction with a rating of 4.5 or higher? sys: 天安门城楼, 篱街小吃和北京欢乐谷都是很不错的地方呢。 Tiananmen, Gui Street, and Beijing Happy Valley are very nice places. usr: 我喜欢北京欢乐谷, 你知道这个景点周边的酒店都是什么吗? I like Beijing Happy Valley. What hotels are around this attraction? sys: 那可多了, 有A酒店, B酒店, C酒店。 There are many, such as hotel A, hotel B, and hotel C.
Schema	usr: 太好了, 我正打算在景点附近找个酒店住宿呢, 知道哪家评分是4分以上, 提供叫醒服务的不? Great! I am planning to find a hotel to stay near the attraction . Which one has a rating of 4 or higher and offers wake-up call service?
usr: I want a hotel in San Diego and I want to check out on Thursday next week. ... usr: I need a one way flight to go there.	

CrossWOZ:

A Large-Scale Chinese Cross-Domain Task-Oriented Dialogue Dataset

Type	Single-domain goal					Multi-domain goal		
Dataset	DSTC2	WOZ 2.0	Frames	KVRET	M2M	MultiWOZ	Schema	CrossWOZ
Language	EN	EN	EN	EN	EN	EN	EN	CN
Speakers	H2M	H2H	H2H	H2H	M2M	H2H	M2M	H2H
# Domains	1	1	1	3	2	7	16	5
# Dialogues	1,612	600	1,369	2,425	1,500	8,438	16,142	5,012
# Turns	23,354	4,472	19,986	12,732	14,796	115,424	329,964	84,692
Avg. domains	1	1	1	1	1	1.80	1.84	3.24
Avg. turns	14.5	7.5	14.6	5.3	9.9	13.7	20.4	16.9
# Slots	8	4	61	13	14	25	214	72
# Values	212	99	3,871	1363	138	4,510	14,139	7,871

Source: Zhu, Qi, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. "Crosswoz: A large-scale chinese cross-domain task-oriented dialogue dataset." arXiv preprint arXiv:2002.11893 (2020).

Task-Oriented Dialogue

Initial user state (=user goal)

id=1(Attraction): fee=free,
name=?, nearby hotels=?

id=2(Hotel): **name=near (id=1)**,
wake-up call=yes, rating=?

id=3(Taxi): **from=(id=1), to=(id=2)**,
car type=? plate number=?

Final user state

id=1 (Attraction): name=Tiananmen Square,
fee=free, nearby hotels=[Beijing Capital
Hotel, Guidu Hotel Beijing]

id=2 (Hotel): **name=Beijing Capital Hotel**,
wake-up call=yes, rating=4.6

id=3 (Taxi): **from=Tiananmen Square,**
to=Beijing Capital Hotel,
car type=#CX, plate number=#CP

id=1(Attraction): fee=free,
name=?, nearby hotels=?

你好, 帮我找一个免费的景点。
Hello, find me a free attraction please.

1

Attraction: fee=free

天安门广场怎么样?
How about Tiananmen Square?

...

id=2(Hotel): **name=near (id=1)**,
wake-up call=yes, rating=?

多谢。我还想在天安门广场旁边找一家有叫醒服务
的酒店住宿。
Thanks. I'm also looking for a place to stay near
Tiananmen Square. It must have wake-up call.

5

Hotel: nearby=Tiananmen Square,
facilities=[wake-up call]

向您推荐北京首都宾馆。
I recommend you Beijing Capital Hotel.

...

id=3(Taxi): **from=Tiananmen Square,**
to=Beijing Capital Hotel,
car type=? plate number=?

帮我叫一辆从天安门广场到酒店的出租车吧, 告
诉我车型和车牌。
Book a taxi from Tiananmen Square to the
hotel. Tell me the car type and plate number.

7

Taxi: from=Tiananmen Square,
to=Beijing Capital Hotel

好的。车型是 #CX, 车牌是 #CP。
Ok. Car type is #CX. Plate number is #CP.

An example dialog from the test set for MultiWOZ (en→zh) sub-task

Speaker	Utterance	Dialog State Update
User	Hello! I am looking for a local guesthouse in the centre. 你好！我在市中心找一家本地宾馆。	hotel: {area: centre, type: guesthouse} 旅馆: {区域: 中心, 类型: 宾馆}
System	OK. I am glad to recommend Alexander Bed and Breakfast to you. 好的。这边很高兴向您推荐亚历山大住宿加早餐旅馆。	
User	Where is it? 它在哪里？	hotel: {name: alexander bed and breakfast} 旅馆: {名称: 亚历山大住宿加早餐旅馆}
System	It is at 56 Saint Barnabas Road. 圣巴纳巴斯路56号。	
User	I also wish to have a meal in a local European restaurant in the centre. 我还想在市中心的一家本地欧洲餐厅吃饭。	restaurant: {food: european, area: centre} 餐厅: {食物: 欧洲的, 区域: 中心}
System	You can choose Eraina. 您可以选择伊莱娜。	
User	Please give me its address. 那请给我它的地址。	restaurant: {name: eraina} 餐厅: {名称: 伊莱娜}
System	It is in Free School Lane City Centre. 市中心自由校园巷。	
User	Ok. I'll go there. I need to book a taxi from Alexander Bed and Breakfast to Eraina after 07:00. 好。那我去那里。我还要预订一辆7:00时从亚历山大住宿加早餐旅馆到伊莱娜的出租车。	taxi: {leaveAt: 07:00, destination: eraina, departure: alexander bed and breakfast} 出租车: {出发时间: 07:00, 目的地: 伊莱娜, 出发地: 亚历山大住宿加早餐旅馆}
System	Well. I find a yellow Skoda. 好的。是一辆黄色的斯柯达。	
User	How about its phone number? 它的电话号码是多少？	No update
System	It is 78519675253. 78519675253。	
User	Thank you for your help. Bye! 谢谢你帮忙。再见！	No update
System	A pleasure. Bye bye! 我很乐意。再见！	

Reinforcement Learning from Human Feedback (RLHF)

ChatGPT: Optimizing Language Models for Dialogue

Step 1

Collect demonstration data and train a supervised policy.

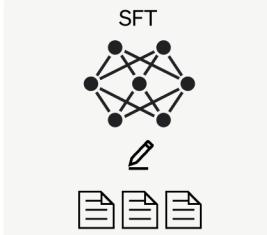
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



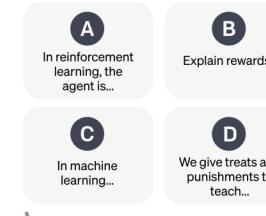
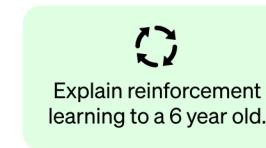
This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

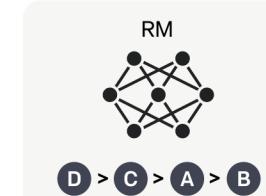
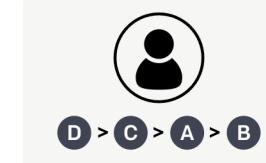
Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



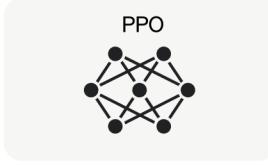
Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

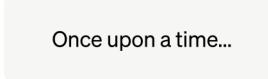
A new prompt is sampled from the dataset.



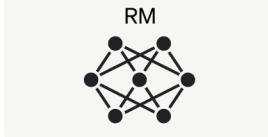
The PPO model is initialized from the supervised policy.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



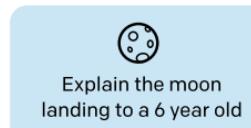
Training language models to follow instructions with human feedback

InstructGPT and GPT 3.5

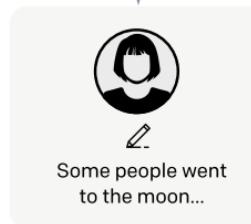
Step 1

Collect demonstration data, and train a supervised policy.

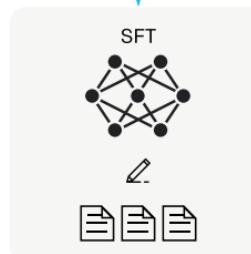
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



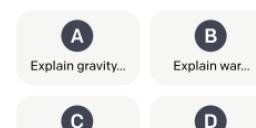
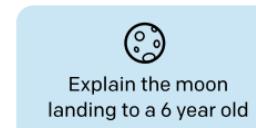
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

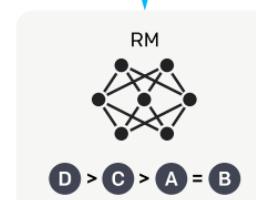
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



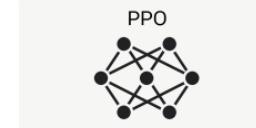
Step 3

Optimize a policy against the reward model using reinforcement learning.

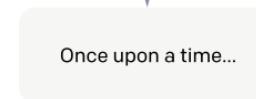
A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



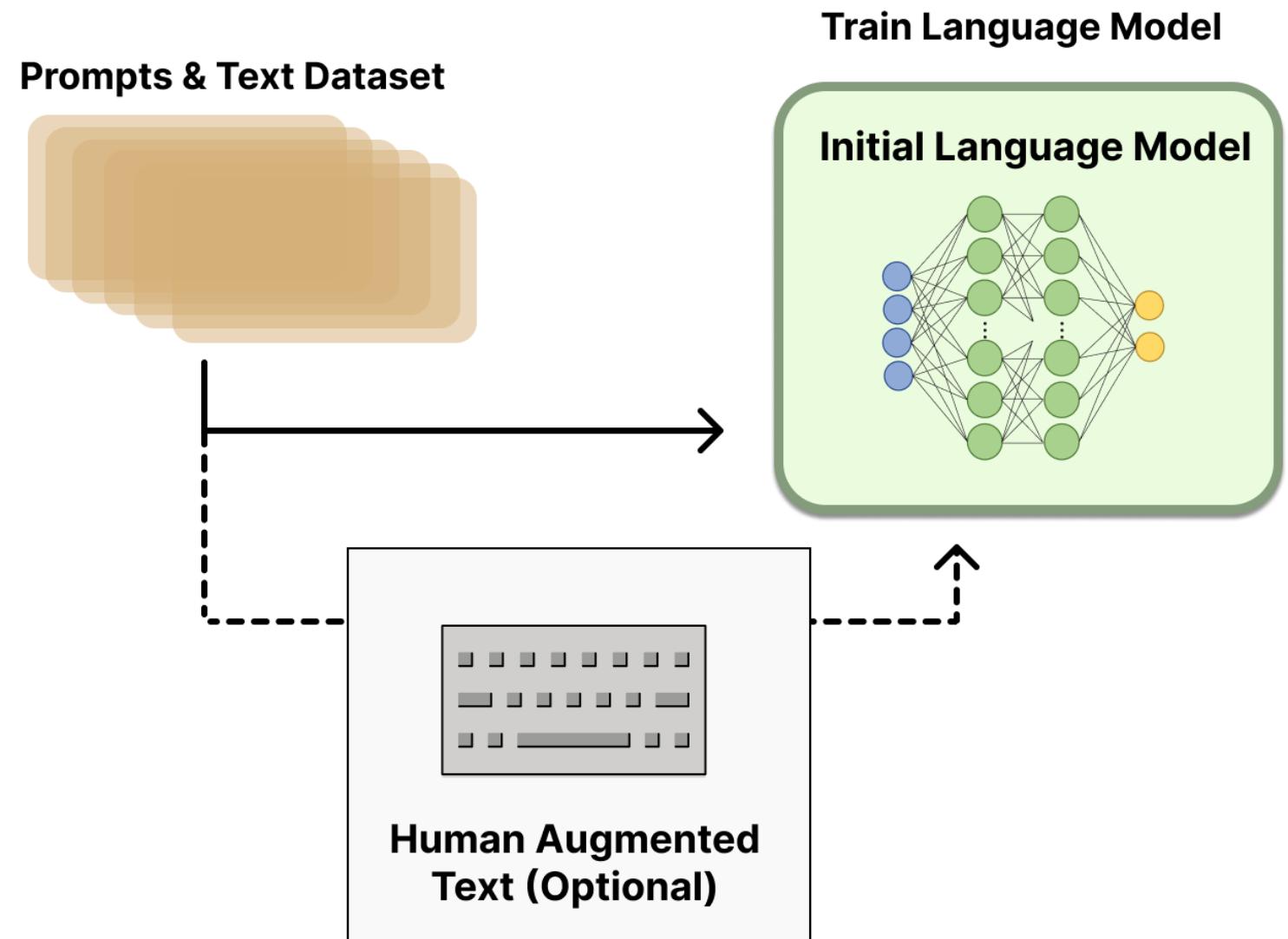
r_k

Reinforcement Learning from Human Feedback (RLHF)

- 1. Pretraining a Language Model (LM)**
- 2. Gathering Data and Training a Reward Model**
- 3. Fine-tuning the LM with Reinforcement Learning**

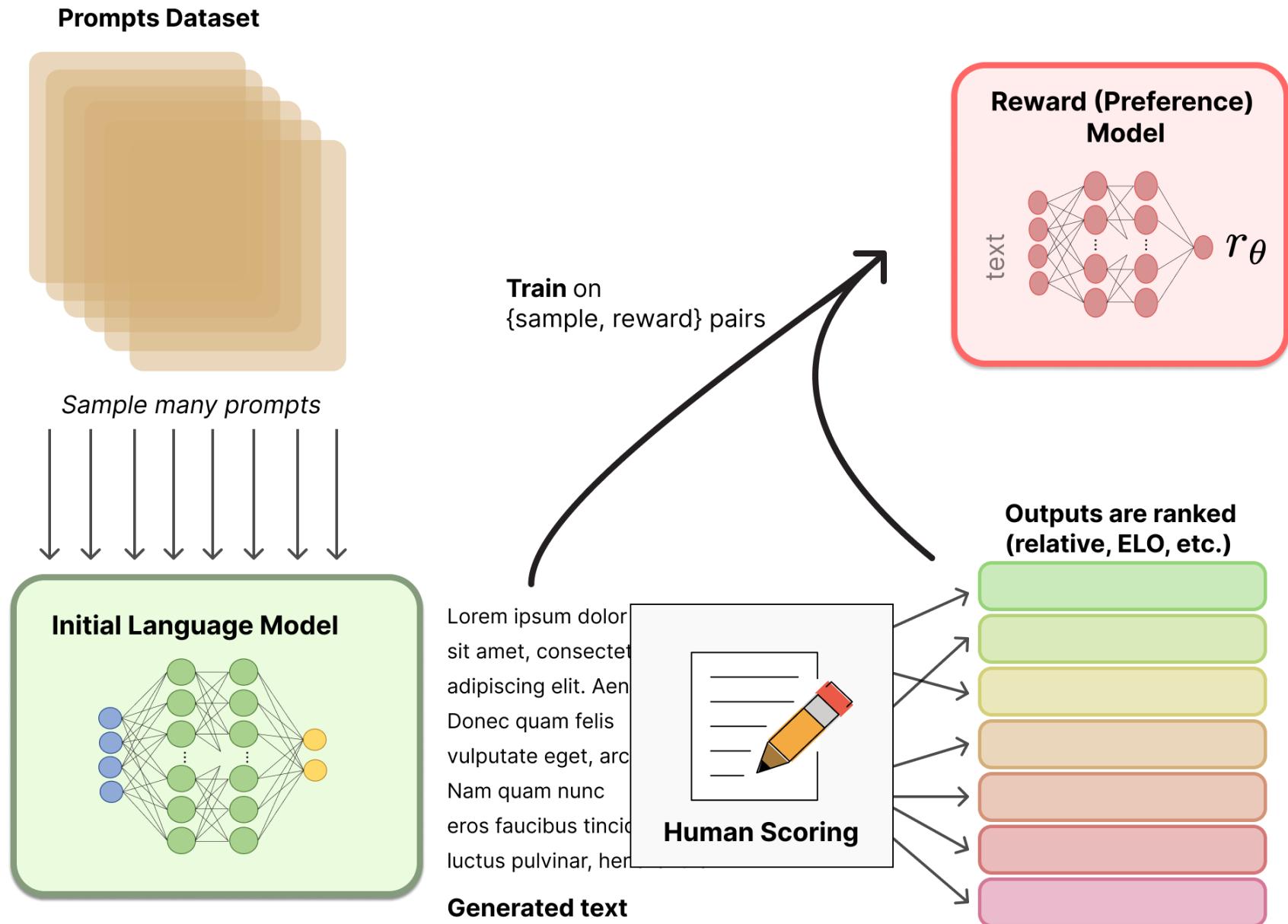
Reinforcement Learning from Human Feedback (RLHF)

**Step 1. Pretraining
a Language Model
(LM)**



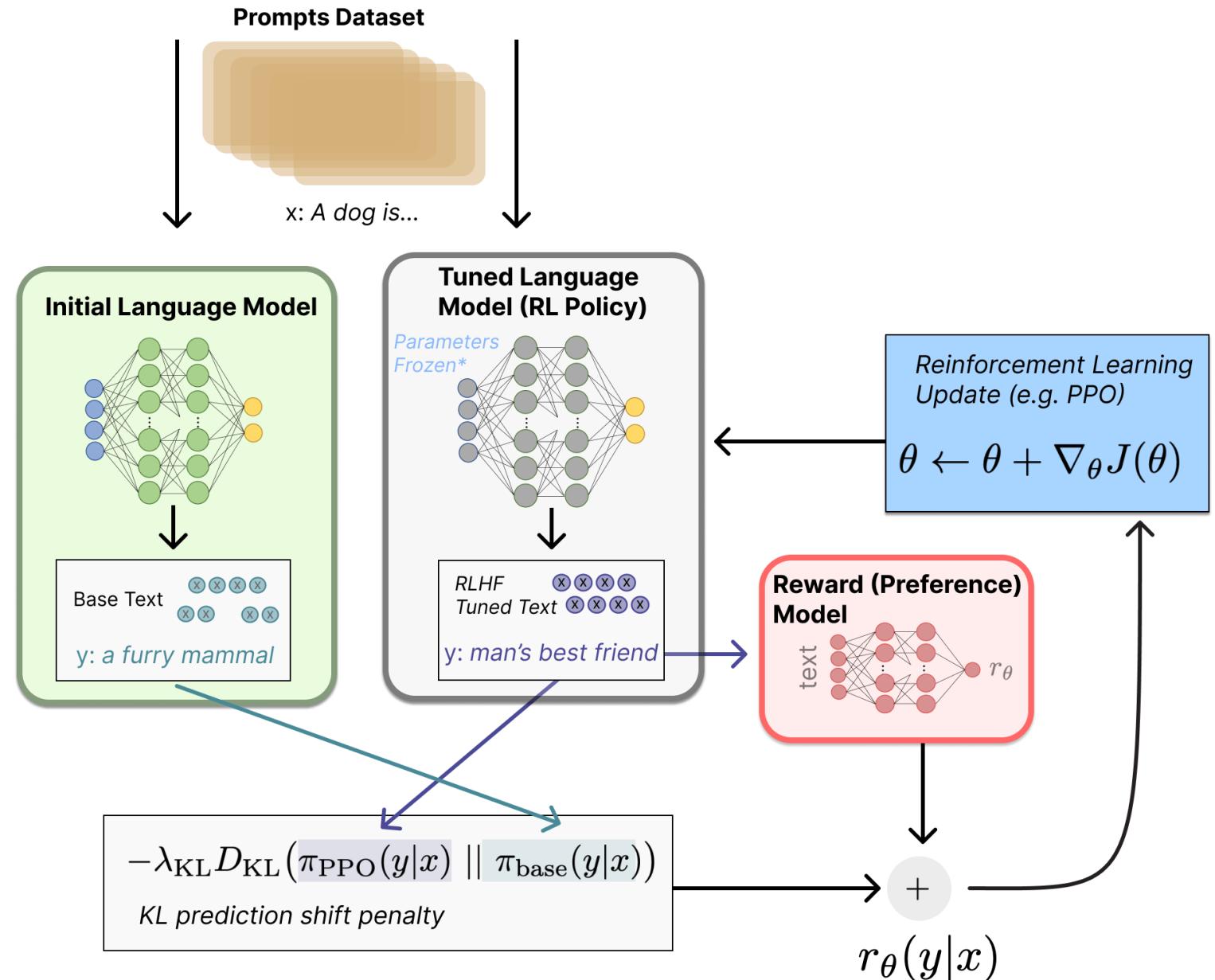
Reinforcement Learning from Human Feedback (RLHF)

Step 2. Gathering Data and Training a Reward Model

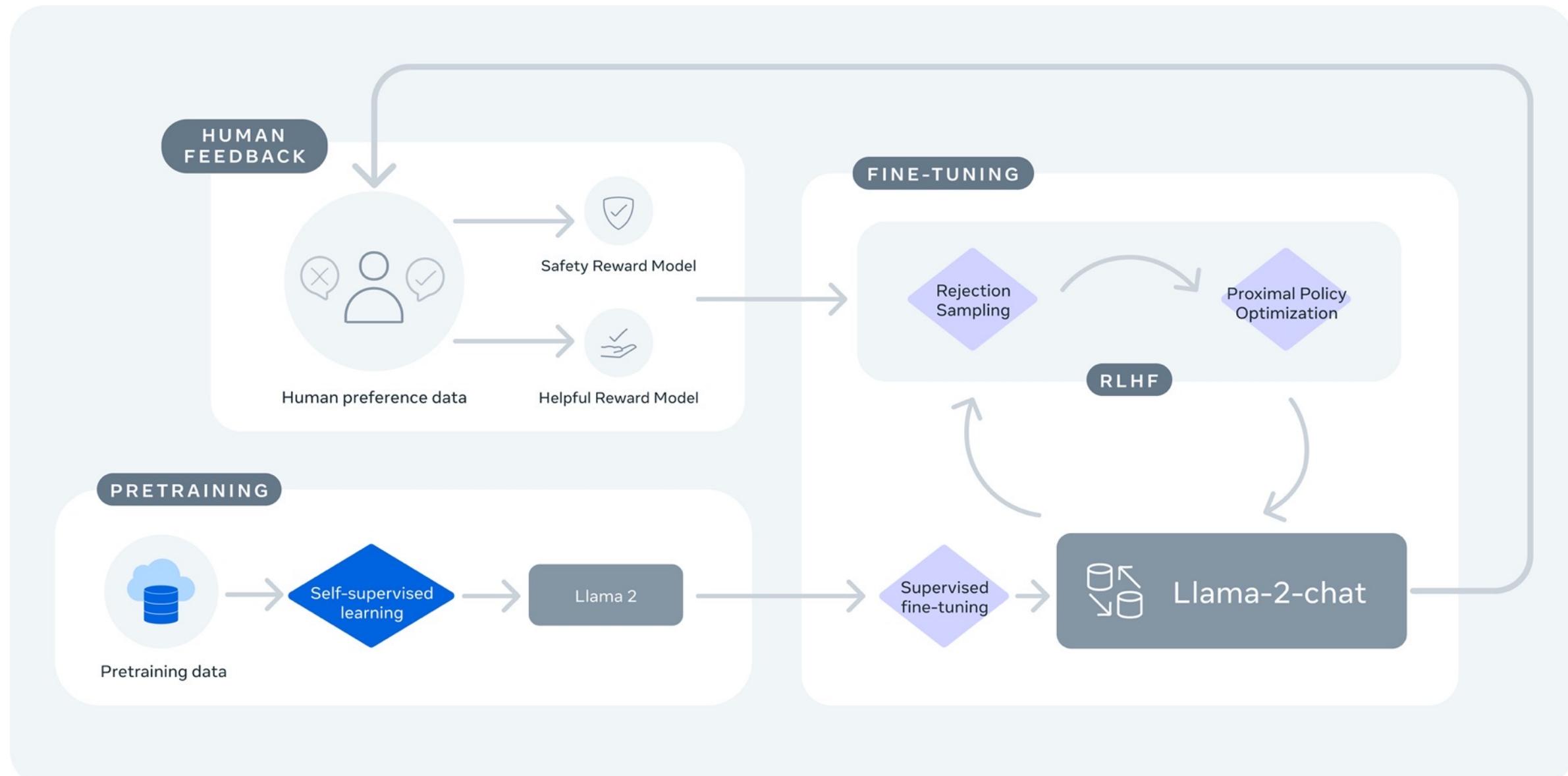


Reinforcement Learning from Human Feedback (RLHF)

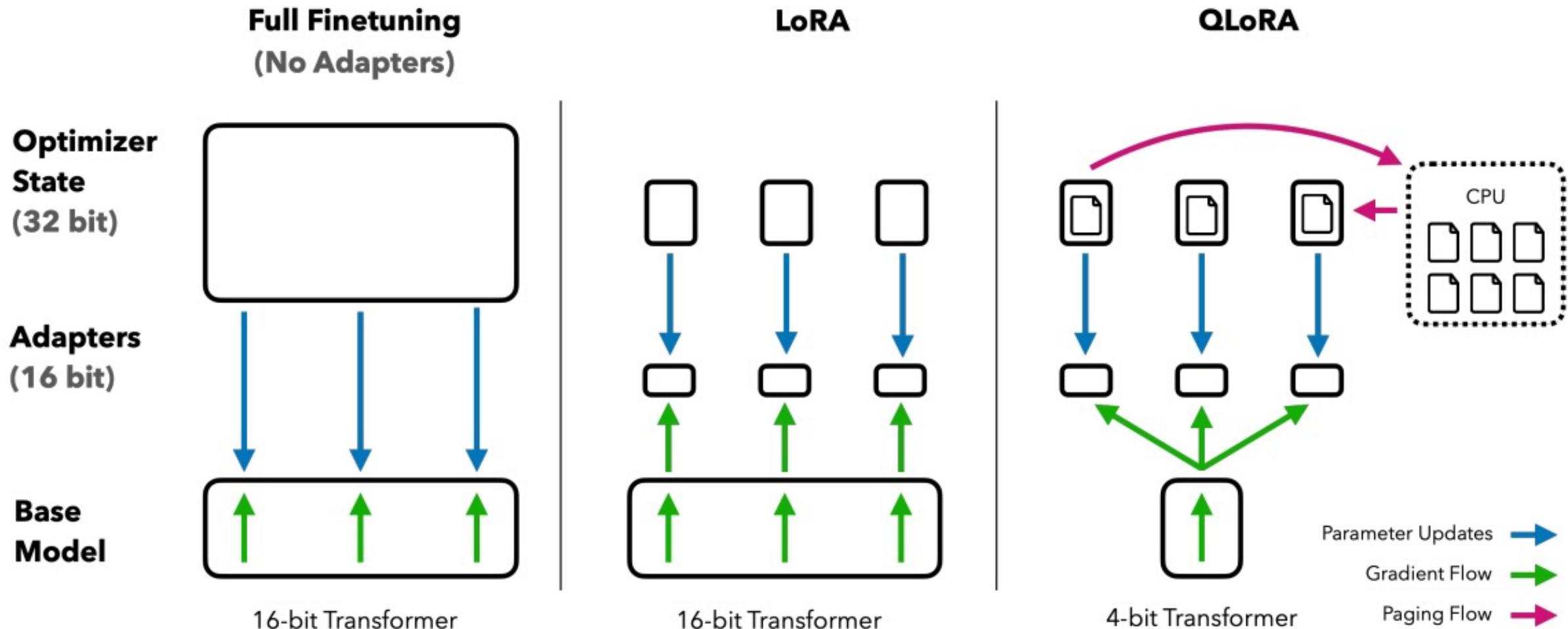
Step 3. Fine-tuning the LM with Reinforcement Learning



Llama-2-chat uses RLHF to ensure safety and helpfulness



QLoRA: Efficient Finetuning of Quantized LLMs



QLoRA: Efficient Finetuning of Quantized LLMs

QLoRA reduces the average memory requirements of finetuning a **65B** parameter model from >780GB of **GPU memory** to **<48GB**

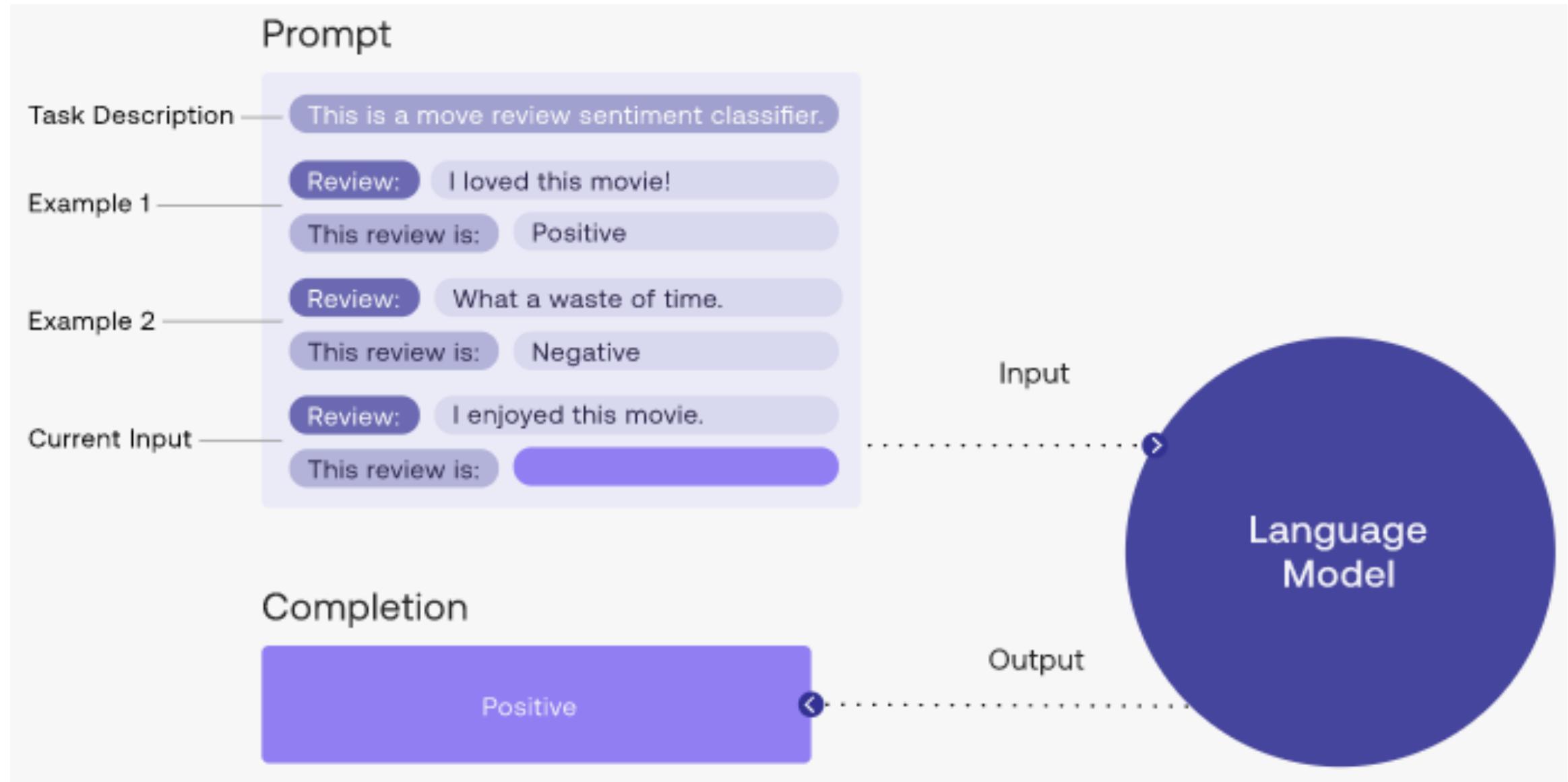
Model	Size	Elo
GPT-4	-	1348 ± 1
Guanaco 65B	41 GB	1022 ± 1
Guanaco 33B	21 GB	992 ± 1
Vicuna 13B	26 GB	974 ± 1
ChatGPT	-	966 ± 1
Guanaco 13B	10 GB	916 ± 1
Bard	-	902 ± 1
Guanaco 7B	6 GB	879 ± 1

QLoRA: Efficient Finetuning of Quantized LLMs

Model / Dataset	Params	Model bits	Memory	ChatGPT vs Sys	Sys vs ChatGPT	Mean	95% CI
GPT-4	-	-	-	119.4%	110.1%	114.5%	2.6%
Bard	-	-	-	93.2%	96.4%	94.8%	4.1%
Guanaco	65B	4-bit	41 GB	96.7%	101.9%	99.3%	4.4%
Alpaca	65B	4-bit	41 GB	63.0%	77.9%	70.7%	4.3%
FLAN v2	65B	4-bit	41 GB	37.0%	59.6%	48.4%	4.6%
Guanaco	33B	4-bit	21 GB	96.5%	99.2%	97.8%	4.4%
Open Assistant	33B	16-bit	66 GB	91.2%	98.7%	94.9%	4.5%
Alpaca	33B	4-bit	21 GB	67.2%	79.7%	73.6%	4.2%
FLAN v2	33B	4-bit	21 GB	26.3%	49.7%	38.0%	3.9%
Vicuna	13B	16-bit	26 GB	91.2%	98.7%	94.9%	4.5%
Guanaco	13B	4-bit	10 GB	87.3%	93.4%	90.4%	5.2%
Alpaca	13B	4-bit	10 GB	63.8%	76.7%	69.4%	4.2%
HH-RLHF	13B	4-bit	10 GB	55.5%	69.1%	62.5%	4.7%
Unnatural Instr.	13B	4-bit	10 GB	50.6%	69.8%	60.5%	4.2%
Chip2	13B	4-bit	10 GB	49.2%	69.3%	59.5%	4.7%
Longform	13B	4-bit	10 GB	44.9%	62.0%	53.6%	5.2%
Self-Instruct	13B	4-bit	10 GB	38.0%	60.5%	49.1%	4.6%
FLAN v2	13B	4-bit	10 GB	32.4%	61.2%	47.0%	3.6%
Guanaco	7B	4-bit	5 GB	84.1%	89.8%	87.0%	5.4%
Alpaca	7B	4-bit	5 GB	57.3%	71.2%	64.4%	5.0%
FLAN v2	7B	4-bit	5 GB	33.3%	56.1%	44.8%	4.0%

Source: Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. "QLoRA: Efficient Finetuning of Quantized LLMs." arXiv preprint arXiv:2305.14314 (2023).

Prompt Engineering with ChatGPT for NLP



Hugging Face Tasks

Natural Language Processing



Text Classification

3345 models



Token Classification

1492 models



Question Answering

1140 models



Translation

1467 models



Summarization

323 models



Text Generation

3959 models



Fill-Mask

2453 models



Sentence Similarity

352 models

NLP with Transformers Github

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search / Sign in Sign up

[nlp-with-transformers / notebooks](#) Public

Notifications Fork 170 Star 1.1k

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Code

lewtn Merge pull request #21 from JingchaoZhang/patch-3 ... ae5b7c1 15 days ago 71 commits

.github/ISSUE_TEMPLATE Update issue templates 25 days ago

data Move dataset to data directory 4 months ago

images Add README last month

scripts Update issue templates 25 days ago

.gitignore Initial commit 4 months ago

01_introduction.ipynb Remove Colab badges & fastdoc refs 27 days ago

02_classification.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

03_transformer-anatomy.ipynb [Transformers Anatomy] Remove cells with figure references 22 days ago

04_multilingual-ner.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

05_text-generation.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

About

Jupyter notebooks for the Natural Language Processing with Transformers book

[transformersbook.com/](#)

Readme Apache-2.0 License 1.1k stars 33 watching 170 forks

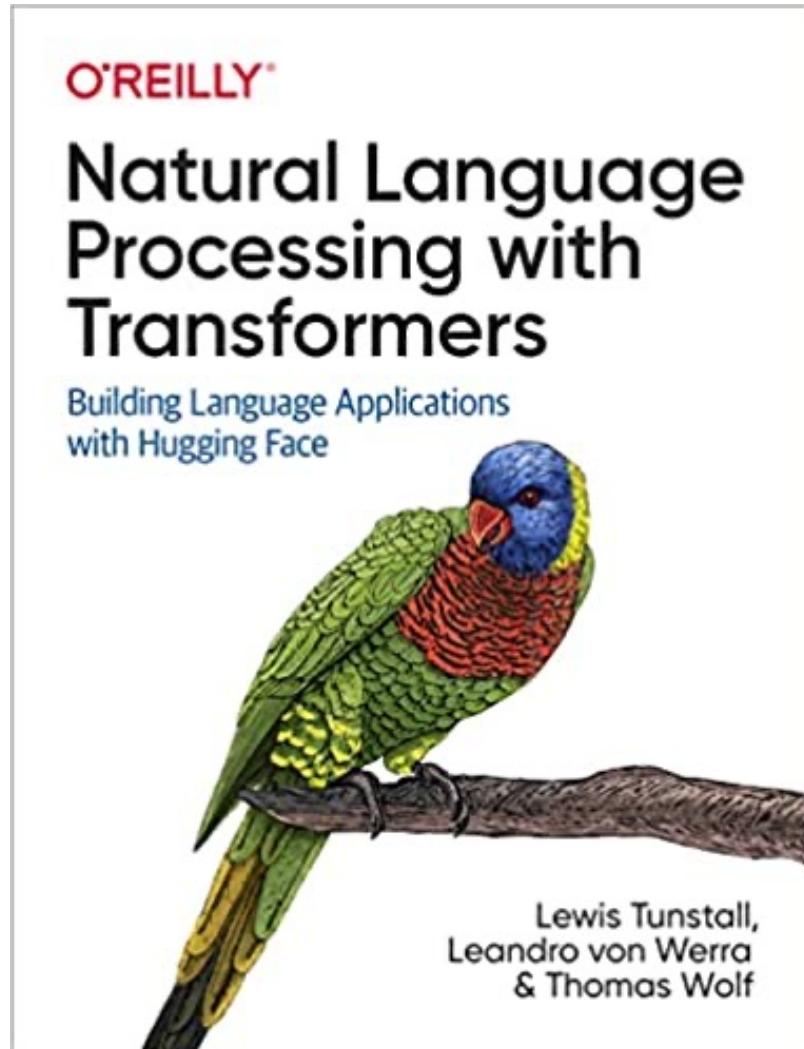
O'REILLY® Natural Language Processing with Transformers Building Language Applications with Hugging Face

Releases No releases published

Packages

<https://github.com/nlp-with-transformers/notebooks>

NLP with Transformers Github Notebooks



Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity Recognition	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Generation	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Summarization	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Question Answering	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Making Transformers Efficient in Production	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Dealing with Few to No Labels	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Training Transformers from Scratch	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Future Directions	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using [Kaggle](#), [Gradient](#), or [SageMaker Studio Lab](#). These platforms tend to provide more performant GPUs like P100s, all for free!

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title bar reads "python101.ipynb" and "All changes saved". The main content area has a red header "NLP with Transformers". On the left, there's a sidebar with a "Table of contents" section containing various topics like "Natural Language Processing with Transformers", "Text Classification", etc. The main content area has two sections: "Natural Language Processing with Transformers" and "Text Classification". Under "Natural Language Processing with Transformers", there are code cells [1] through [12]. Cell [1] contains a git clone command. Cells [3] and [12] contain text samples. Under "Text Classification", there are code cells [13] and [14]. Cell [13] imports the transformers pipeline, and cell [14] uses it to classify text from cell [12]. The top right of the screen shows "Comment", "Share", "Settings", and a profile icon.

NLP with Transformers

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Github: <https://github.com/nlp-with-transformers/notebooks>

```
[1]: 1 !git clone https://github.com/nlp-with-transformers/notebooks.git
2 %cd notebooks
3 from install import *
4 install_requirements()

[3]: 1 from utils import *
2 setup_chapter()

[12]: 1 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
2 from your online store in Germany. Unfortunately, when I opened the package, \
3 I discovered to my horror that I had been sent an action figure of Megatron \
4 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
5 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
6 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
7 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
"""

[13]: 1 from transformers import pipeline
2 classifier = pipeline("text-classification")

[14]: 1 import pandas as pd
2 outputs = classifier(text)
3 pd.DataFrame(outputs)
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook titled "Text Classification". The left sidebar contains a "Table of contents" with several sections under "Text Classification with Transformers", such as "The Dataset", "From Datasets to DataFrames", "From Text to Tokens", etc. The main area displays code cells and their outputs. Cell [10] shows a command to check GPU usage: !nvidia-smi. Cell [11] contains code to clone a GitHub repository and install requirements: # Uncomment and run this cell if you're on Colab or Kaggle, !git clone https://github.com/nlp-with-transformers/notebooks.git, %cd notebooks, from install import *, install_requirements(). Cell [12] hides imports: # hide, from utils import *, setup_chapter(). Cell [13] lists available datasets: from datasets import list_datasets, all_datasets = list_datasets(), print(f"There are {len(all_datasets)} datasets currently available on the Hub"), print(f"The first 10 are: {all_datasets[:10]}"). The output of this cell shows there are 3783 datasets and the first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa', ...]. The top right corner shows "All changes saved" and various status indicators like RAM and Disk usage.

Text Classification

Table of contents

- Text Classification with Transformers
 - The Dataset
 - From Datasets to DataFrames
 - From Text to Tokens
 - Character Tokenization
 - Word Tokenization
 - Subword Tokenization
 - Tokenizing the Whole Dataset
 - Training a Text Classifier
 - Transformers as Feature Extractors
 - Extracting the last hidden states
 - Creating a feature matrix
 - Visualizing the training set
 - Training a simple classifier
 - Fine-Tuning Transformers
 - Loading a pretrained model
 - Defining the performance metrics
 - Training the model
 - Sidebar: Fine-Tuning with Keras
 - Error analysis
 - Saving and sharing the model

Text Classification with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

[10] !nvidia-smi

[11] `# Uncomment and run this cell if you're on Colab or Kaggle`
!git clone <https://github.com/nlp-with-transformers/notebooks.git>
%cd notebooks
from install import *
install_requirements()

[12] `# hide`
from utils import *
setup_chapter()

The Dataset

[13] from datasets import list_datasets
all_datasets = list_datasets()
print(f"There are {len(all_datasets)} datasets currently available on the Hub")
print(f"The first 10 are: {all_datasets[:10]}")

There are 3783 datasets currently available on the Hub
The first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa', ...]

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

python101.ipynb

Named Entity Recognition (NER)

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing A

Multilingual Named Entity Recognition (NER)

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

```
[ ] 1 #NER: https://huggingface.co/tasks/token-classification
2 !pip install transformers
3 from transformers import pipeline
4 classifier = pipeline("ner")
5 classifier("Hello I'm Omar and I live in Zürich.")

▶ 1 from transformers import pipeline
2 classifier = pipeline("ner")
3 classifier("Hello I'm Omar and I live in Zürich.")

[{"end": 14,
 'entity': 'I-PER',
 'index': 5,
 'score': 0.99770516,
 'start': 10,
 'word': 'Omar'},
 {'end': 35,
 'entity': 'I-LOC',
 'index': 10,
 'score': 0.9968976,
 'start': 29,
 'word': 'Zürich'}]
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title bar reads "python101.ipynb" and "Text Summarization". The main content area has a heading "Text Summarization" and a list of sources:

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

Below the sources, there are two code cells. The first cell contains Python code for summarizing a text about Paris:

```
1 #Source: https://huggingface.co/tasks/summarization
2 !pip install transformers
3 from transformers import pipeline
4 classifier = pipeline("summarization")
5 text = "Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents as of 2018, in an area of more than
6 classifier(text, max_length=30)
```

The output of this cell shows the summarized text:

```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 (https://huggingface.co/sshleifer/distilbart-cnn-12-6)
Your min_length=56 must be inferior than your max_length=30.
[{'summary_text': 'Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents . The City of Paris'}]
```

The second cell contains Python code for summarizing a longer text about a package exchange:

```
1 #!pip install transformers
2 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
3 from your online store in Germany. Unfortunately, when I opened the package, \
4 I discovered to my horror that I had been sent an action figure of Megatron \
5 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
6 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
7 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
8 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
9 from transformers import pipeline
10 summarizer = pipeline("summarization")
11 outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
12 print(outputs[0]['summary_text'])
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook titled "Text Generation". The notebook interface includes a top bar with file navigation, a sidebar with search and file management, and a main workspace with code cells and output.

Title: Text Generation

Code Cells:

- [9]

```
1 #Source: https://huggingface.co/tasks/text-generation
2 #!pip install transformers
3 from transformers import pipeline
4 generator = pipeline('text-generation', model = 'gpt2')
5 generator("Hello, I'm a language model", max_length = 30, num_return_sequences=3)
```

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': "Hello, I'm a language model.\n\nBut then, one day, I'm not trying to teach the language in my head.\n\n"}, {'generated_text': "Hello, I'm a language model. I'm an implementation for the type system. I'm working with types and programming language constructs. I am a language modeler, not a programmer. As you know, languages are not a linear model. The thing that jumps out at"}, {'generated_text': "Hello, I'm a language modeler, not a programmer. As you know, languages are not a linear model. The thing that jumps out at"}]
- [18s]

```
1 from transformers import pipeline
2 generator = pipeline('text-generation', model = 'gpt2')
3 outputs = generator("Once upon a time", max_length = 30, num_return_sequences=3)
4 print(outputs[0]['generated_text'])
```

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Once upon a time, every person who ever saw Jesus, knew that He was Christ. And even though he might not have known Him, He was
- [1]

```
1 from transformers import pipeline
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook titled "python101.ipynb". The main title of the notebook is "Question Answering and Dialogue Systems". The notebook structure includes a section titled "Question Answering and Dialogue Systems" which contains two bullet points:

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

Below this, there is a section titled "Question Answering" containing a code cell. The code cell contains the following Python code:

```
[3] 1 !pip install transformers
2 from transformers import pipeline
3 qamodel = pipeline("question-answering")
4 question = "Where do I live?"
5 context = "My name is Michael and I live in Taipei."
6 qamodel(question = question, context = context)
```

The code cell has been run, and the output is displayed below it:

```
1 from transformers import pipeline
2 qamodel = pipeline("question-answering")
3 question = "Where do I live?"
4 context = "My name is Michael and I live in Taipei."
5 qamodel(question = question, context = context)
6 #{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

A note at the bottom of the output states: "No model was supplied, defaulted to distilbert-base-cased-distilled-squad (<https://huggingface.co/distilbert-base-cased-distilled-squad>)".

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows the Google Colab interface with three code cells. The first cell (line 12) asks about precipitation causes and receives the answer 'gravity'. The second cell (line 13) asks about other forms of precipitation and receives the answer 'graupel'. The third cell (line 14) asks where water droplets collide with ice crystals to form precipitation and receives the answer 'within a cloud'. The interface includes a toolbar with file operations, a sidebar with a file tree, and a status bar at the bottom.

```
[12]: 1 from transformers import pipeline
2 qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "What causes precipitation to fall?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

```
[13]: 1 from transformers import pipeline
2 qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

```
[14]: 1 #from transformers import pipeline
2 #qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "Where do water droplets collide with ice crystals to form precipitation?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab interface with the following details:

- Title:** python101.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Header Buttons:** Comment, Share, Settings, Font Size.
- Table of Contents (Left Sidebar):**
 - Semantic Analysis
 - Named Entity Recognition (NER)
 - NER with CRF
 - NER with CRF RandomizedSearchCV
 - Sentiment Analysis
 - Sentiment Analysis - Unsupervised Lexical
 - Sentiment Analysis - Supervised Machine Learning
 - Sentiment Analysis - Supervised Deep Learning Models
 - Sentiment Analysis - Advanced Deep Learning
 - Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
 - Question Answering and Dialogue Systems** (highlighted in yellow)
 - Question Answering (QA)
 - BERT for Question Answering
 - Dialogue Systems
 - Joint Intent Classification and Slot Filling with Transformers
 - Data Visualization**
 - Code/Text Buttons:** + Code, + Text
 - Resource Buttons:** RAM, Disk, Editing

Main Content Area:

Question Answering and Dialogue Systems

Question Answering (QA)

BERT for Question Answering

Source: Apoorv Nandan (2020), BERT (from HuggingFace Transformers) for Text Extraction, https://keras.io/examples/nlp/text_extraction_with_bert/

Description: Fine tune pretrained BERT from HuggingFace Transformers on SQuAD.

Introduction

This demonstration uses SQuAD (Stanford Question-Answering Dataset). In SQuAD, an input consists of a question, and a paragraph for context. The goal is to find the span of text in the paragraph that answers the question. We evaluate our performance on this data with the "Exact Match" metric, which measures the percentage of predictions that exactly match any one of the ground-truth answers.

We fine-tune a BERT model to perform this task as follows:

 1. Feed the context and the question as inputs to BERT.
 2. Take two vectors S and T with dimensions equal to that of hidden states in BERT.
 3. Compute the probability of each token being the start and end of the answer span. The probability of a token being the start of the answer is given by a dot product between S and the representation of the token in the last layer of BERT, followed by a softmax over all tokens. The probability of a token being the end of the answer is computed similarly with the vector T.
 4. Fine-tune BERT and learn S and T along the way.

References:

 - [BERT](#)
 - [SQuAD](#)

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The left sidebar contains a 'Table of contents' with sections like RandomizedSearchCV, Sentiment Analysis, Deep Learning and Universal Sentence-Embedding Models, Question Answering and Dialogue Systems, and Data Visualization. A specific section titled 'BERT for Question Answering' is highlighted. The main workspace displays code execution results, including progress bars for file downloads and a detailed table of model layers and their properties.

Table of contents

- RandomizedSearchCV
- Sentiment Analysis
 - Sentiment Analysis - Unsupervised
 - Lexical
 - Sentiment Analysis - Supervised
 - Machine Learning
 - Sentiment Analysis - Supervised
 - Deep Learning Models
 - Sentiment Analysis - Advanced Deep
 - Learning
- Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
- Question Answering and Dialogue Systems
 - Question Answering (QA)
 - BERT for Question Answering**
 - Dialogue Systems
 - Joint Intent Classification and Slot Filling with Transformers
- Data Visualization

+ Code + Text

Downloading: 100% 433/433 [00:29<00:00, 14.5B/s]

Downloading: 100% 536M/536M [00:29<00:00, 18.3MB/s]

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[None, 384]	0	
input_3 (InputLayer)	[None, 384]	0	
input_2 (InputLayer)	[None, 384]	0	
tf_bert_model (TFBertModel)	((None, 384, 768), (109482240	input_1[0][0]	
start_logit (Dense)	(None, 384, 1)	768	tf_bert_model[0][0]
end_logit (Dense)	(None, 384, 1)	768	tf_bert_model[0][0]
flatten (Flatten)	(None, 384)	0	start_logit[0][0]
flatten_1 (Flatten)	(None, 384)	0	end_logit[0][0]
activation_7 (Activation)	(None, 384)	0	flatten[0][0]
activation_8 (Activation)	(None, 384)	0	flatten_1[0][0]

Total params: 109,483,776
Trainable params: 109,483,776
Non-trainable params: 0

CPU times: user 20.8 s, sys: 7.75 s, total: 28.5 s
Wall time: 1min 42s

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title bar indicates the file is named "python101.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a status message "All changes saved". The top right features a Comment button, a Share button, a settings gear, and a font size A icon.

The left sidebar contains a "Table of contents" panel with the following structure:

- RandomizedSearchCV
- Sentiment Analysis
 - Sentiment Analysis - Unsupervised
 - Lexical
 - Sentiment Analysis - Supervised
 - Machine Learning
 - Sentiment Analysis - Supervised
 - Deep Learning Models
 - Sentiment Analysis - Advanced Deep
 - Learning
- Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
- Question Answering and Dialogue Systems
 - Question Answering (QA)
 - BERT for Question Answering
 - Dialogue Systems
 - Joint Intent Classification and Slot Filling with Transformers** (highlighted with a yellow bar)
- Data Visualization

The main content area has a header "Dialogue Systems" in red. Below it, a code cell shows two lines of Python code:

```
[ ] 1 #Source: Olivier Grisel (2020), Transformers (BERT fine-tuning): Joint Intent Classification and S
2 #https://github.com/m2dsupdlclass/lectures-labs/blob/master/labs/06_deep_nlp/Transformers_Joint_I
```

A collapsible section titled "Joint Intent Classification and Slot Filling with Transformers" is expanded, containing the following text:

The goal of this notebook is to fine-tune a pretrained transformer-based neural network model to convert a user query expressed in English into a representation that is structured enough to be processed by an automated service.

Here is an example of interpretation computed by such a Natural Language Understanding system:

```
>>> nlu("Book a table for two at Le Ritz for Friday night",
        tokenizer, joint_model, intent_names, slot_names)
```

```
{
  'intent': 'BookRestaurant',
  'slots': {
    'party_size_number': 'two',
    'restaurant_name': 'Le Ritz',
    'timeRange': 'Friday night'
  }
}
```

Intent classification is a simple sequence classification problem. The trick is to treat the structured knowledge extraction part ("Slot Filling") as token-level classification problem using BIO-annotations:

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows the Google Colab interface with a notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various sections such as RandomizedSearchCV, Sentiment Analysis, Sentiment Analysis - Unsupervised Lexical, Sentiment Analysis - Supervised Machine Learning, Sentiment Analysis - Supervised Deep Learning Models, Sentiment Analysis - Advanced Deep Learning, Deep Learning and Universal Sentence-Embedding Models, Universal Sentence Encoder (USE), Universal Sentence Encoder Multilingual (USEM), Question Answering and Dialogue Systems, Question Answering (QA), BERT for Question Answering, Dialogue Systems, and Joint Intent Classification and Slot Filling with Transformers. The main workspace shows a code cell with the following Python code:

```
1 def show_predictions(text, tokenizer, model, intent_names, slot_names):
2     inputs = tf.constant(tokenizer.encode(text))[None, :] # batch_size = 1
3     outputs = model(inputs)
4     slot_logits, intent_logits = outputs
5     slot_ids = slot_logits.numpy().argmax(axis=-1)[0, 1:-1]
6     intent_id = intent_logits.numpy().argmax(axis=-1)[0]
7     print("Text:", text)
8     print("Intent:", intent_names[intent_id])
9     print("Slots:")
10    for token, slot_id in zip(tokenizer.tokenize(text), slot_ids):
11        print(f"{token}>10} : {slot_names[slot_id]}")
12
13 show_predictions("Book a table for two at Le Ritz for Friday night!",
14                   tokenizer, joint_model, intent_names, slot_names)
```

The output of the code cell is displayed below the code:

```
Text: Book a table for two at Le Ritz for Friday night!
Intent: BookRestaurant
Slots:
    Book : O
        a : O
    table : O
        for : O
    two : B-party_size_number
        at : O
    Le : B-restaurant_name
        R : I-restaurant_name
    #itz : I-restaurant_name
        for : O
    Friday : B-timeRange
        night : O
        ! : O
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows the Google Colab interface with a notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various sections such as NER with CRF, Sentiment Analysis, Deep Learning and Universal Sentence-Embedding Models, Question Answering and Dialogue Systems, and Joint Intent Classification and Slot Filling with Transformers. The main workspace displays a block of Python code related to slot filling and intent classification.

```
# NAIVE BIO: handling: treat B- and I- the same...
new_slot_name = current_word_slot_name[2:]
if active_slot_name is None:
    active_slot_words.append(word)
    active_slot_name = new_slot_name
elif new_slot_name == active_slot_name:
    active_slot_words.append(word)
else:
    collected_slots[active_slot_name] = " ".join(active_slot_words)
    active_slot_words = [word]
    active_slot_name = new_slot_name
if active_slot_name:
    collected_slots[active_slot_name] = " ".join(active_slot_words)
info["slots"] = collected_slots
return info

def nlu(text, tokenizer, model, intent_names, slot_names):
    inputs = tf.constant(tokenizer.encode(text))[None, :] # batch_size = 1
    outputs = model(inputs)
    slot_logits, intent_logits = outputs
    slot_ids = slot_logits.numpy().argmax(axis=-1)[0, 1:-1]
    intent_id = intent_logits.numpy().argmax(axis=-1)[0]

    return decode_predictions(text, tokenizer, intent_names, slot_names,
                             intent_id, slot_ids)

nlu("Book a table for two at Le Ritz for Friday night",
    tokenizer, joint_model, intent_names, slot_names)

{'intent': 'BookRestaurant',
 'slots': {'party_size_number': 'two',
           'restaurant_name': 'Le Ritz',
           'timeRange': 'Friday night'}}}
```

<https://tinyurl.com/aintpuppython101>

NLP with Transformers

```
!git clone https://github.com/nlp-with-transformers/notebooks.git  
%cd notebooks  
from install import *  
install_requirements()
```

```
from utils import *  
setup_chapter()
```

Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
classifier = pipeline("text-classification")

import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Text Classification

```
from transformers import pipeline
classifier = pipeline("text-classification")

import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Named Entity Recognition

```
ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
pd.DataFrame(outputs)
```

	entity_group	score	word	start	end
0	ORG	0.879010	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556570	Mega	208	212
4	PER	0.590256	##tron	212	216
5	ORG	0.669692	Decept	253	259
6	MISC	0.498349	##icons	259	264
7	MISC	0.775362	Megatron	350	358
8	MISC	0.987854	Optimus Prime	367	380
9	PER	0.812096	Bumblebee	502	511

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

<https://github.com/nlp-with-transformers/notebooks>

Question Answering

```
reader = pipeline("question-answering")
question = "What does the customer want?"
outputs = reader(question=question, context=text)
pd.DataFrame([outputs])
```

	score	start	end	answer
0	0.631292	335	358	an exchange of Megatron

Summarization

```
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Text Summarization

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Translation

```
translator = pipeline("translation_en_to_de",
                     model="Helsinki-NLP/opus-mt-en-de")
outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)
print(outputs[0]['translation_text'])
```

Sehr geehrter Amazon, letzte Woche habe ich eine Optimus Prime Action Figur aus Ihrem Online-Shop in Deutschland bestellt. Leider, als ich das Paket öffnete, entdeckte ich zu meinem Entsetzen, dass ich stattdessen eine Action Figur von Megatron geschickt worden war! Als lebenslanger Feind der Decepticons, Ich hoffe, Sie können mein Dilemma verstehen. Um das Problem zu lösen, Ich fordere einen Austausch von Megatron für die Optimus Prime Figur habe ich bestellt. Anbei sind Kopien meiner Aufzeichnungen über diesen Kauf. Ich erwarte, bald von Ihnen zu hören. Aufrichtig, Bumblebee.

Text Generation

```
from transformers import set_seed  
set_seed(42) # Set the seed to get reproducible results  
  
generator = pipeline("text-generation")  
response = "Dear Bumblebee, I am sorry to hear that your order was mixed up."  
prompt = text + "\n\nCustomer service response:\n" + response  
outputs = generator(prompt, max_length=200)  
print(outputs[0]['generated_text'])
```

Customer service response:

Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

Text Generation

Dear Amazon, last week I ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead! As a lifelong enemy of the Decepticons, I hope you can understand my dilemma. To resolve the issue, I demand an exchange of Megatron for the Optimus Prime figure I ordered. Enclosed are copies of my records concerning this purchase. I expect to hear from you soon. Sincerely, Bumblebee.

Customer service response:

Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

Question Answering

```
!pip install transformers
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)
```

```
{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

Question Answering

```
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
output = qamodel(question = question, context = context)
print(output['answer'])
```

Taipei

Text Generation with LLM (zephyr-7b-beta)

```
# Install transformers from source - only needed for versions <= v4.34
!pip install git+https://github.com/huggingface/transformers.git
!pip install accelerate
```

Text Generation with LLM (zephyr-7b-beta)

```
import torch
from transformers import pipeline

pipe = pipeline("text-generation",
model="HuggingFaceH4/zephyr-7b-beta",
torch_dtype=torch.bfloat16,
device_map="auto")
```

Text Generation with LLM (zephyr-7b-beta)

```
# We use the tokenizer's chat template to format each message - see
# https://huggingface.co/docs/transformers/main/en/chat_templating
messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",
    },
    {"role": "user", "content": "How many helicopters can a human eat in one sitting?"},
]
prompt = pipe.tokenizer.apply_chat_template(messages,
tokenize=False, add_generation_prompt=True)

outputs = pipe(prompt, max_new_tokens=256,
do_sample=True, temperature=0.7, top_k=50, top_p=0.95)
print(outputs[0]["generated_text"])
```

Text Generation with LLM (zephyr-7b-beta)

```
import torch
from transformers import pipeline

pipe = pipeline("text-generation", model="HuggingFaceH4/zephyr-7b-beta",
torch_dtype=torch.bfloat16, device_map="auto")

# We use the tokenizer's chat template to format each message - see
# https://huggingface.co/docs/transformers/main/en/chat_templating
messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",
    },
    {"role": "user", "content": "How many helicopters can a human eat in one sitting?"},
]

prompt = pipe.tokenizer.apply_chat_template(messages, tokenize=False,
add_generation_prompt=True)
outputs = pipe(prompt, max_new_tokens=256, do_sample=True, temperature=0.7,
top_k=50, top_p=0.95)
print(outputs[0]["generated_text"])
```

Summary

- Question Answering
- Dialogue Systems
 - Task Oriented Dialogue (TOD) System
 - Task-Oriented Dialogue System (TDS)
 - Conversational Dialogue System (CDS)
 - Question Answering Dialogue System (QADS)

References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yıldırım and Meysam Asgari-Chenaglu (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. arXiv preprint arXiv:2305.18290.
- Tunstall, Lewis, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang et al. "Zephyr: Direct Distillation of LM Alignment." arXiv preprint arXiv:2310.16944 (2023).
- Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun (2023). "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT." arXiv preprint arXiv:2303.04226.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. (2023) "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." ACM Computing Surveys 55, no. 9 (2023): 1-35.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. (2023) "A Survey of Large Language Models." arXiv preprint arXiv:2303.18223.
- Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov et al. (2023) "Llama 2: Open Foundation and Fine-Tuned Chat Models." arXiv preprint arXiv:2307.09288 (2023).
- Junliang Wang, Chuqiao Xu, Jie Zhang, and Ray Zhong (2022). "Big data analytics for intelligent manufacturing systems: A review." Journal of Manufacturing Systems 62 (2022): 738-752.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155.
- Gozalo-Brizuela, Roberto, and Eduardo C. Garrido-Merchan (2023). "ChatGPT is not all you need. A State of the Art Review of large Generative AI models." arXiv preprint arXiv:2301.04655 (2023).
- Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. (2023) "InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning." arXiv preprint arXiv:2305.06500 (2023).
- Shahab Saquib Sohail, Faiza Farhat, Yassine Himeur, Mohammad Nadeem, Dag Øivind Madsen, Yashbir Singh, Shadi Atalla, and Wathiq Mansoor (2023). "The Future of GPT: A Taxonomy of Existing ChatGPT Research, Current Challenges, and Possible Future Directions." Current Challenges, and Possible Future Directions (April 8, 2023) (2023).
- Longbing Cao (2022). "Decentralized ai: Edge intelligence and smart blockchain, metaverse, web3, and desci." IEEE Intelligent Systems 37, no. 3: 6-19.
- Qinglin Yang, Yetong Zhao, Huawei Huang, Zehui Xiong, Jiawen Kang, and Zibin Zheng (2022). "Fusing blockchain and AI with metaverse: A survey." IEEE Open Journal of the Computer Society 3 : 122-136.
- Russell Belk, Mariam Humayun, and Myriam Brouard (2022). "Money, possessions, and ownership in the Metaverse: NFTs, cryptocurrencies, Web3 and Wild Markets." Journal of Business Research 153: 198-205.
- Thien Huynh-The, Quoc-Viet Pham, Xuan-Qui Pham, Thanh Thi Nguyen, Zhu Han, and Dong-Seong Kim (2022). "Artificial Intelligence for the Metaverse: A Survey." arXiv preprint arXiv:2202.10336.
- Thippa Reddy Gadekallu, Thien Huynh-The, Weizheng Wang, Gokul Yenduri, Pasika Ranaweera, Quoc-Viet Pham, Daniel Benevides da Costa, and Madhusanka Liyanage (2022). "Blockchain for the Metaverse: A Review." arXiv preprint arXiv:2203.09738.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805.
- OpenAI (2023), A Survey of Techniques for Maximizing LLM Performance, <https://www.youtube.com/watch?v=ahnGLM-RC1Y>
- The Super Duper NLP Repo, <https://notebooks.quantumstat.com/>
- NLP with Transformer, <https://github.com/nlp-with-transformers/notebooks>
- Min-Yuh Day (2023), Python 101, <https://tinyurl.com/aintpupython101>