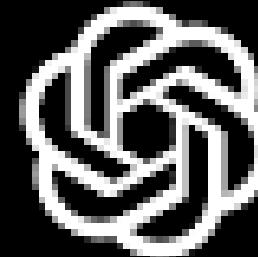


THE PROMISE OF ARTIFICIAL INTELLIGENCE IN MEDICINE IS TO PROVIDE COMPOSITE, PANORAMIC VIEWS OF INDIVIDUALS' MEDICAL DATA; TO IMPROVE DECISION MAKING.

Medical Chatbot



Improving medical decision making



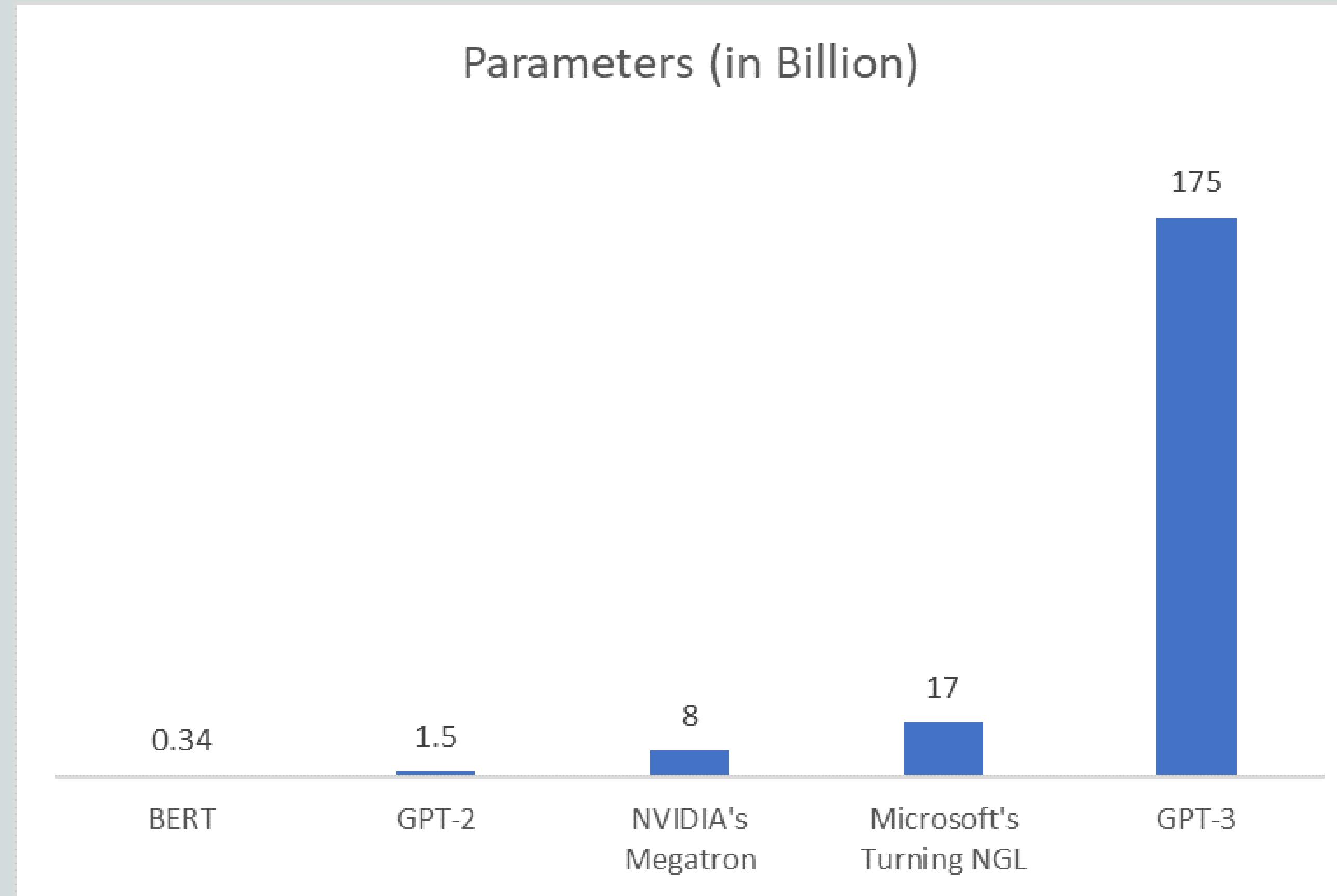
OpenAI



What is OpenAI?

OpenAI has trained cutting-edge language models that are very good at understanding and generating text. OpenAI provides access to these models and can be used to solve virtually any task that involves processing language.

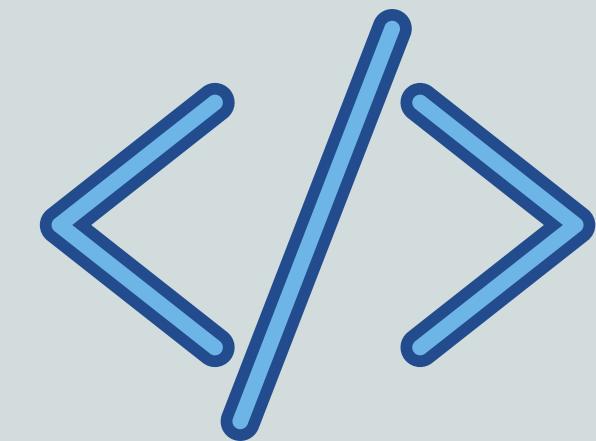
GPT-3



To make sense of a model, we need to set some variables

POST /v1/completions

```
1 import os  
2 import openai  
3  
4 openai.api_key = os.getenv("OPENAI_API_KEY")  
5  
6 response = openai.Completion.create(  
7     model="davinci:ft-personal-2022-05-13-09-16-40",  
8     prompt="",  
9     temperature=0.03,  
10    max_tokens=256,
```



Prompt & Completion

- disease diagnosis
- describe it
- give 3 precautions



Chat With Bot



close

my symptoms: skin rash,
nodal skin eruptions,
dischromic patches



Name: fungal infection.

Precaution one: bath twice.

Precaution two: use detol or
neem in bathing water.

Precaution three: keep
infected area dry.

Precaution four: use clean
cloths.

Description: In humans,
fungal infections occur when

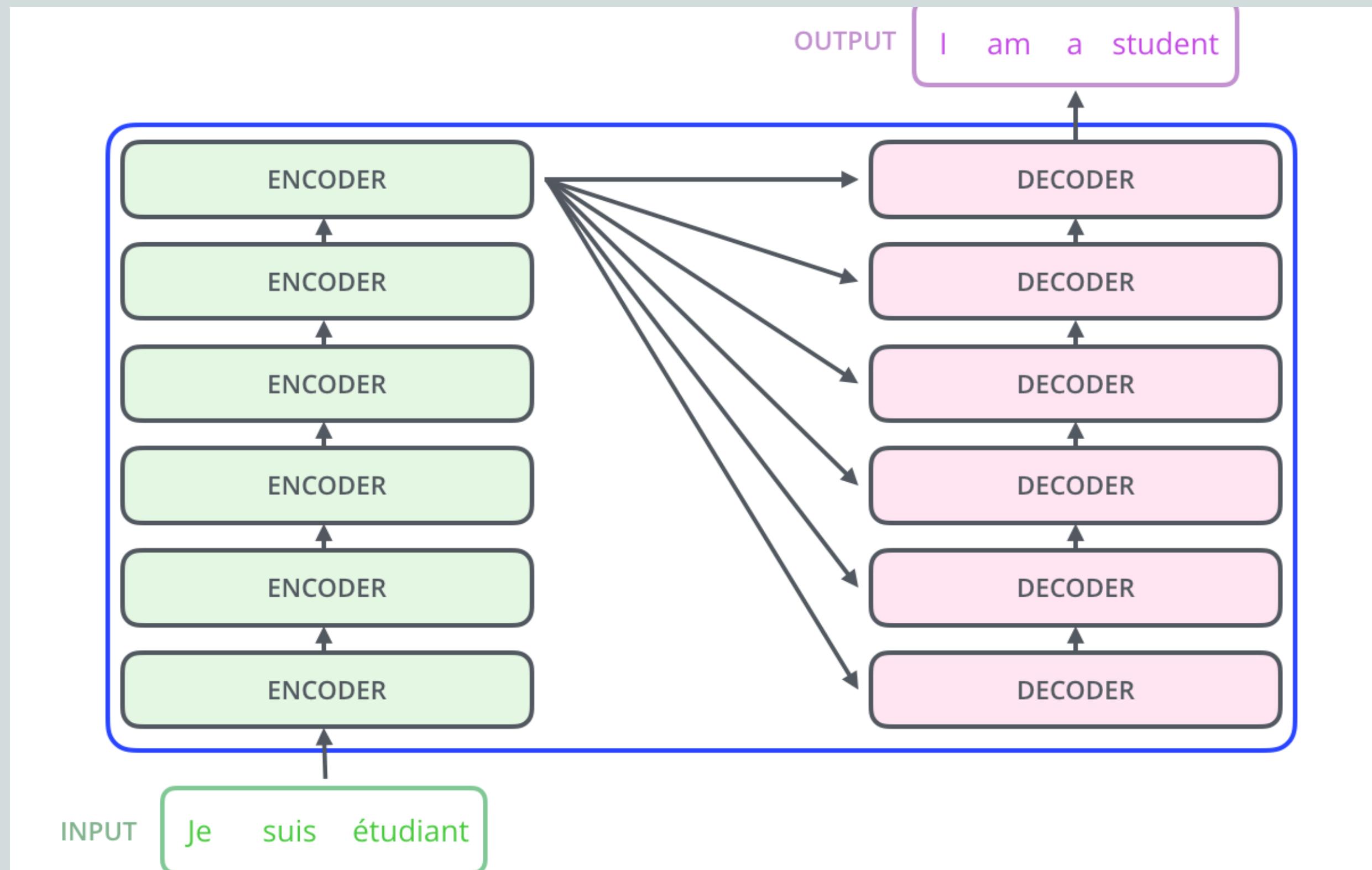
Send a message...

send



How GPT-3 Actually Works, From the Ground Up?

GPT-3 uses a special language model that is divided into two parts: the encoder and the decoder

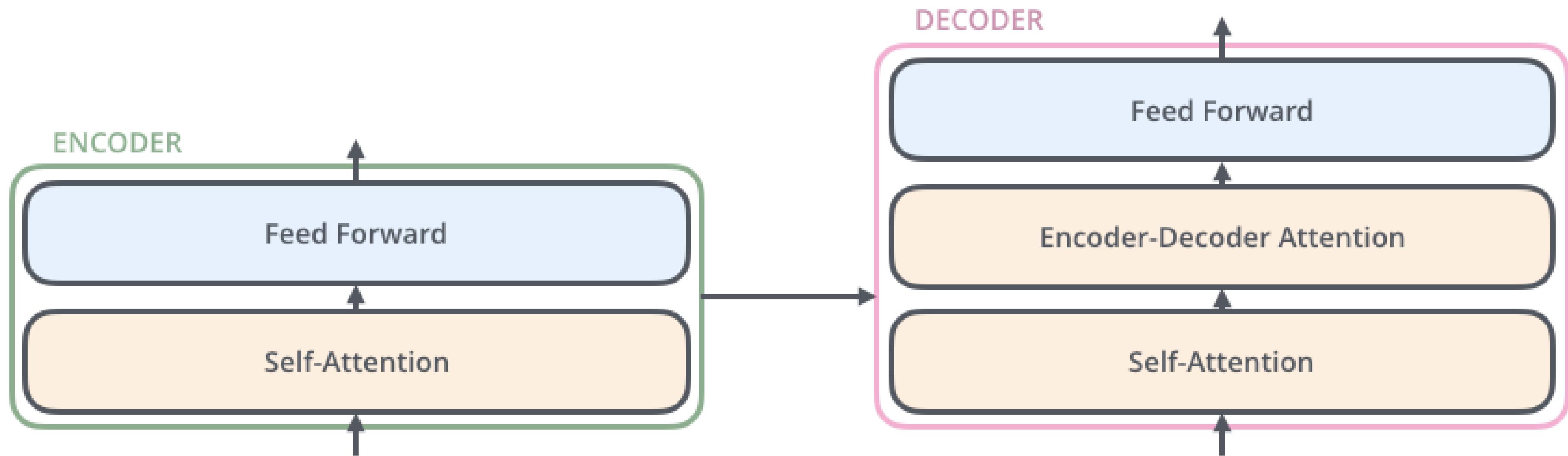


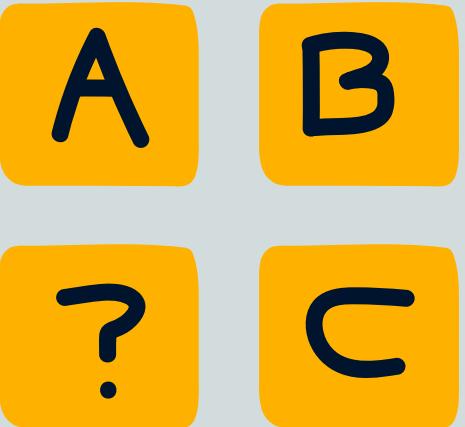
ENCODER

Feed Forward Neural Network

Self-Attention







Embedding algorithm

Transforming center words into vectors

Corpus I am happy because I am learning

Vocabulary am, because, happy, I, learning

Corpus

I am happy because I am learning

Vocabulary

am, because, happy, I, learning

One-hot
vector

am
because
happy
I
learning



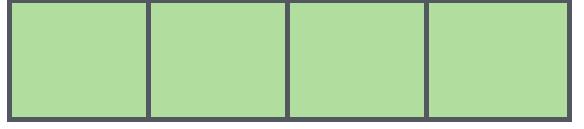
Average of individual one-hot vectors

$$\left[\begin{array}{c} I \\ am \\ because \\ happy \\ I \\ learning \end{array} \right] + \left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{array} \right] + \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{array} \right] / 4 = \left[\begin{array}{c} 0.25 \\ 0.25 \\ 0 \\ 0.5 \\ 0 \end{array} \right]$$

Final prepared training set

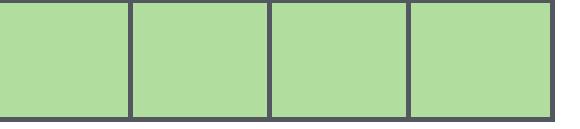
<i>Context words</i>	<i>Context words vector</i>	<i>Center word</i>	<i>Center word vector</i>
I am because I	[0.25; 0.25; 0; 0.5; 0]	happy	[0; 0; 1; 0; 0]

x₁



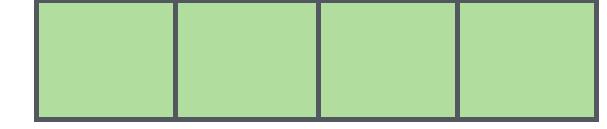
Je

x₂



suis

x₃



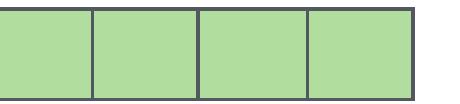
étudiant

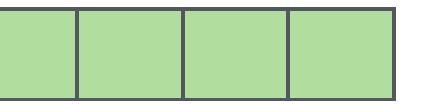
Input

Thinking

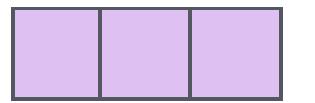
Machines

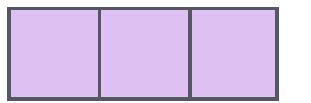
Embedding

\mathbf{x}_1 

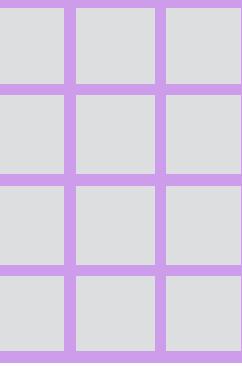
\mathbf{x}_2 

Queries

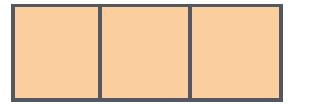
\mathbf{q}_1 

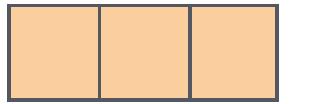
\mathbf{q}_2 

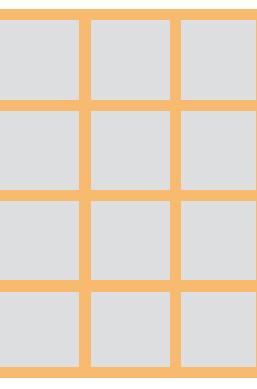
\mathbf{WQ}



Keys

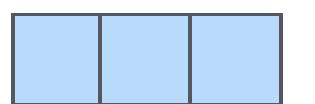
\mathbf{k}_1 

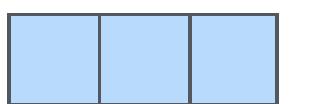
\mathbf{k}_2 

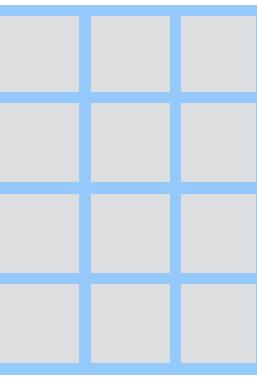


\mathbf{WK}

Values

\mathbf{v}_1 

\mathbf{v}_2 



\mathbf{WV}

Input

Embedding

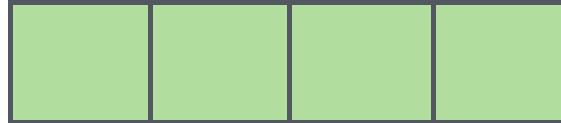
Queries

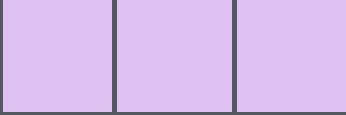
Keys

Values

Score

Thinking

x_1 

q_1 

k_1 

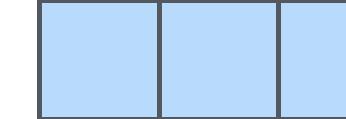
v_1 

Machines

x_2 

q_2 

k_2 

v_2 

$$q_1 \bullet k_1 = 112$$

$$q_1 \bullet k_2 = 96$$

Input

Embedding

Queries

Keys

Values

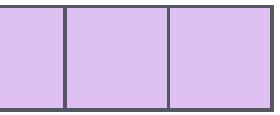
Score

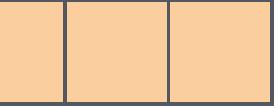
Divide by 8 ($\sqrt{d_k}$)

Softmax

Thinking

x_1 

q_1 

k_1 

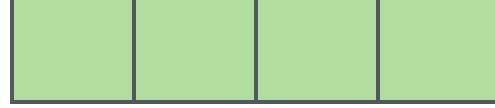
v_1 

$$q_1 \bullet k_1 = 112$$

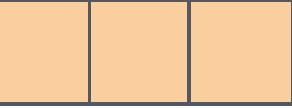
14

0.88

Machines

x_2 

q_2 

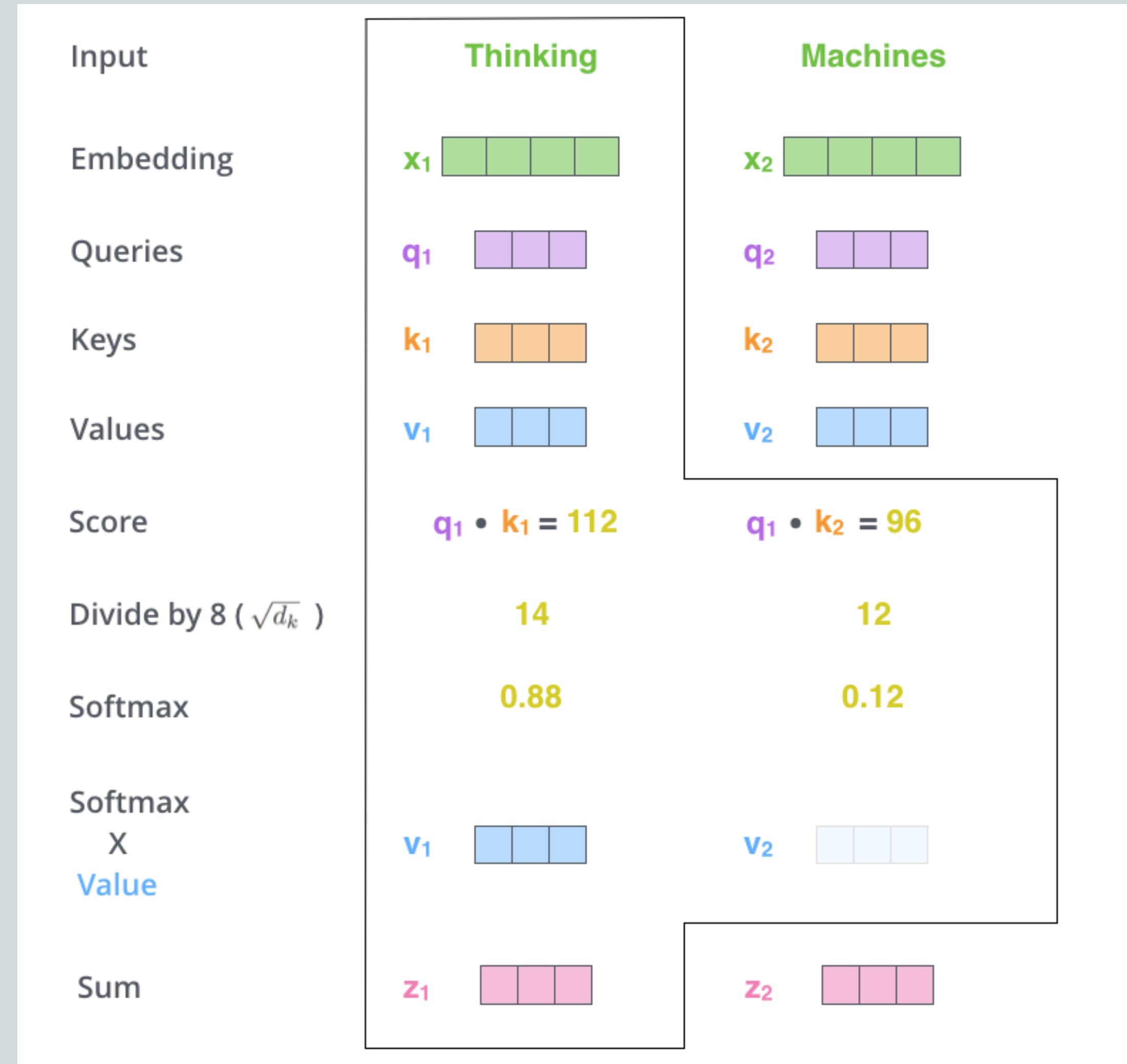
k_2 

v_2 

$$q_1 \bullet k_2 = 96$$

12

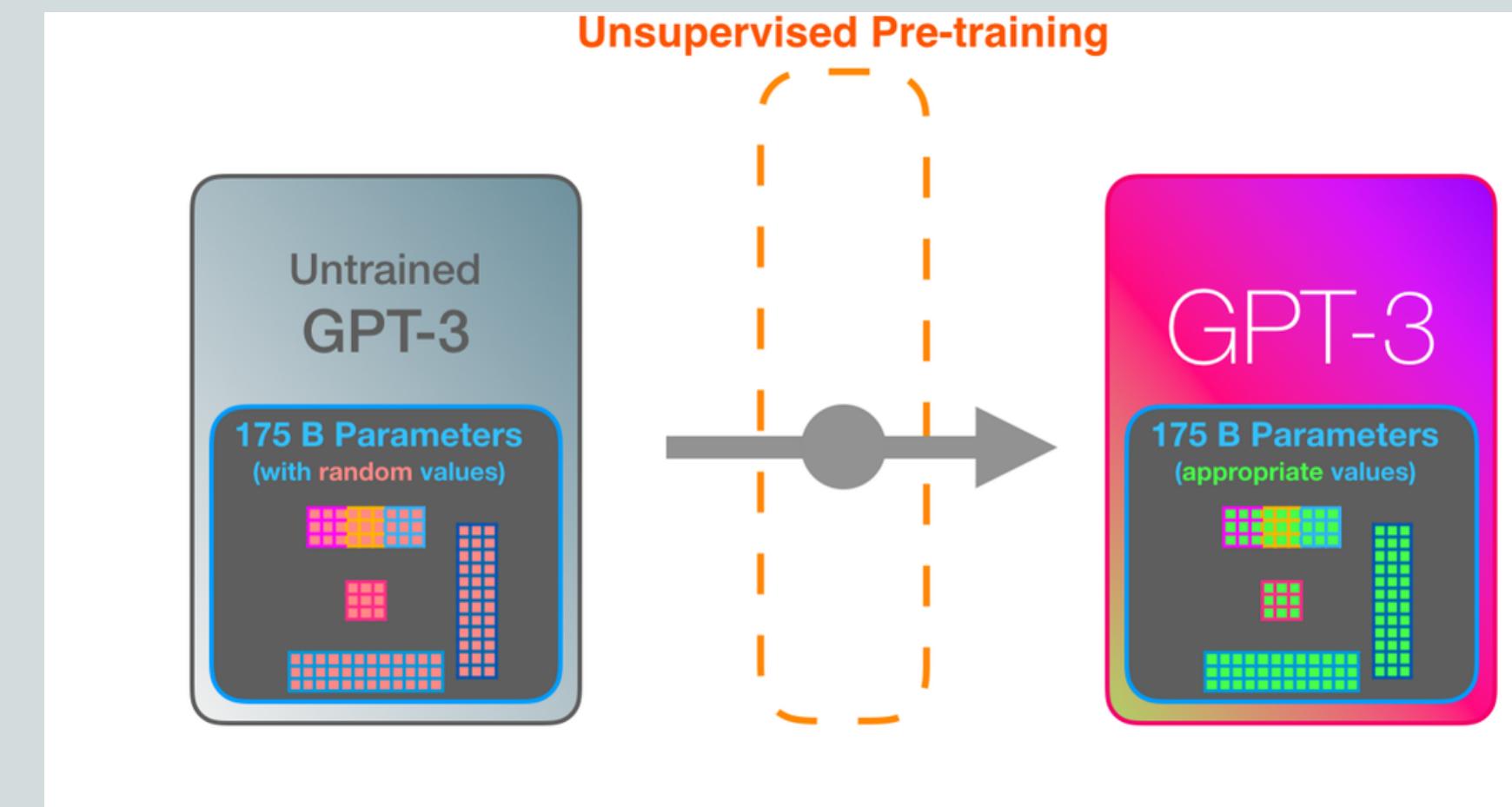
0.12



Training the model

Teaching the model how to process given input, by using next word prediction

It's done by giving the parameter a random value before training. And after giving the input and repeating this process millions of times, error prediction is used to generate the best prediction



Input Prompt:

Recite the first law of robotics

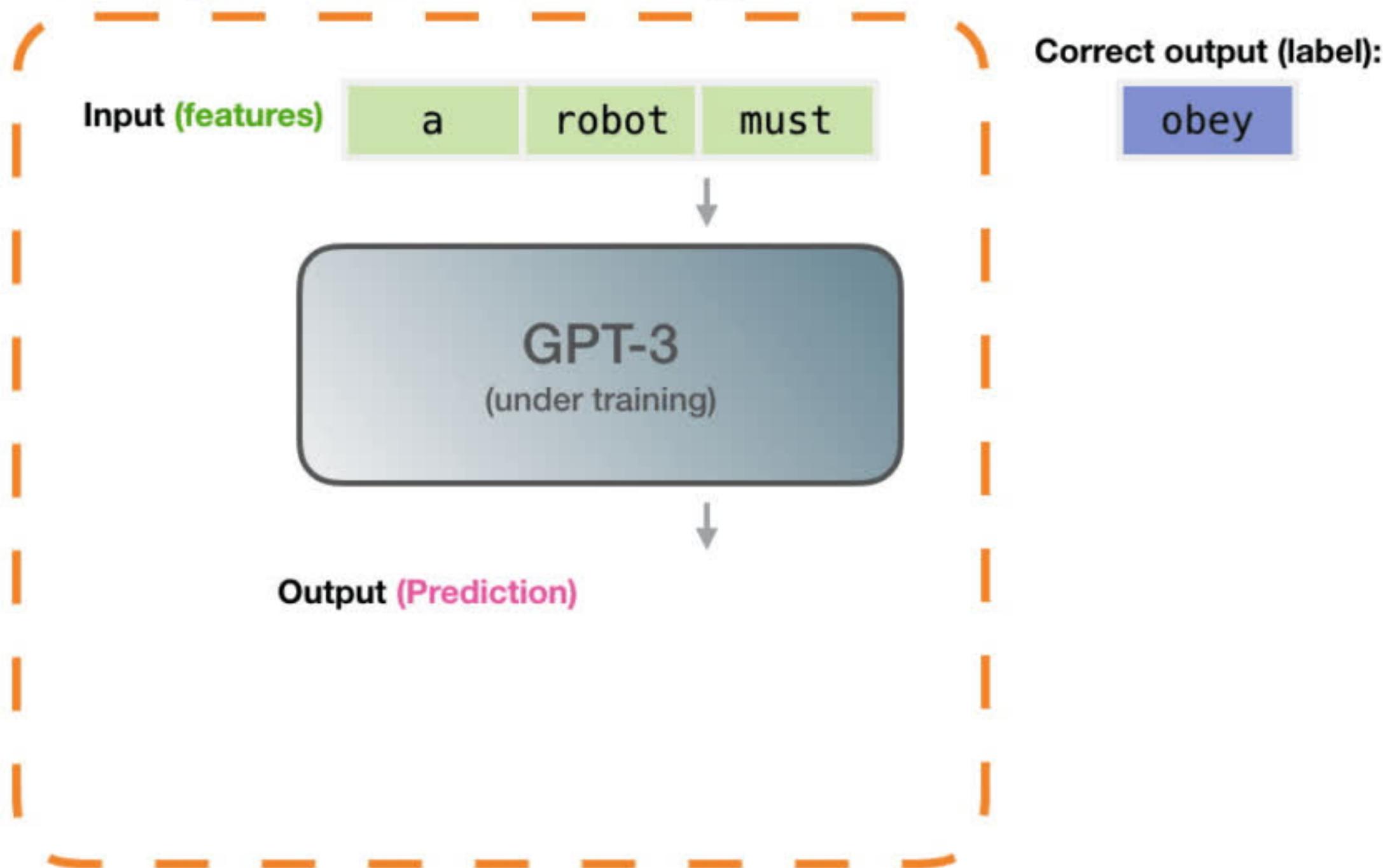


GPT-3



Output:

Unsupervised Pre-training



First , We must download OpenAi Api by using command prompt (pip install —upgrade openai)

```
(Users\user>pip install --upgrade openai
Requirement already satisfied: openai in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (0.18.1)
Collecting openai
  Downloading openai-0.19.0.tar.gz (42 kB)
    42.4/42.4 KB 518.5 kB/s eta 0:00:00
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: pandas-stubs>=1.1.0.11 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from openai) (1.2.0.58)
Requirement already satisfied: requests>=2.20 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from openai) (2.27.1)
Requirement already satisfied: pandas>=1.2.3 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from openai) (1.4.2)
Requirement already satisfied: tqdm in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from openai) (4.64.0)
Requirement already satisfied: openpyxl>=3.0.7 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from openai) (3.0.9)
Requirement already satisfied: et-xmlfile in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from openpyxl>=3.0.7->openai) (1.1.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from pandas>=1.2.3->openai) (2022.1)
Requirement already satisfied: numpy>=1.21.0 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from pandas>=1.2.3->openai) (1.22.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from pandas>=1.2.3->openai) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.20->openai) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.20->openai) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.20->openai) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from requests>=2.20->openai) (2.0.4)

Requirement already satisfied: colorama in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from tqdm->openai) (0.4.4)
Requirement already satisfied: six>=1.5 in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.8.1->pandas>=1.2.3->openai) (1.16.0)
Building wheels for collected packages: openai
Building wheel for openai (pyproject.toml) ... done
Created wheel for openai: filename=openai-0.19.0-py3-none-any.whl size=53511 sha256=cd9bc44daaa6201943ce257149cee8217ea7e5e1a7f98d3bc5ea9fb397237e6a
Stored in directory: c:\users\user\appdata\local\pip\cache\wheels\ca\5c\0f\2e2eeef583f5b213b178cfbf881921046b259f79fb5527aa6e
Successfully built openai
Installing collected packages: openai
Attempting uninstall: openai
  Found existing installation: openai 0.18.1
```

Second , Add OPEN_AI_KEY to the system environment

System variables	
Variable	Value
JAVA_HOME	C:\Program Files\OpenJDK\openjdk-11.0.15_10
NUMBER_OF_PROCESSORS	4
OPENAI_API_KEY	sk-wNKPLdEIKpiej8G7nAd0T3B1bkFJVisKbizU63gZB0y1gnlz
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
POWERSHELL DISTRIBUTIO...	MSI:Windows 10 Pro

Third, The data training must be in JSONL format:

```
`{"prompt": "<prompt text>", "completion": "<ideal generated text>"}`
```

We used python to split the data into prompt and completion

- Read Data from CSV file

```
import pandas as pd  
df = pd.read_csv('dataset.csv')
```

1

Data example



	prompt	Completion
0	i have: itching, skin rash, nodal skin eruptio...	fungal infection END
1	i have: skin rash, nodal skin eruptions, disc...	fungal infection END
2	i have: itching, nodal skin eruptions, dischro...	fungal infection END
3	i have: itching, skin rash, dischromic patche...	fungal infection END
4	i have: itching, skin rash, nodal skin eruptio...	fungal infection END
...
4915	i have: vomiting, headache, nausea, spinning ... (vertigo) paroxysmal positional vertigo END	
4916	i have: skin rash, pus filled pimples, blackh...	acne END
4917	i have: burning micturition, bladder discomfo...	urinary tract infection END
4918	i have: skin rash, joint pain, skin peeling, ...	psoriasis END
4919	i have: skin rash, high fever, blister, red s...	impetigo END
4920	rows × 2 columns	

Adding "I have" to make the AI sound more Human

```
▶ from numpy import NaN
p = []
for rowIndex, row in df.iterrows():
    str1="i have: "
    for columnIndex, value in row.items():
        if columnIndex != 'Disease' and str(value) != "nan":
            str1=str1+str(value).lower()+" ,"
            str1=str1.replace("_"," ")
    str1=str1[0:len(str1)-1]+" ->"
    p.append(str1)
```

```
[ ] df1=pd.DataFrame()
df1['prompt'] = p

[ ] p1=[]
for value in df['Disease']:
    p1.append(str(value).lower()+" END")

[ ] df1['Completion']=p1

[ ] df1
```

+ Code

+ Text

Final data form

	prompt	Completion
0	i have: itching, skin rash, nodal skin eruptio...	fungal infection END
1	i have: skin rash, nodal skin eruptions, disc...	fungal infection END
2	i have: itching, nodal skin eruptions, dischro...	fungal infection END
3	i have: itching, skin rash, dischromic patche...	fungal infection END
4	i have: itching, skin rash, nodal skin eruptio...	fungal infection END
...
4915	i have: vomiting, headache, nausea, spinning ... (vertigo) paroxysmal positional vertigo END	
4916	i have: skin rash, pus filled pimples, blackh...	acne END
4917	i have: burning micturition, bladder discomfo...	urinary tract infection END
4918	i have: skin rash, joint pain, skin peeling, ...	psoriasis END
4919	i have: skin rash, high fever, blister, red s...	impetigo END
4920 rows x 2 columns		

Converting from CSV to JSONL

`openai tools fine_tunes.prepare_data -f <csv file name>`

```
C:\Users\user\Desktop>openai tools fine_tunes.prepare_data -f ds4.csv

Your file extension, your file is formatted as a CSV file
Your file contains 5420 prompt-completion pairs
The `completion` column/key should be lowercase
From your data it seems like you're trying to fine-tune a model for classification
For classification, we recommend you try one of the faster and cheaper models, such as `ada`
For classification, you can estimate the expected model performance by keeping a held out dataset, which is not used for training
4616 duplicated prompt-completion sets. These are rows: [5, 6, 7, 8, 9, 15, 16, 17, 18, 19, 27, 28, 29, 38, 39, 46, 47, 48, 49, 53, 54, 59, 65, 86, 87, 88, 89, 97, 98, 99, 106, 107, 108, 109, 121, 127, 128, 129, 131, 136, 137, 138, 139, 141, 152, 159, 188, 191, 201, 216, 218, 219, 232, 281, 282, 288, 289, 291, 296, 297, 298, 299, 301, 309, 311, 313, 321, 332, 341, 348, 349, 351, 356, 358, 359, 361, 368, 369, 371, 376, 377, 378, 389, 391, 398, 399, 401, 404, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 12010, 12011, 12012, 12013, 12014, 12015, 12016, 12017, 12018, 12019, 12020, 12021, 12022, 12023, 12024, 12025, 12026, 12027, 12028, 12029, 12030, 12031, 12032, 12033, 12034, 12035, 12036, 12037, 12038, 12039, 12040, 12041, 12042, 12043, 12044, 12045, 12046, 12047, 12048, 12049, 12050, 12051, 12052, 12053, 12054, 12055, 12056, 12057, 12058, 12059, 12060, 12061, 12062, 12063, 12064, 12065, 12066, 12067, 12068, 12069, 12070, 12071, 12072, 12073, 12074, 12075, 12076, 12077, 12078, 12079, 12080, 12081, 12082, 12083, 12084, 12085, 12086, 12087, 12088, 12089, 12090, 12091, 12092, 12093, 12094, 12095, 12096, 12097, 12098, 12099, 120100, 120101, 120102, 120103, 120104, 120105, 120106, 120107, 120108, 120109, 120110, 120111, 120112, 120113, 120114, 120115, 120116, 120117, 120118, 120119, 120120, 120121, 120122, 120123, 120124, 120125, 120126, 120127, 120128, 120129, 120130, 120131, 120132, 120133, 120134, 120135, 120136, 120137, 120138, 120139, 120140, 120141, 120142, 120143, 120144, 120145, 120146, 120147, 120148, 120149, 120150, 120151, 120152, 120153, 120154, 120155, 120156, 120157, 120158, 120159, 120160, 120161, 120162, 120163, 120164, 120165, 120166, 120167, 120168, 120169, 120170, 120171, 120172, 120173, 120174, 120175, 120176, 120177, 120178, 120179, 120180, 120181, 120182, 120183, 120184, 120185, 120186, 120187, 120188, 120189, 120190, 120191, 120192, 120193, 120194, 120195, 120196, 120197, 120198, 120199, 120200, 120201, 120202, 120203, 120204, 120205, 120206, 120207, 120208, 120209, 120210, 120211, 120212, 120213, 120214, 120215, 120216, 120217, 120218, 120219, 120220, 120221, 120222, 120223, 120224, 120225, 120226, 120227, 120228, 120229, 120230, 120231, 120232, 120233, 120234, 120235, 120236, 120237, 120238, 120239, 120240, 120241, 120242, 120243, 120244, 120245, 120246, 120247, 120248, 120249, 120250, 120251, 120252, 120253, 120254, 120255, 120256, 120257, 120258, 120259, 120260, 120261, 120262, 120263, 120264, 120265, 120266, 120267, 120268, 120269, 120270, 120271, 120272, 120273, 120274, 120275, 120276, 120277, 120278, 120279, 120280, 120281, 120282, 120283, 120284, 120285, 120286, 120287, 120288, 120289, 120290, 120291, 120292, 120293, 120294, 120295, 120296, 120297, 120298, 120299, 120300, 120301, 120302, 120303, 120304, 120305, 120306, 120307, 120308, 120309, 120310, 120311, 120312, 120313, 120314, 120315, 120316, 120317, 120318, 120319, 120320, 120321, 120322, 120323, 120324, 120325, 120326, 120327, 120328, 120329, 120330, 120331, 120332, 120333, 120334, 120335, 120336, 120337, 120338, 120339, 120340, 120341, 120342, 120343, 120344, 120345, 120346, 120347, 120348, 120349, 120350, 120351, 120352, 120353, 120354, 120355, 120356, 120357, 120358, 120359, 120360, 120361, 120362, 120363, 120364, 120365, 120366, 120367, 120368, 120369, 120370, 120371, 120372, 120373, 120374, 120375, 120376, 120377, 120378, 120379, 120380, 120381, 120382, 120383, 120384, 120385, 120386, 120387, 120388, 120389, 120390, 120391, 120392, 120393, 120394, 120395, 120396, 120397, 120398,
```

RNN & LSTM

Multi-Layer Perceptrons applied to NLP

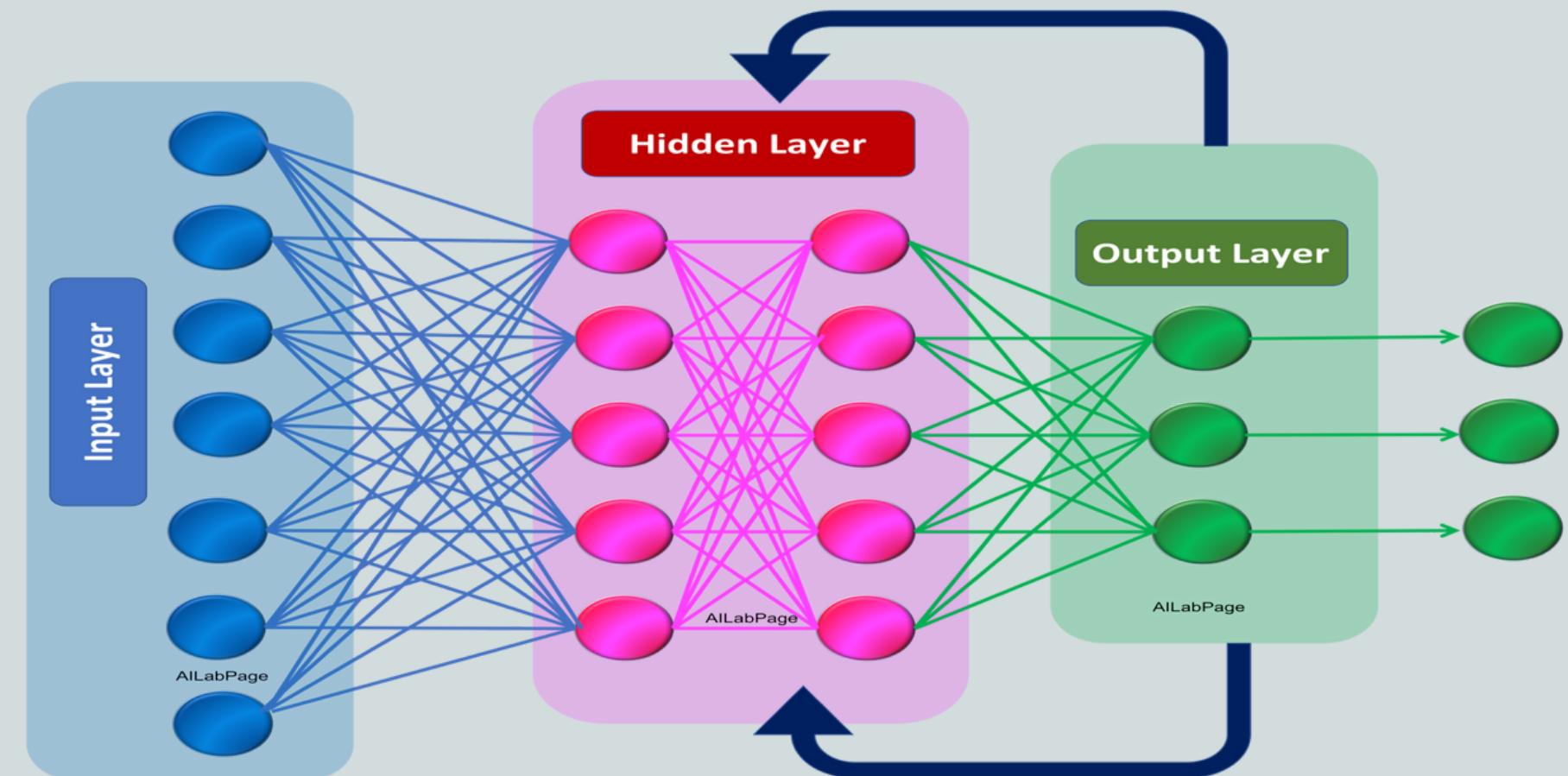
Limitation of MLPs when applied to NLP

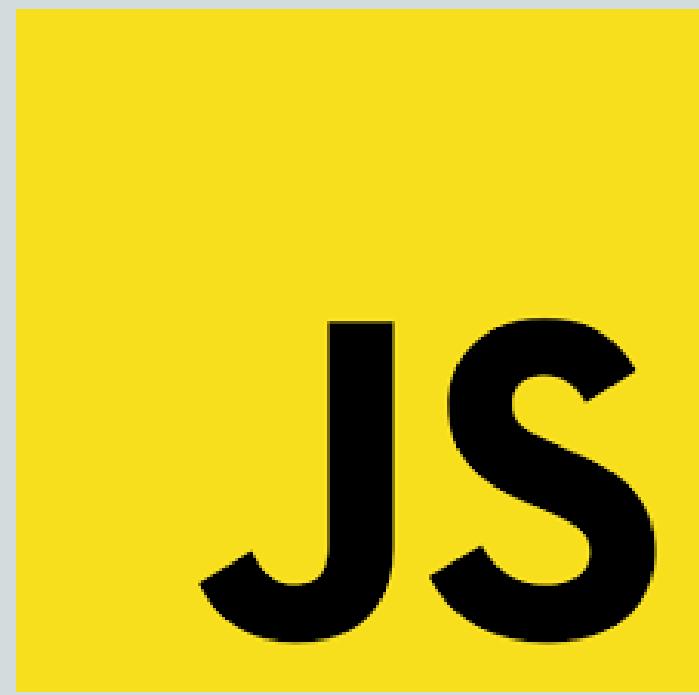
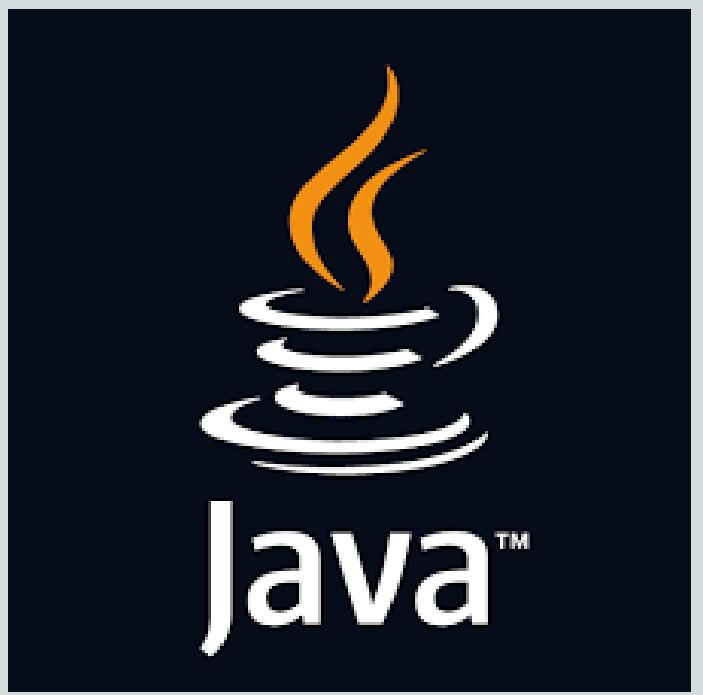
Why Recurrent Neural Networks (RNNs) are more appropriate

The vanishing gradient problem

Long Short Term Memory Networks

Recurrent Neural Networks





DEMO