PaLM: Scaling Language Modeling with Pathways

Adrienne Wright Yuting Zhao Martin Ivanov



Structure of presentation

- 1. Introduction to Pathways Language Model (PaLM)
 - a. Motivations
 - b. Training data
 - c. Breakthrough capabilities
- 2. PaLM architecture
- 3. GPT-3 embeddings
- 4. Implementation
- 5. Future work
- 6. Conclusion

Motivations

1. To further the understanding of the impact of scale on few-shot learning



As the scale of the model increases, the performance improves across tasks while also unlocking new capabilities.

Motivations

2. Pathways system (released in 10.2021) enables highly efficient training across multiple TPU Pods

Model	# of Parameters (in billions)	Accelerator chips	Model FLOPS utilization
GPT-3	175B	V100	21.3%
Gopher	$280\mathrm{B}$	4096 TPU v3	32.5%
Megatron-Turing NLG	530B	2240 A100	30.2%
PaLM	540B	6144 TPU v4	46.2%

Training data

Total dataset size $= 780$ billion tokens						
Data source	Proportion of data					
Social media conversations (multilingual)	50%					
Filtered webpages (multilingual)	27%					
Books (English)	13%					
GitHub (code)	5%					
Wikipedia (multilingual)	4%					
News (English)	1%					

Breakthrough capabilities

Language Understanding and Generation



PaLM 540B performance improvement over prior state-of-the-art (SOTA) results on 29 English-based NLP tasks.

Breakthrough capabilities

Language Understanding and Generation



Scaling behavior of PaLM on a subset of 58 BIG-bench tasks



PaLM results on the BIG-bench textual task collection (150 tasks)

Breakthrough capabilities on several BIG-Bench tasks

Cause & Effect

Prompt

Which of the following sentences makes more sense?

- 1. I studied hard because I got an A on the test.
- 2. I got an A on the test because I studied hard.

Model Response

I got an A on the test because I studied hard.

Breakthrough capabilities on several BIG-Bench tasks

Emoji Movie Game

Prompt

What movie does this emoji describe? 齡 🕷 🎢 🌖

- 1. l.a. confidential
- 2. wall-e
- 3. léon: the professional
- 4. big
- 5. rush

Model Response

wall-e

Breakthrough capabilities on several BIG-Bench tasks

Synonym Game

Prompt

Try to identify the 2 words best associated with the word STUMBLE from the following list: table, jam, kangaroo, lock, australia, centaur, ham, calf, web, robin, battery, spell, pilot, ruler, trip, jack, dragon, fall. Give your answer in alphabetical order.

Model Response

fall, trip

Breakthrough capabilities

Explaining a joke

Prompt

Explain this joke:

Joke: Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!

Model Response

Prediction: TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two groups of TPUs.

The answer is 50.

Breakthrough capabilities

Reasoning

Standard Prompting	Chain of thought prompting
Example Input	Example Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?	Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
Example Output	Example Output
A: The answer is 11.	Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.
Prompt	Prompt
The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?	The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Response 🗙	Model Response

The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23-20 = 3. They bought 6 more apples, so they have 3+6=9. The answer is 9.

Breakthrough capabilities

Code Generation

Prompt	Model Response
<pre>def exercise5(): """Marie ordered one chicken meal that costs \$12, 5 packs of milk that cost \$3 each, 4 apples that cost \$1.50 each, and some boxes of pizza. Marie paid a total of \$50. How many boxes of pizza did Marie order if box costs \$8.50?"""</pre>	<pre>total_paid = 50 cost_of_pizza_per_box = 8.50 cost_of_meal = 12 cost_of_milk_per_pack = 3 cost_of_apples_per_apple = 1.50 packages_of_milk = 5</pre>
	<pre>return float(total_paid - cost_of_meal- cost_of_milk_per_pack * packages_of_milk - cost_of_apples_per_apple * 4)/ cost_of_pizza_per_box</pre>

Examples of a fine-tuned PaLM 540B model on text-to-code tasks, such as GSM8K-Python and HumanEval, and code-to-code tasks, such as Transcoder.

Breakthrough capabilities

Fixing Code Errors



An example from the DeepFix Code Repair task. The fine-tuned PaLM-Coder 540B fixes compilation errors (**left**, in red) to a version of code that compiles.

PaLM: Scaling Language Modelling with Pathways ARCHITECTURE

- decoder-only, densely activated, autoregressive
- trained using pathways
- 780B tokens of data
- 540B parameters
- Multi-Query Attention
- Dropout
- SwiGLU Activation
- Parallel Layers

780B tokens

Total dataset size $= 780$ billion tokens							
Data source	Proportion of data						
Social media conversations (multilingual)	50%						
Filtered webpages (multilingual)	27%						
Books (English)	13%						
GitHub (code)	5%						
Wikipedia (multilingual)	4%						
News (English)	1%						

ISO Code	Language	$\operatorname{Tokens}(B)$	Percentage
en	English	578.064	77.984%
$\mathbf{d}\mathbf{e}$	German	25.954	3.501%
fr	French	24.094	3.250%
es	Spanish	15.654	2.112%
und	Unknown	12.064	1.628%
pl	Polish	10.764	1.452%
it	Italian	9.699	1.308%
nl	Dutch	7.690	1.037%
sv	Swedish	5.218	0.704%
tr	Turkish	4.855	0.655%
\mathbf{pt}	Portuguese	4.701	0.634%
ru	Russian	3.932	0.530%
fi	Finnish	3.101	0.418%
cs	Czech	2.991	0.404%
$\mathbf{z}\mathbf{h}$	Chinese	2.977	0.402%

Pathways



Pathways: Asynchronous Distributed Dataflow for ML



Pathways: Asynchronous Distributed Dataflow for ML





(b) Parallel dispatch



Pathways in PaLM



Multi-query attention

• Classic multi-head attention



Multi-query attention

- Multi-query multi-head attention
- Key / value projections shared \rightarrow [1,h]
- Query \rightarrow [k,h]



Multi-query attention

def MultiheadAttentionBatched(def MultiqueryAttentionBatched(
X, M, mask, P_q, P_k, P_v, P_o):	X, M, mask, P_q, P_k, P_v, P_o):
""" Multi-head Attention.	""" Multi-Query Attention.
Args:	Args:
X: a tensor with shape [b, n, d]	X: a tensor with shape [b, n, d]
M: a tensor with shape [b, m, d]	M: a tensor with shape [b, m, d]
mask: a tensor with shape [b, h, n, m]	mask: a tensor with shape [b, h, n, m]
P_q : a tensor with shape $[h, d, k]$	P_q : a tensor with shape $[h, d, k]$
P_k : a tensor with shape $[h, d, k]$	P_k : a tensor with shape $[d, k]$
P_v: a tensor with shape [h, d, v]	P_v : a tensor with shape d, v]
$P_o: a tensor with shape [h, d, v]$	P o: a tensor with shape $[h, d, v]$
Returns:	Returns:
Y: a tensor with shape [b, n, d]	Y: a tensor with shape [b, n, d]
$Q = tf.einsum("bnd, hdk->bhnk", X, P_q)$	$Q = tf.einsum("bnd.hdk->bhnk", X, P_q)$
$K = tf.einsum("bmd, hdk->bhmk", M, P_k)$	$K = tf.einsum("bmd.dk->bmk", M, P_k)$
$V = tf.einsum("bmd, hdv->bhmv", M, P_v)$	$V = tf.einsum("bmd.dv->bmv", M, P_v)$
logits = tf.einsum("bhnk,bhnk->bhnm", Q, K)	logits = tf.einsum("bhnk,bmk->bhnm", Q, K)
weights = $tf.softmax(logits + mask)$	weights = $tf.softmax(logits + mask)$
O = tf.einsum("bhnm,bhmv->bhnv", weights, V)	O = tf.einsum("bhnm,bmv->bhnv", weights, V)
$Y = tf.einsum("bhnv,hdv->bnd", O, P_o)$	$Y = tf.einsum("bhnv,hdv->bnd", O, P_o)$
return Y	return Y

Training

- No dropout during training
- Typically 0.1 for fine-tuning

• No bias \rightarrow which takes us to the next slide / point

Activation function

- SwiGLU variation
 - A Swish function from the GLU family.

Swish - $f(x) = x \cdot sigmoid(\beta x)$

SwiGLU(x,W,V,b,c, β)=Swish β (xW +b) \otimes (xV +c)

FFN SwiGLU(x,W,V,W2) = (Swish(xW) \otimes xV)W2 \rightarrow typically used in transformers



Parallel Layers

- y = x + MLP(LayerNorm(x)) + Attention(LayerNorm(x))
- (Instead of y = x + MLP(LayerNorm(x + Attention(LayerNorm(x))))



Model Name	$n_{ m params}$	$n_{ m layers}$	$d_{ m model}$	$n_{ m heads}$	$d_{ m head}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 imes 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 imes 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 imes 10^{-4}$
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	$0.6 imes 10^{-4}$

Model	Layers	# of Heads	$d_{ m model}$	# of Parameters (in billions)	Batch Size
PaLM 8B	32	16	4096	8.63	$256 \rightarrow 512$
PaLM 62B	64	32	8192	62.50	$512 \rightarrow 1024$
PaLM 540B	118	48	18432	540.35	$512 \rightarrow 1024 \rightarrow 2048$

Model Name	$n_{ m params}$	$n_{\rm layers}$	$d_{ m model}$	$n_{ m heads}$	$d_{ m head}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 imes 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 imes 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 imes 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 imes 10^{-4}$
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	$0.6 imes 10^{-4}$

Model	Layers $\#$ of He		$d_{ m model}$	# of Parameters (in billions)	Batch Size
PaLM 8B	32	16	4096	8.63	$256 \rightarrow 512$
PaLM 62]	3 64	32	8192	62.50	$512 \rightarrow 1024$
PaLM 540)B 118	48	18432	540.35	$512 \rightarrow 1024 \rightarrow 2048$

Model Name	$n_{ m params}$	$n_{ m layers}$	$d_{ m model}$	$n_{ m heads}$	$d_{ m head}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 imes 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 imes 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 imes 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	$1.6 imes 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 imes 10^{-4}$
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	$0.6 imes 10^{-4}$

Model	Layers	# of Heads	$d_{ m model}$	# of Parameters (in billions)	Batch Size
PaLM 8B PaLM 62B PaLM 540B	$32 \\ 64 \\ 118$	$16\\32\\48$	$4096 \\ 8192 \\ 18432$	$8.63 \\ 62.50 \\ 540.35$	$\begin{array}{c} 256 \rightarrow 512 \\ 512 \rightarrow 1024 \\ 512 \rightarrow 1024 \rightarrow 2048 \end{array}$

Model Name	$n_{ m params}$	$n_{ m layers}$	$d_{ m model}$	$n_{ m heads}$	$d_{ m head}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 imes 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 imes 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5 imes 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 imes 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 imes 10^{-4}$
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	$0.6 imes 10^{-4}$

Model	
PaLM 8B PaLM 62B PaLM 540B	256

However...



Vocabulary

- 256K tokens (vs 50K for GPT-3)
- SentencePiece tokenizer

Rotational Positional Encoding

- Word order during training is extremely important as illustrated in the other transformer architectures
- RoPE encodes the absolute position with a rotation matrix and incorporates the explicit relative position dependency in self-attention formulation





Figure 1: Implementation of Rotary Position Embedding(RoPE).

Time complexity of RoPE-models

 As opposed to the absolute and relative positional encoding used in e.g. GPT-3



Text and Code Embeddings by Contrastive Pre-Training

Arvind Neelakantan^{*1} Tao Xu^{*1} Raul Puri¹ Alec Radford¹ Jesse Michael Han¹ Jerry Tworek¹ Qiming Yuan¹ Nikolas Tezak¹ Jong Wook Kim¹ Chris Hallacy¹ Johannes Heidecke¹ Pranav Shyam¹ Boris Power¹ Tyna Eloundou Nekoul¹ Girish Sastry¹ Gretchen Krueger¹ David Schnurr¹ Felipe Petroski Such¹ Kenny Hsu¹ Madeleine Thompson¹ Tabarak Khan¹ Toki Sherbakov¹ Joanne Jang¹ Peter Welinder¹ Lilian Weng¹

Embedding models

- explicitly optimized to learn a low dimensional representation
- captures the semantic meaning of the input







 trained with a series of unsupervised text embedding models (cpt-text) of different sizes, ranging from 300M to 175B parameters

Text search Code search Sentence similari	ty Text classification
Model	Performance
text-embedding-ada-002	90.1
text-similarity-davinci-001	92.2
text-similarity-curie-001	91.5
text-similarity-babbage-001	91.1
text-similarity-ada-001	89.3

Dataset: SentEval (MR, CR, SUBJ, MPQA, SST, TREC, MRPC)

Training Pairs

- The training set consists of paired samples,

 $\{(x_{i}^{}, y_{i}^{})\}_{i=1}^{N}$

where (x_i, y_i) corresponds to a positive example pair, indicating that x_i and y_i are semantically similar or contextually relevant.

- Two special token delimiters, [SOS] and [EOS], to the start and end of the input sequence respectively.
- input trained with a transformer encoder



labels = np.arange(M)
l_r = cross_entropy(logits, labels, axis=0)
l_c = cross_entropy(logits, labels, axis=1)
loss = (l_r + l_c) / 2

$$v_x = E([SOS]_x \oplus x \oplus [EOS]_x)$$

$$v_y = E([SOS]_y \oplus y \oplus [EOS]_y)$$

$$sim(x,y) = \frac{v_x \cdot v_y}{\|v_x\| \cdot \|v_y\|}$$
logit(x_i, y_j) = sim(x_i, y_j) $\cdot exp(\tau)$,
 $\forall (i, j), i, j \in \{1, 2, ..., M\}$



Advantages

- No explicit labels
 - goal is to maximise the cosine similarity between the pairs and dissimilarity with the inverse.
- Positive pairs were neighboring pieces of text on the Internet

Implementation

```
import pandas as pd
import openai
from openai.embeddings_utils import get_embedding
def split_dataframe_by_position(df, splits):
    Takes a dataframe and an integer of the number of splits to create.
    Returns a list of dataframes.
    dataframes = []
                                         split a dataframe into
    index_to_split = len(df) // splits
                                         arbitrary parts
    start = 0
    end = index to split
    for split in range(splits):
        temporary_df = df.iloc[start:end, :]
        dataframes.append(temporary_df)
        start += index_to_split
        end += index_to_split
                                                    concatenate text + headline
    return dataframes
                                                    drop other fields
input_datapath = './data/news/classification/classification_news_eval.jsonl'
df = pd.read_json(input_datapath, lines=True)
df['text'] = df['headline'] + ". " + df['short_description']
df.drop(['headline', 'short_description', 'authors', 'link', 'date'], axis=1, inplace=True)
dfs = split_dataframe_by_position(df, 5)
                                            query the GPT3 embeddings API
openai.api_key = "secret"
                                            for each dataframe
for i in range(len(dfs)):
    print(f"processing df part {i+1}")
    dfs[i]['ada_similarity'] = dfs[i].text.apply(lambda x: get_embedding(x, engine='text-embedding-ada-002'))
    dfs[i].to_json(f"./classification_news_eval_with_embeddings_ada_similarity_part_{i}.json")
```

+ -

Terminal

... _ 🗆 X

Q

xlebu@toaster:code/embeddings_data_news% tree -h

	4].	
_	[7] eva	1
	— [51M]	<pre>classification_news_eval_with_embeddings_ada_similarity_part_0.json</pre>
	— [51M]	classification_news_eval_with_embeddings_ada_similarity_part_1.json
	— [51M]	<pre>classification_news_eval_with_embeddings_ada_similarity_part_2.json</pre>
	— [51M]	<pre>classification_news_eval_with_embeddings_ada_similarity_part_3.json</pre>
	└── [51M]	<pre>classification_news_eval_with_embeddings_ada_similarity_part_4.json</pre>
	[22] tra	in
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_0.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_10.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_11.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_12.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_13.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_14.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_15.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_16.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_17.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_18.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_19.jsc
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_1.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_2.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_3.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_4.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_5.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_6.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_7.jsor
	— [51M]	classification_news_train_with_embeddings_ada_similarity_part_8.jsor
	└── [51M]	classification_news_train_with_embeddings_ada_similarity_part_9.jsor

2 directories, 25 files xlebu@toaster:code/embeddings_data_news% 📕

		len <mark>(</mark> set(df_	train['ca	tegory'])) == 1	len <mark>(</mark> se	t <mark>(</mark> df_e	val['c	ategory']))
		True	train d	categ	ories	== e	val	categ	gories	
		len <mark>(</mark> set(df_	_train['ca	tegory']))					
		34	number	of c	ategor	ies				
		len(df_trai	ln['ada_sir	milarit	y'][0])					
		1536	embedd	ings	vecto	r le	ngth			
traa 21 21 15 16 16 16 16 16 16 16 16 16 16 16 17 17 17 17 17 17 17 17 18 18 18 18 18	in% :13 :23 :33 :12 :22 :33 :43 :55 :16 :25 :16 :25 :36 :25 :36 :25 :36 :25 :36 :25 :36 :25 :36 :25 :36 :25 :36 :25 :36 :25 :36 :25 :36 :36 :25 :36 :36 :36 :36 :36 :36 :36 :36 :36 :36	<pre>ls -lha s</pre>	ort -n s to get ings ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t ion_news_t	rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_ rain_						

18:51 classification_news_train_ 19:58 classification_news_train_ 20:08 classification_news_train_

, 9 10 11 12 13 14	<pre># load data train_base_path = " eval_base_path = " parts_train = [f"{tr parts_eval = [f"{trains_eval = [fd.read] dfs_eval = [pd.read]</pre>	<pre>/embeddings_data_news/train/classification_news_1 ./embeddings_data_news/evak/classification_news_evatrain_base_path}{i}.json" for i in range(20)] ain_base_path}{i}.json" for i in range(5)] d_json(part) for part in parts_train] _json(part) for part in parts_eval]</pre>	train_with_embeddings_ada_similarity_part_" al_with_embeddings_ada_similarity_part_"	0 2 ▲7 ☆3 ^ ∨
	df_train = pd.concat df_train	t(dfs_train)		
	<pre></pre>	1180 rows × 3 columns text Dr. Seuss' Prescription For A Bad Day (PHOTOS). G Bill Would Turn Veteran Bars Into Wellness Center Dana Carvey Relives A Classic 'SNL' Moment.	<pre>* ada_similarity . [-0.0051075476, 0.035722412200000005, 0.015265178 [0.0010024955000000001, -0.010291514, 0.010914037 [-0.016778038800000002, -0.022007722400000002, 0</pre>	csv ~ ± > ♀ : ; =
	df_eval = pd.concat df_eval	(dfs_eval)		
	<pre></pre>	2795 rows × 3 columns text Dr. Seuss' Prescription For A Bad Day (PHOTOS). G. Bill Would Turn Veteran Bars Into Wellness Center.	<pre>* ada_similarity . [-0.0051075476, 0.035722412200000005, 0.015265178 [0.0010024955000000001, -0.010291514, 0.010914037</pre>	CSV ∨ ± > • • : ÷
	df_train = pd.read_; df_train set(df_train['catego	json(train_path) ory'])		

	ARTS & CULTURE BLACK VOICES BUSINESS COLLEGE COMEDY CRIME CULTURE & ARTS DIVORCE EDUCATION ENTERTAINMENT ENVIRONMENT FIFTY GOOD NEWS HEALTHY LIVING	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	- 800		<pre>even on a third of the data the max iterations of 100 was reached 1 sample_test = df_train.sample(frac=.3) 2 len(sample_test) 15354</pre>
1	HOME & LIVING	-3 0 1 0 1 0 2 0 0 2 5 1 3 12272 0 5 0 0 1 0 0 0 6 4 5 2 2 0 1 1 2 0 -1 3 9 2 2 2 2 0 5 0 5 4 2 7 21191 3 3 1 1 8 2 2 0 0 3 3 1 0 2 4 7 10			from sklearn import preprocessing
labe	MEDIA	-3 4 1 0 6 3 2 0 1 2 0 0 1 0 0 21460 0 4 1 1 1 2 0 0 1 1 0 0 1 0 5 2			rrow oncears import preprocessing
ne	MONEY	0 1 13 2 0 2 0 0 3 0 0 1 2 0 0 2 0 73 0 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1			V train - liet(cample test ada cimilarity values)
Ē	PARENTING				A_train = tist(sample_test.aua_simitarity.vatues)
		-4 7 1 2 0 8 7 1 1 6 0 2 0 2 0 6 1 0 2 3278 1 7 4 1 1 2 0 2 2 1 8 1			scaler = preprocessing.standardScaler().fit(x_train)
	RELIGION	-1 1 0 2 2 1 0 0 0 0 2 0 5 2 0 6 1 0 2 2 21401 2 0 0 0 0 0 0 0 2 1 6	- 400		X_train_scaled = scaler.transform(X_train)
	SCIENCE	-1 1 0 1 0 0 0 0 1 0 11 1 3 4 1 1 1 0 1 2 0 11170 1 0 1 2 0 0 4 6 2 1			
	SPORTS	-071232120600430140034332611120102103			<pre>y_train = sample_test.category</pre>
	STYLE	-2 1 0 1 0 1 4 0 0 0 0 0 0 2 1 2 0 2 0 0 0 0 11119 2 0 0 0 2 0 1 1		8	
	STYLE & BEAUTY	-0122210641412121520310102334 <mark>70</mark> 04551334		q	X test = list(df eval ada similarity values)
	TECH	-0.1801100201010102112002110089105110		10	<pre>scaler = prepresenting StandardScaler() fit(V test)</pre>
	TRAVEL	-3 2 8 1 1 0 6 3 1 1 5 10 3 4 13 6 3 10 3 1 2 5 4 2 0 4 10 3 17 4 10 4 1 6		10	scater = preprocessing.standardscater().iit(x_test)
	WEDDINGS	-000011060002000013203000061302160200	- 200	11	X_test_scaled = scaler.transform(X_test)
	WEIRD NEWS	-1 1 0 0 11 4 1 0 0 2 4 1 4 1 1 1 0 0 0 0 1 0 4 3 1 0 2 2 1 01170 1 1	200	12	
	WELLNESS			13	<pre>y_test = df_eval.category</pre>
	WORLD	-3 0 9 0 1 17 7 0 3 1 11 1 3 1 022 7 0 010 4 13 9 0 0 0 0 5 5 0 6 1 4429		14	
	WORLD			15	<pre>clf = LogisticRegression(C=1e5. solver='saga'. multi class='multinomial'</pre>
				16	clf fit(X train scaled v train)
				17	prode = olf prodict(V test scaled)
			0	1/	preus - ctr.preurct(A_test_scateu)
		Predicted label			

- 1000

/usr/lib/python3.10/site-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge

warnings.warn(

	Solvers						
Penalties	ʻlbfgs'	ʻliblinear'	'newton-cg'	'newton-cholesky'	'sag'	'saga'	
Multinomial + L2 penalty	yes	no	yes	no	yes	yes	
OVR + L2 penalty	yes	yes	yes	yes	yes	yes	
Multinomial + L1 penalty	no	no	no	no	no	yes	
OVR + L1 penalty	no	yes	no	no	no	yes	
Elastic-Net	no	no	no	no	no	yes	
No penalty ('none')	yes	no	yes	yes	yes	yes	
Behaviors							
Penalize the intercept (bad)	no	yes	no	no	no	no	
Faster for large datasets	no	no	no	no	yes	yes	
Robust to unscaled datasets	yes	yes	yes	yes	no	no	

1		precision	recall	f1-score	support
2		-			
3	ARTS & CULTURE	0.37	0.65	0.47	79
4	BLACK VOICES	0.55	0.70	0.62	312
5	BUSINESS	0.60	0.62	0.61	416
6	COLLEGE	0.37	0.69	0.48	62
7	COMEDY	0.59	0.66	0.62	374
8	CRIME	0.56	0.75	0.64	216
9	CULTURE & ARTS	0.50	0.78	0.61	159
10	DIVORCE	0.75	0.82	0.78	238
11	EDUCATION	0.35	0.65	0.45	63
12	ENTERTAINMENT	0.89	0.63	0.74	1104
13	ENVIRONMENT	0.61	0.72	0.66	274
14	FIFTY	0.32	0.62	0.43	96
15	GOOD NEWS	0.30	0.63	0.41	79
16	HEALTHY LIVING	0.52	0.53	0.52	458
17	HOME & LIVING	0.73	0.82	0.77	277
18	IMPACT	0.37	0.55	0.44	216
19	MEDIA	0.48	0.77	0.59	190
20	MONEY	0.42	0.69	0.52	106
21	PARENTING	0.87	0.70	0.78	856

		precision	recall	f1-score	support
0.0		0.00	0.04	0 70	1 5 0 1
22	POLITICS	0.93	0.64	0.76	1581
23	QUEER VOICES	0.81	0.78	0.79	418
24	RELIGION	0.56	0.77	0.65	181
25	SCIENCE	0.48	0.71	0.57	164
26	SPORTS	0.75	0.81	0.78	321
27	STYLE	0.49	0.78	0.60	143
28	STYLE & BEAUTY	0.92	0.82	0.87	694
29	TASTE	0.89	0.85	0.87	561
30	TECH	0.47	0.72	0.57	123
31	TRAVEL	0.89	0.79	0.84	656
32	WEDDINGS	0.80	0.87	0.83	247
33	WEIRD NEWS	0.45	0.71	0.55	165
34	WELLNESS	0.86	0.60	0.71	1183
35	WOMEN	0.44	0.59	0.51	218
36	WORLD	0.80	0.75	0.77	565
37					
38	accuracy			0.70	12795
39	macro avg	0.61	0.71	0.64	12795
40	weighted avg	0.75	0.70	0.71	12795
11					

- df_train_sample = df_train.sample(500)
- df_train_sample["text"] = df_train_sample.text.apply(lambda x: " ".join(x[0:50].split(" ")[0:-1]))
- df_train_sample.to_json("df_train_sentiment_ada_embeddings_sample_5.json")
- df_train_sample["ada_similarity"] = df_train_sample.text.apply(lambda x: get_embedding(x, engine='text-similarity-ada-001'))
- df_train_sample
- $|\langle \langle 1-10 \vee \rangle \rangle|$ 500 rows \times 3 columns

text		sentiment		ada_similarity
Time has not been kind to this movie. Once		positive		[0.030108904466032982, 0.004708367399871349, 0.01
First i gotta say that this film is way less		positive		[0.012335612438619137, -0.008514531888067722, -0
It seems the makers of this film had trouble		negative		[0.02407647855579853, 0.01442974153906107, -0.019
'Moonstruck' is a love story. There is not one		positive		[0.03885204344987869, -0.0019428624073043466, 0.0
I have seen this movie 4 times in 5 months and i		positive		[-0.014194046147167683, 0.011614245362579823, 0.0
The thing with Ali G is that he takes the mick		positive		[-0.0037947550881654024, -0.013258409686386585,
I disagree strongly with anyone who might dismiss		positive		[0.014401664026081562, 0.027736539021134377, 0.00
As I reach the "backside" of 35 I find myself		positive		[0.030719023197889328, 0.03126721456646919, -0.01
This video guide was the masterpiece of the year		negative		[-0.020596014335751534, 0.020832043141126633, -0
I found this a good movie to pass your time, but		negative		[0.02685760334134102, -0.001097695785574615, -0.0
	text Time has not been kind to this movie. Once First i gotta say that this film is way less It seems the makers of this film had trouble 'Moonstruck' is a love story. There is not one I have seen this movie 4 times in 5 months and i The thing with Ali G is that he takes the mick I disagree strongly with anyone who might dismiss As I reach the "backside" of 35 I find myself This video guide was the masterpiece of the year I found this a good movie to pass your time, but	<pre>text Time has not been kind to this movie. Once First i gotta say that this film is way less It seems the makers of this film had trouble 'Moonstruck' is a love story. There is not one I have seen this movie 4 times in 5 months and i The thing with Ali G is that he takes the mick I disagree strongly with anyone who might dismiss As I reach the "backside" of 35 I find myself This video guide was the masterpiece of the year I found this a good movie to pass your time, but</pre>	text‡ sentimentTime has not been kind to this movie. OncepositiveFirst i gotta say that this film is way lesspositiveIt seems the makers of this film had troublenegative'Moonstruck' is a love story. There is not onepositiveI have seen this movie 4 times in 5 months and ipositiveThe thing with Ali G is that he takes the mickpositiveI disagree strongly with anyone who might dismisspositiveAs I reach the "backside" of 35 I find myselfpositiveThis video guide was the masterpiece of the yearnegativeI found this a good movie to pass your time, butnegative	text* sentiment*Time has not been kind to this movie. OncepositiveFirst i gotta say that this film is way lesspositiveIt seems the makers of this film had troublenegative'Moonstruck' is a love story. There is not onepositiveI have seen this movie 4 times in 5 months and ipositiveThe thing with Ali G is that he takes the mickpositiveI disagree strongly with anyone who might dismisspositiveAs I reach the "backside" of 35 I find myselfpositiveThis video guide was the masterpiece of the yearnegativeI found this a good movie to pass your time, butnegative



sampling and truncating

test sample 500 eval sample 200

Random Fore	st	precision	recall	f1-score	support
dataset too small 😕	negative	0.69	0.68	0.69	101
	positive	0.68	0.69	0.68	99
	accuracy			0.69	200
	macro avg	0.69	0.69	0.68	200
wei	ighted avg	0.69	0.69	0.69	200



precision recall ti-score suppor

	negative	0.58	0.61	0.60	101
<pre>clf = LogisticRegression (C=1e)</pre>	5,				
<pre>solver='lbfgs')</pre>	positive	0.58	0.55	0.56	99
Logistic Regression performed					
worse 🙁	accuracy			0.58	200
	macro avg	0.58	0.58	0.58	200
	weighted avg	0.58	0.58	0.58	200

Implementation

Language classification

	precision	recall	f1-score	support
da	1.00	1.00	1.00	838
de	1.00	1.00	1.00	1443
en	1.00	1.00	1.00	2517
fr	0.97	1.00	0.99	36
it	0.91	0.94	0.92	31
unknown	0.73	0.50	0.59	16
accuracy			1.00	4881
macro avg	0.93	0.91	0.92	4881
weighted avg	1.00	1.00	1.00	4881



Implementation

Author classification

	precision	recall	f1-score	support
Franz Kafka	0.90	0.93	0.92	280
Friedrich Schiller	0.74	0.74	0.74	266
Henrik Ibsen	1.00	1.00	1.00	897
James Joyce	0.93	0.95	0.94	682
Johann Wolfgang von Goethe	0.68	0.68	0.68	228
Virginia Woolf	0.98	0.97	0.98	1901
Wilhelm Busch	0.95	0.95	0.95	627
accuracy			0.94	4881
macro avg	0.88	0.89	0.89	4881
weighted avg	0.94	0.94	0.94	4881



True label

- 350

- 300

- 250

- 200

- 150

- 100

- 50

Conclusion

	0-shot		1-shot		Few-shot	
Task	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B
TriviaQA (EM)	71.3^{a}	76.9	75.8 ^a	81.4	75.8 ^a (1)	81.4 (1)
Natural Questions (EM)	24.7 ^a	21.2	26.3^{a}	29.3	32.5^{a} (1)	39.6 (64)
Web Questions (EM)	19.0^a	10.6	25.3^{b}	22.6	41.1^{b} (64)	43.5 (64)
Lambada (EM)	77.7 ^f	77.9	80.9^{a}	81.8	87.2^{c} (15)	89.7 (8)
HellaSwag	80.8 ^f	83.4	80.2^{c}	83.6	82.4° (20)	83.8 (5)
StoryCloze	83.2^{b}	84.6	84.7^{b}	86.1	87.7 ^b (70)	89.0 (5)
Winograd	88.3 ^b	90.1	89.7 ^b	87.5	88.6 ^a (2)	89.4 (5)
Winogrande	74.9^{f}	81.1	73.7^{c}	83.7	79.2 ^a (16)	85.1 (5)
Drop (F1)	57.3^{a}	69.4	57.8^{a}	70.8	58.6^{a} (2)	70.8 (1)
CoQA (F1)	81.5 ^b	77.6	84.0 ^b	79.9	85.0 ^b (5)	81.5 (5)
QuAC (F1)	41.5^{b}	45.2	43.4^{b}	47.7	44.3 ^b (5)	47.7 (1)
SQuADv2 (F1)	71.1^{a}	80.8	71.8^{a}	82.9	71.8^{a} (10)	83.3 (5)
SQuADv2 (EM)	64.7^{a}	75.5	66.5^{a}	78.7	67.0^{a} (10)	79.6 (5)
RACE-m	64.0^{a}	68.1	65.6^{a}	69.3	66.9 ^{a†} (8)	72.1 (8)
RACE-h	47.9^{c}	49.1	48.7^a	52.1	49.3 ^{a†} (2)	54.6 (5)
PIQA	82.0^{c}	82.3	81.4^{a}	83.9	83.2^{c} (5)	85.2 (5)
ARC-e	76.4^{e}	76.6	76.6 ^a	85.0	80.9 ^e (10)	88.4 (5)
ARC-c	51.4^{b}	53.0	53.2^{b}	60.1	52.0 ^a (3)	65.9 (5)
OpenbookQA	57.6^{b}	53.4	55.8^b	53.6	65.4^{b} (100)	68.0 (32)
BoolQ	83.7^{f}	88.0	82.8^{a}	88.7	84.8 ^c (32)	89.1 (8)
Copa	91.0^{b}	93.0	92.0 ^a	91.0	93.0^{a} (16)	95.0 (5)
RTE	73.3^{e}	72.9	71.5^{a}	78.7	76.8 (5)	81.2 (5)
WiC	50.3^{a}	59.1	52.7^{a}	63.2	58.5° (32)	64.6 (5)
Multirc (F1a)	73.7^{a}	83.5	74.7^{a}	84.9	77.5 ^a (4)	86.3 (5)
WSC	85.3^{a}	89.1	83.9^{a}	86.3	85.6 ^a (2)	89.5 (5)
ReCoRD	90.3^{a}	92.9	90.3^{a}	92.8	90.6 (2)	92.9 (2)
CB	48.2^{a}	51.8	73.2^{a}	83.9	84.8 ^a (8)	89.3 (5)
ANLI R1	39.2^a	48.4	42.4^a	52.6	44.3 ^a (2)	56.9 (5)
ANLI R2	39.9^{e}	44.2	40.0^{a}	48.7	41.2^{a} (10)	56.1 (5)
ANLI R3	41.3^{a}	45.7	40.8^{a}	52.3	44.7^{a} (4)	51.2 (5)

Model	Avg NLG	Avg NLU
GPT-3 175B	52.9	65.4
GLaM 64B/64E	58.4	68.7
PaLM 8B	41.5	59.2
PaLM 62B	57.7	67.3
PaLM 540B	63.9	74.7

Google Natural Language API and the v2 classification model

- Under the hood: PaLM, LaMDA and T5
- PaLM on its own still a decoder-only model (similar to GPT-3)
- Good classification possible with a combination of methods and models
- Not free 🛸
- Only implementation available

Future work

- Classification tasks with help of the Google Cloud NLP API
 - 34 categories in our news dataset vs. 1000+ categories in the v2 model
 - $\circ~$ a surjective mapping of the v2 categories to the original categories C2 \rightarrow C1
 - manual classification report

References

- 1. Google research. (2022). Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance. <u>https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html</u>
- 2. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., ... & Fiedel, N. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- 3. Neelakantan, A., Xu, T., Puri, R., Radford, A., Han, J. M., Tworek, J., ... & Weng, L. (2022). Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.
- 4. OpenAI-Cookbook. <u>https://github.com/openai/openai-cookbook</u>
- 5. Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. CoRR, abs/2104.09864, 2021. URL <u>https://arxiv.org/abs/2104.09864</u>.
- 6. Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
- 7. Barham, P., Chowdhery, A., Dean, J., Ghemawat, S., Hand, S., Hurt, D., ... & Wu, Y. (2022). Pathways: Asynchronous distributed dataflow for ml. *Proceedings of Machine Learning and Systems*, *4*, 430-449.
- 8. Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.