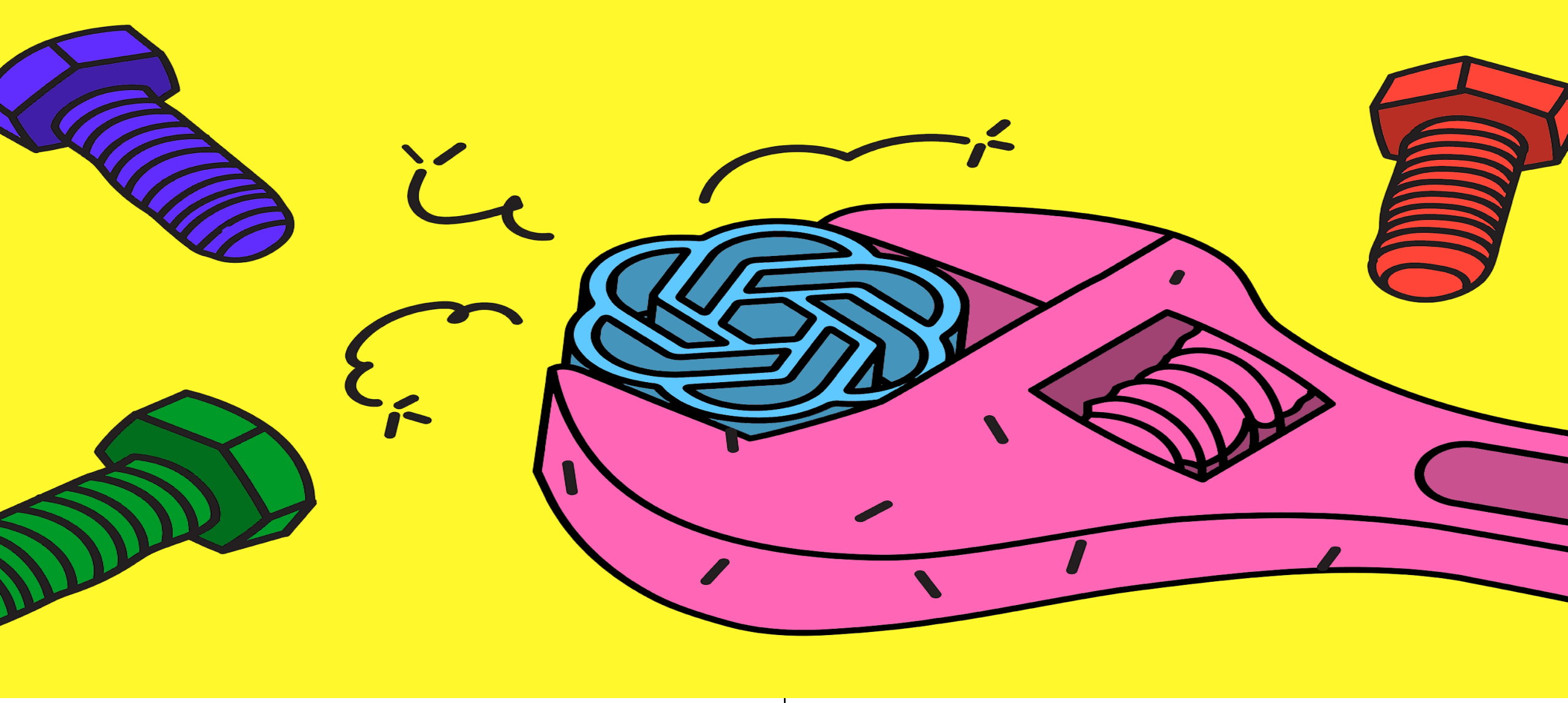
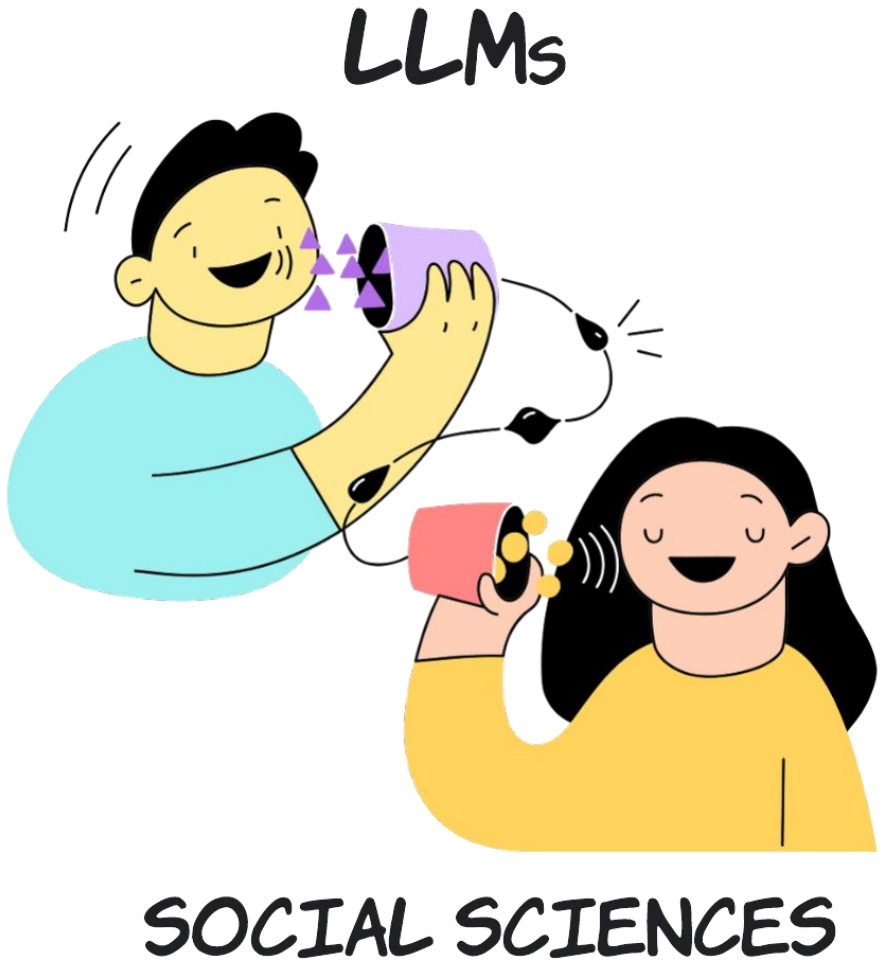


Large Language Models in Social Science Research

Professor Jonathan Colner

Plan for today

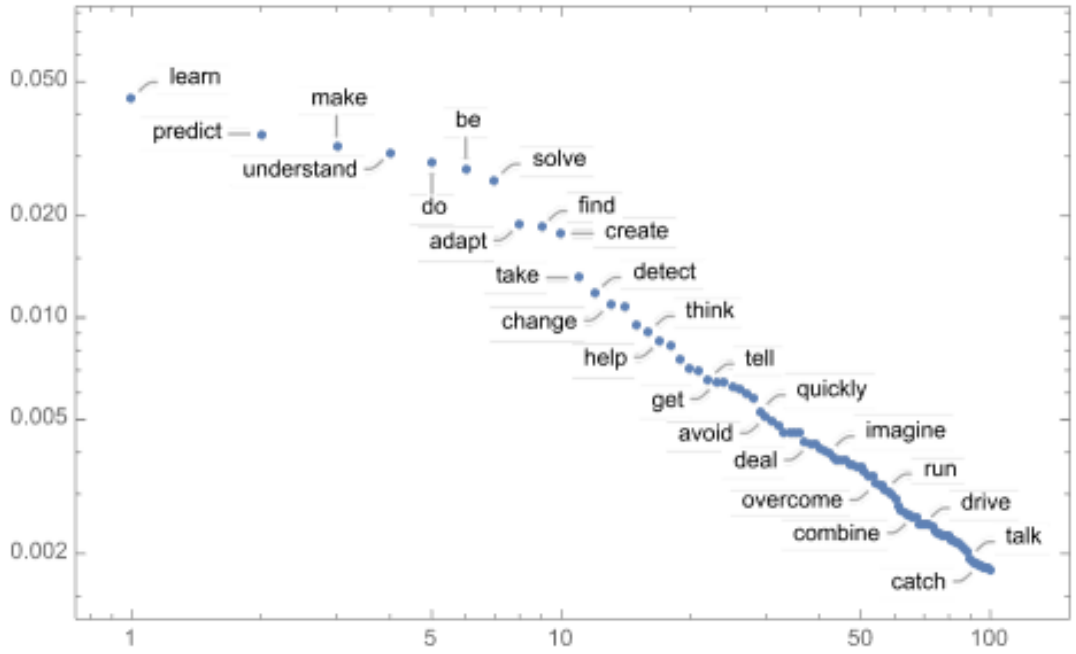
1. What are LLMs anyway?
2. How can we use them in Social Science Research?
3. Basic examples with code



What are LLMs anyway?

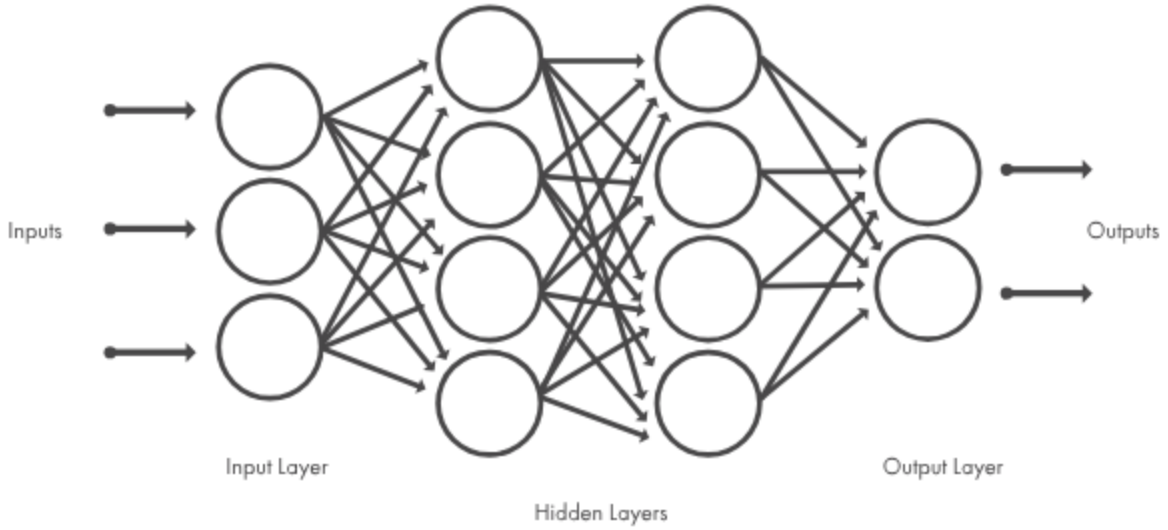
- At its most basic… auto-complete

"The best thing about AI is its ability to…"

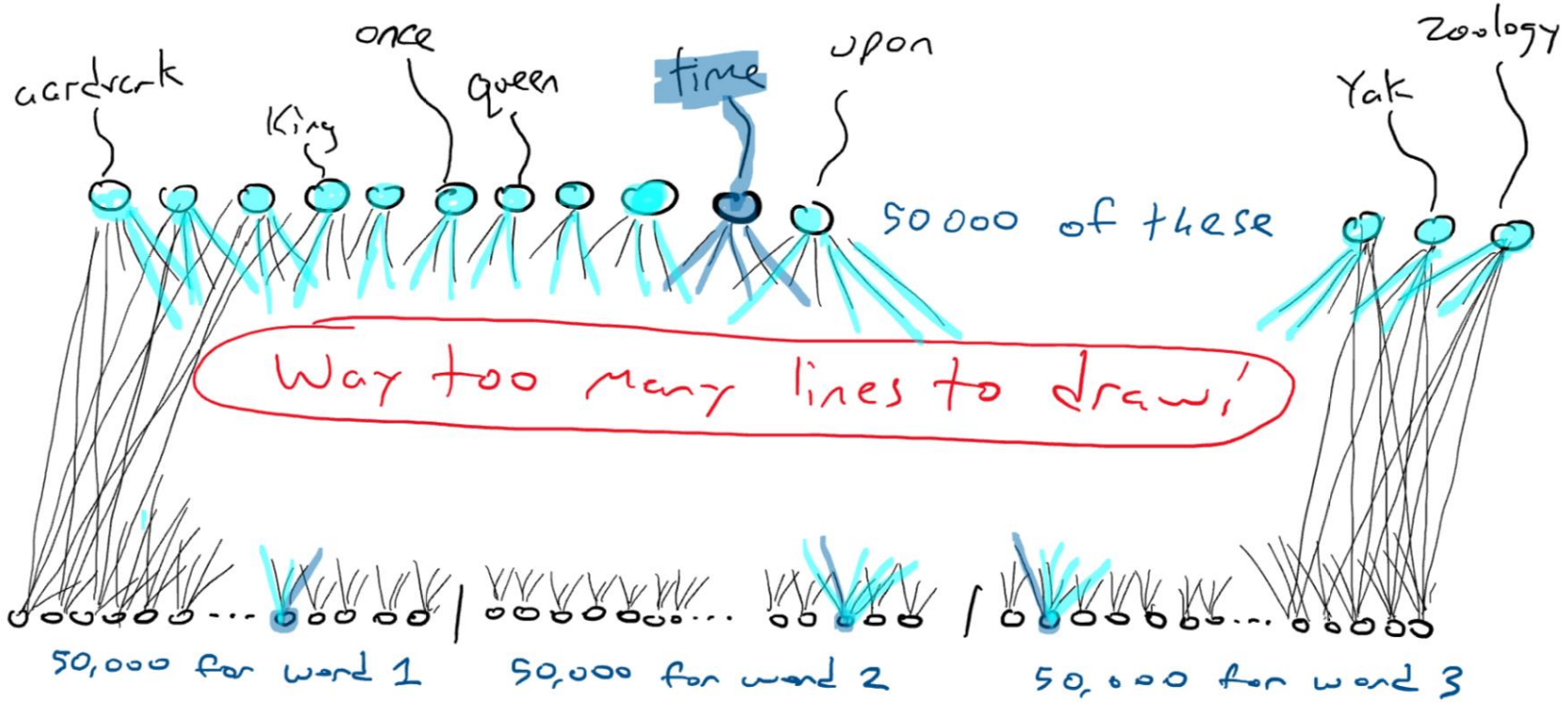


Neural Networks and LLMs

- Neural nets are roughly based on the human brain
- Individual neurons take inputs, transform them, then pass them along to connected neurons



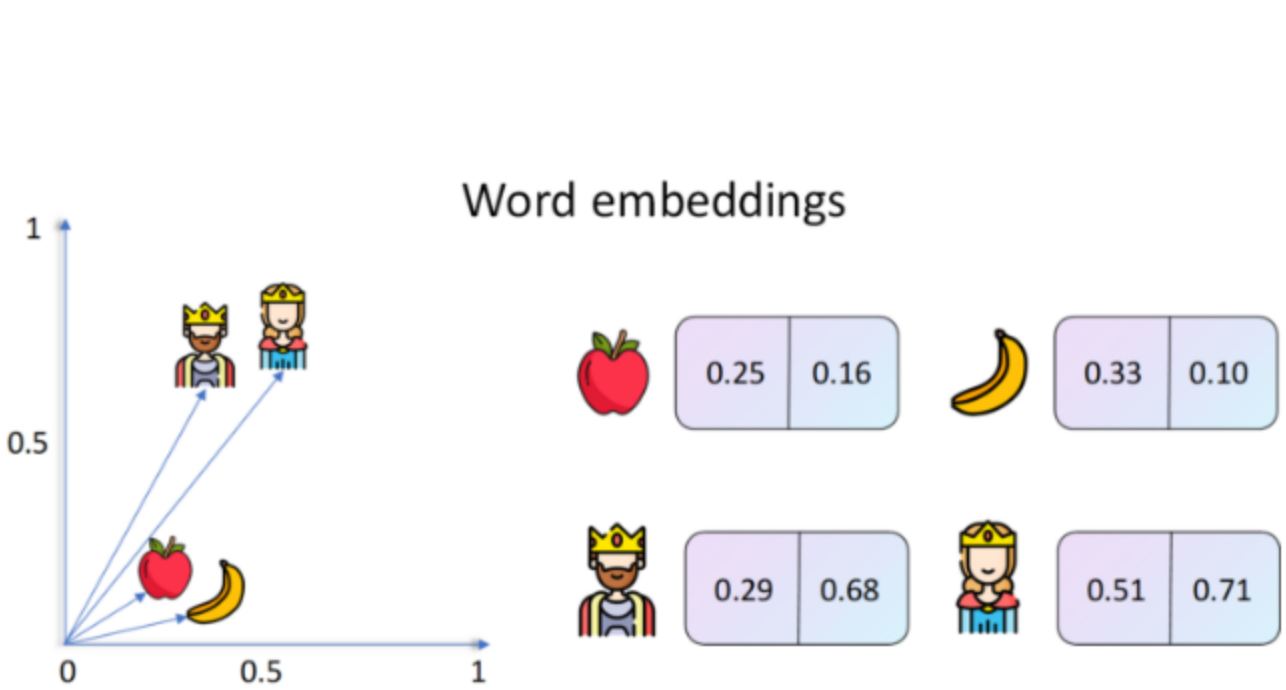
Once upon a…



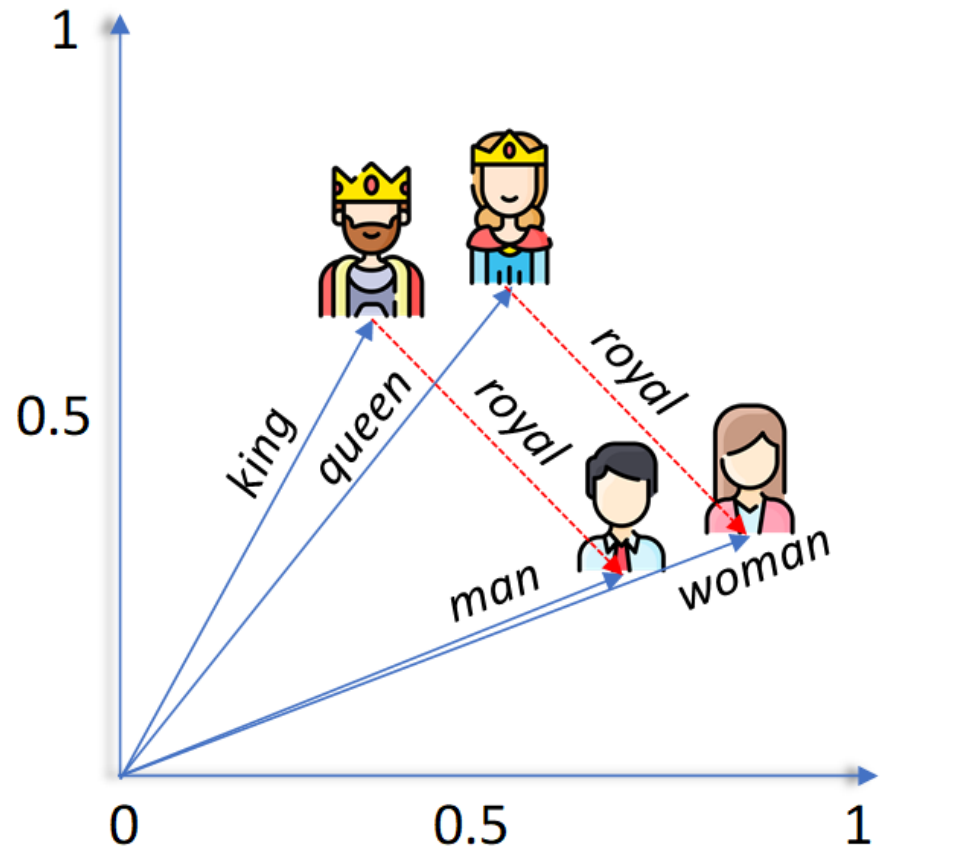
- This becomes extremely computationally intensive incredibly fast

Word and Text Embeddings

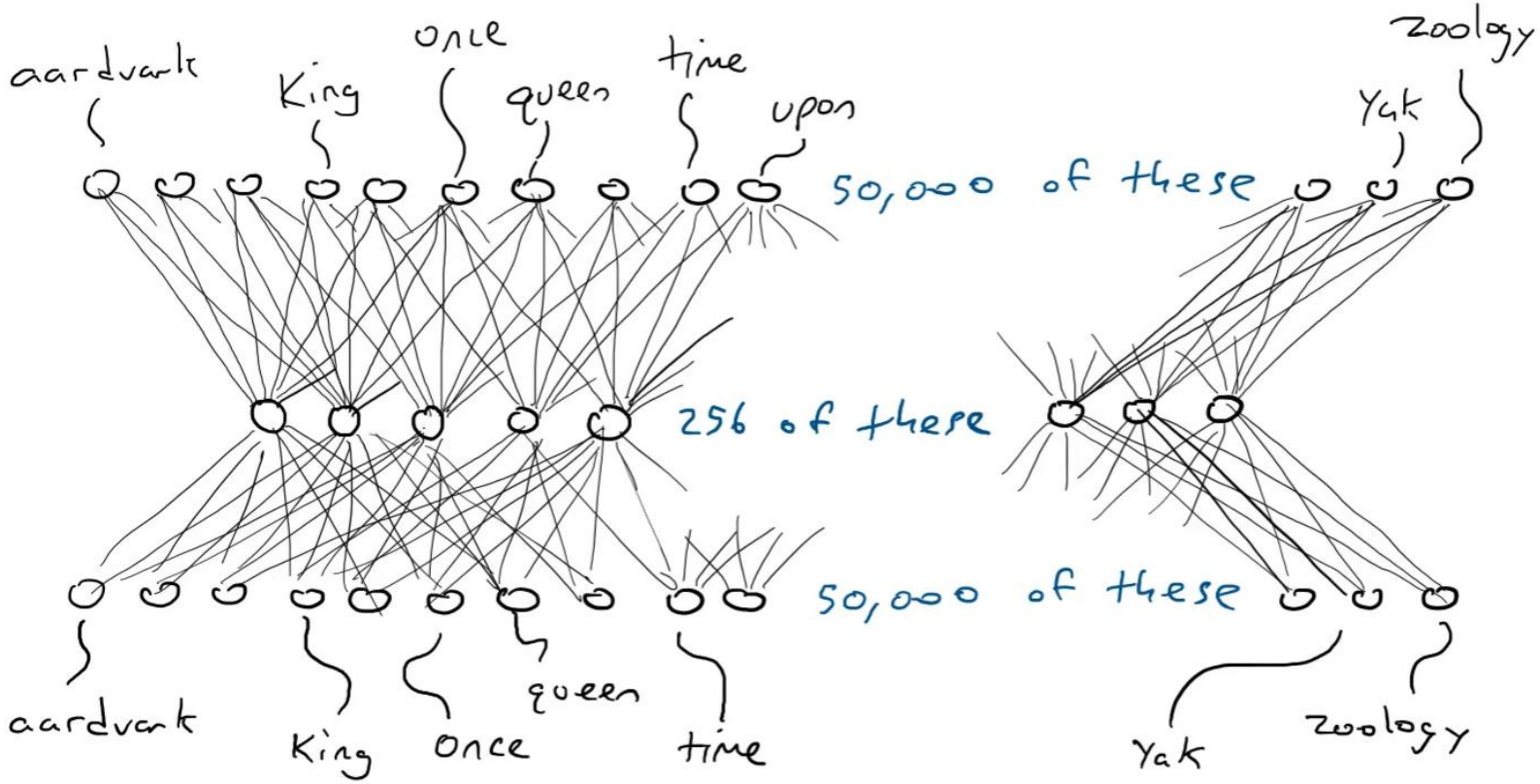
- Word embeddings convert text into numerical vectors that represent that text
- Extremely high dimensional space
- Placing words that have semantic similarity closer together
- The number of dimensions can vary, but often range from ~300 to 1,000+



2 dimensional word embedding representation of our example words

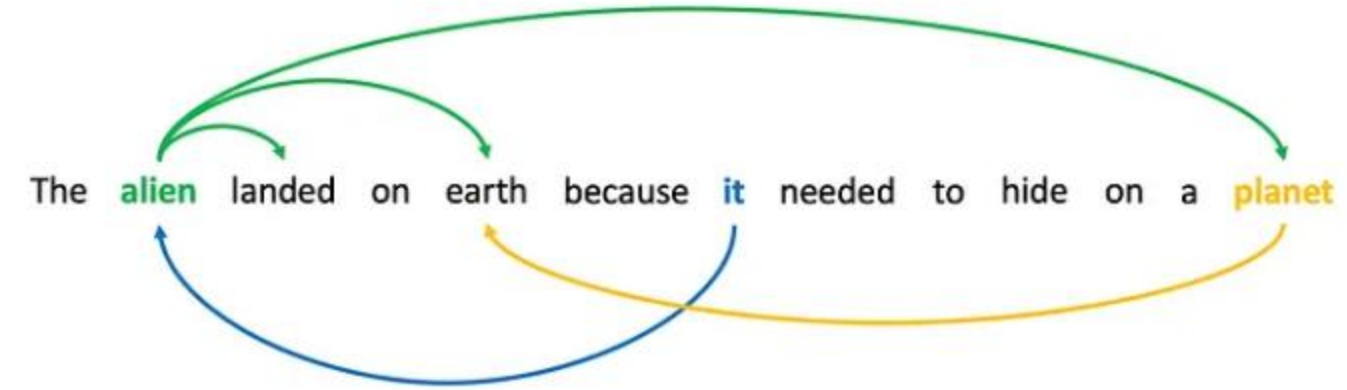


Operations with word embeddings. The king, queen, man, woman example is very popular for illustrating this.

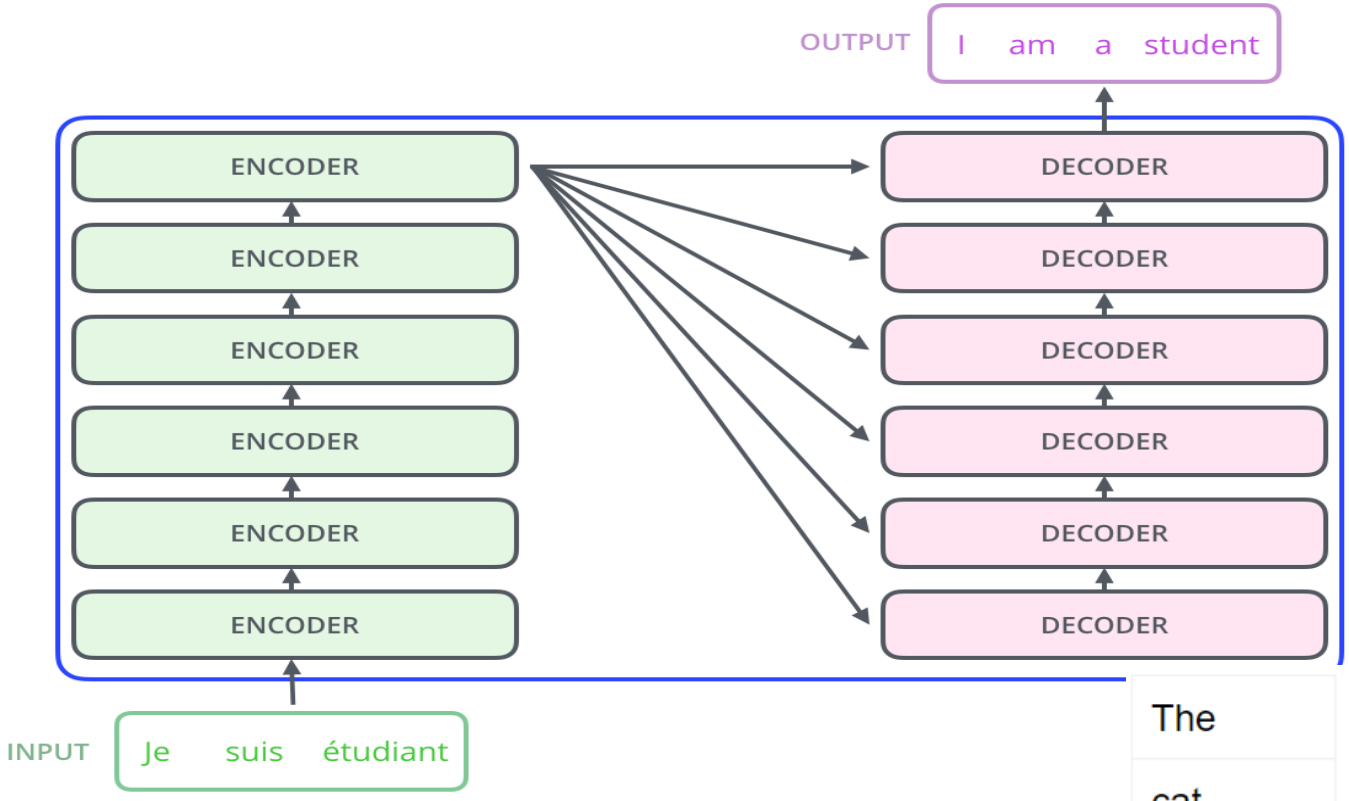

aardvark
King
once
queen
time
upon
zoology
Yak
50,000 of these
256 of these
50,000 of these
aardvark
King
once
queen
time
Yak
zoology

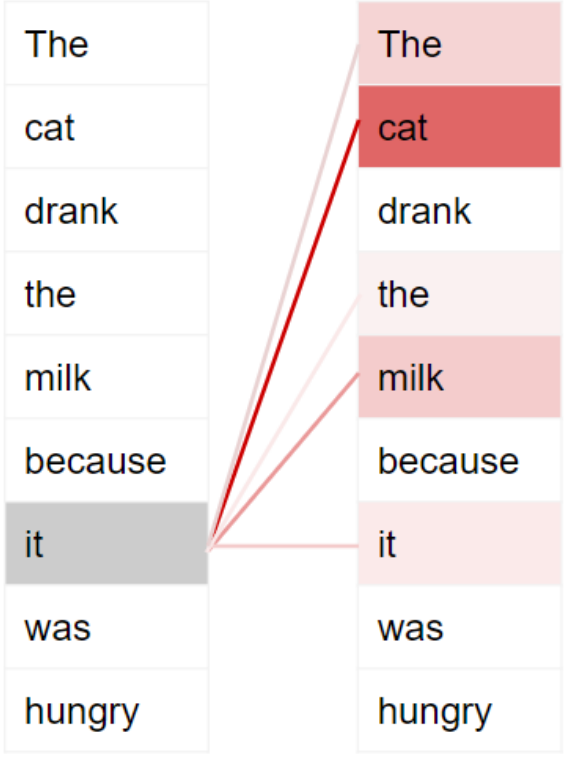
Transformers and Self-Attention

- Transformers make it much easier to predict the next word
- They identify how words in a sequence of words are related to each other.

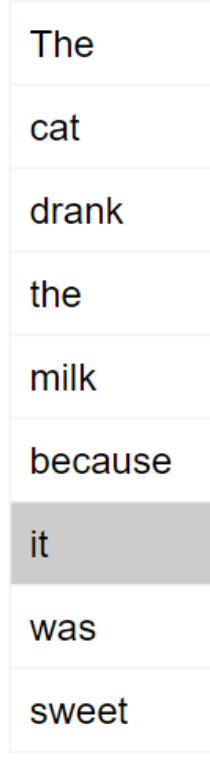


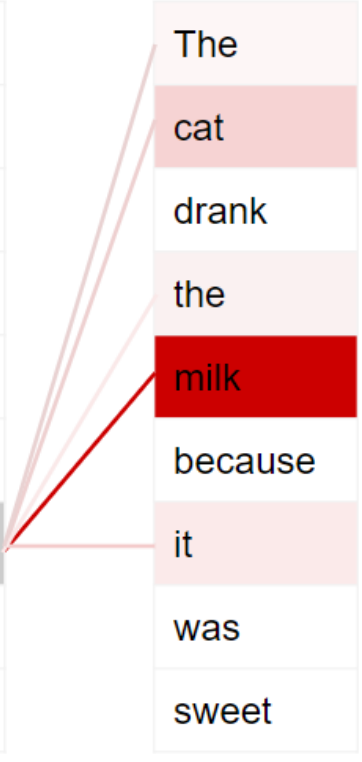
Words are related to other words by function, by referring to the same thing, or by informing the meanings of each other.


OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant


The
cat
drank
the
milk
because
it
was
hungry

The
cat
drank
the
milk
because
it
was
hungry


The
cat
drank
the
milk
because
it
was
sweet


The
cat
drank
the
milk
because
it
was
sweet

How do we train these models?

- BERT (Bidirectional Encoder Representations from Transformers)
  - Learns to predict masked words in a sentence
  - Learns to predict whether two sentences appear consecutively or not

- Massive amounts of unlabeled text data
  - All English Wikipedia pages and 11,000+ books from the BookCorpus dataset

- BERT-base has 110 million parameters
  - Parameters are the weights and biases connecting nodes

Main Takeaways

- LLMs are very good at predicting and generating plausible language in response to other language

- They are based on an extremely large amount of training data

More Resources

- What Is ChatGPT Doing … and Why Does It Work?
- A Very Gentle Introduction to Large Language Models without the Hype
- Deep Learning for NLP: Word Embeddings

How can we use them in Social Science Research?

- Modeling Human Behavior
- Simulating Social Relationships
- Interacting with Human Agents
- Text Annotation and Text/Metadata Extraction

(Not including the study of LLMs)

Modeling Human Behavior

- Homo Silicus or Silicon Parrot?
- Test by recreating results from experiments real human participants*
- Some debate over how well it works

Synthetic Replacements for Human Survey Data? The Perils of Large Language Models

Published online by Cambridge University Press: **17 May 2024**

James Bisbee, Joshua D. Clinton, Cassy Dorff, Brenton Kenkel and Jennifer M. Larson

Show author details

"It is [YEAR]. You are a [AGE] year-old, [MARST], [RACETH] [GENDER] with [EDUCATION] making [INCOME] per year, living in the United States. You are [IDEO], [REGIS] [PID] who [INTEREST] pays attention to what's going on in government and politics."[13]

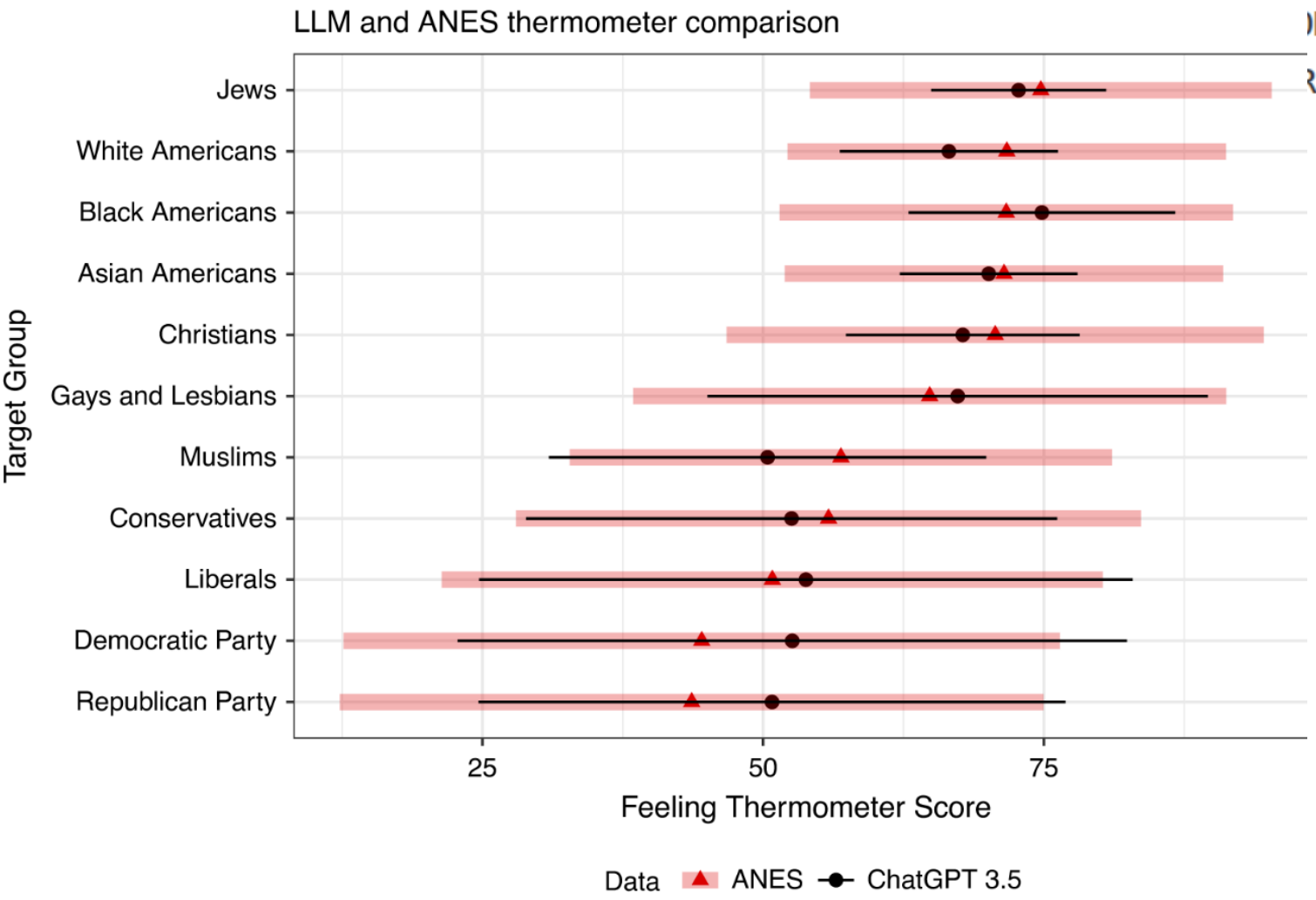
Synthetic Replacements for Human Survey Data? The Perils of Large Language Models

Published online by Cambridge University Press: **17 May 2024**

James Bisbee, Joshua D. Clinton, Cassy Dorff, Brenton Kenkel and Jennifer M. Larson

Show author details

"It is [YEAR]. You are a [AGE] year-old, [MARST], [RACETH] [GENDER] with
ME] per year, living in the United States. You are [IDEO],
EST] pays attention to what's going on in government



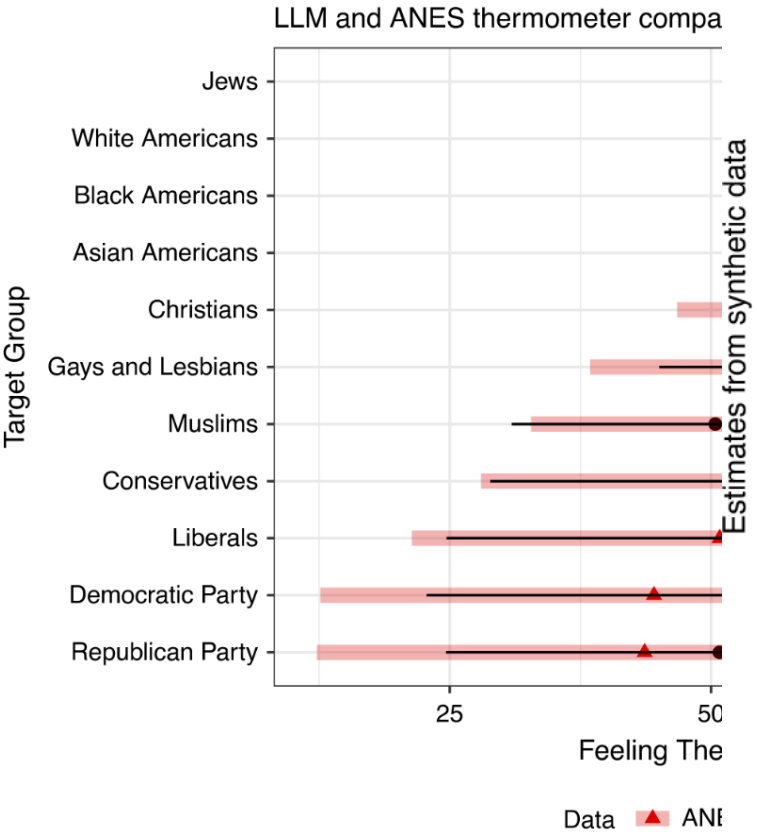
Synthetic Replacements for Human Survey Data? The Perils of Large Language Models

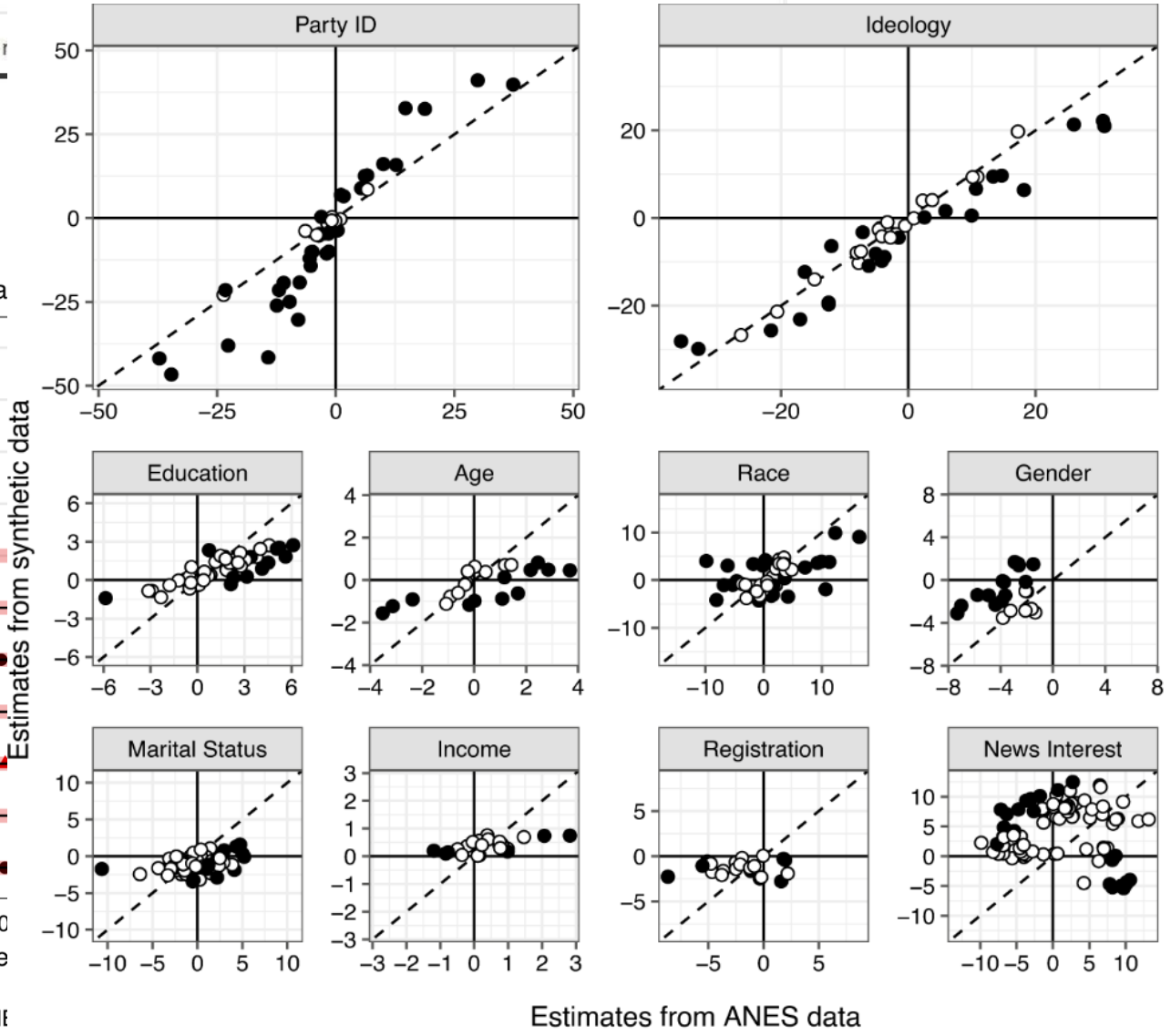
Published online by Cambridge University Press: **17 May 2024**

James Bisbee, Joshua D. Clinton, Cassy Dorff, Brenton Kenkel and Jennifer M. Larson

Show author details

Article Figures Supplementary





R] with
ou are [IDEO],
government

Potential pitfalls and things to keep in mind

- May work best for American contexts
- There is significant doubt in academic community
  - How can you validate it?
- Must carefully defend use of model and prompt
  - Newer models may be better but more opaque

Simulating Social Relationships

- Simulate entire communities and societies
- Use simulations to better understand societal rules
- how social media shapes human behavior or how new moderation rules could help or hurt users

Generative Agents: Interactive Simulacra of Human Behavior

Joon Sung Park
Stanford University
Stanford, USA
joonspk@stanford.edu

Joseph C. O'Brien
Stanford University
Stanford, USA

Carrie J. Cai
Google Research
Mountain View, CA, USA

Meredith Ringel Morris
Google DeepMind
Seattle, WA, USA
merrie@google.com

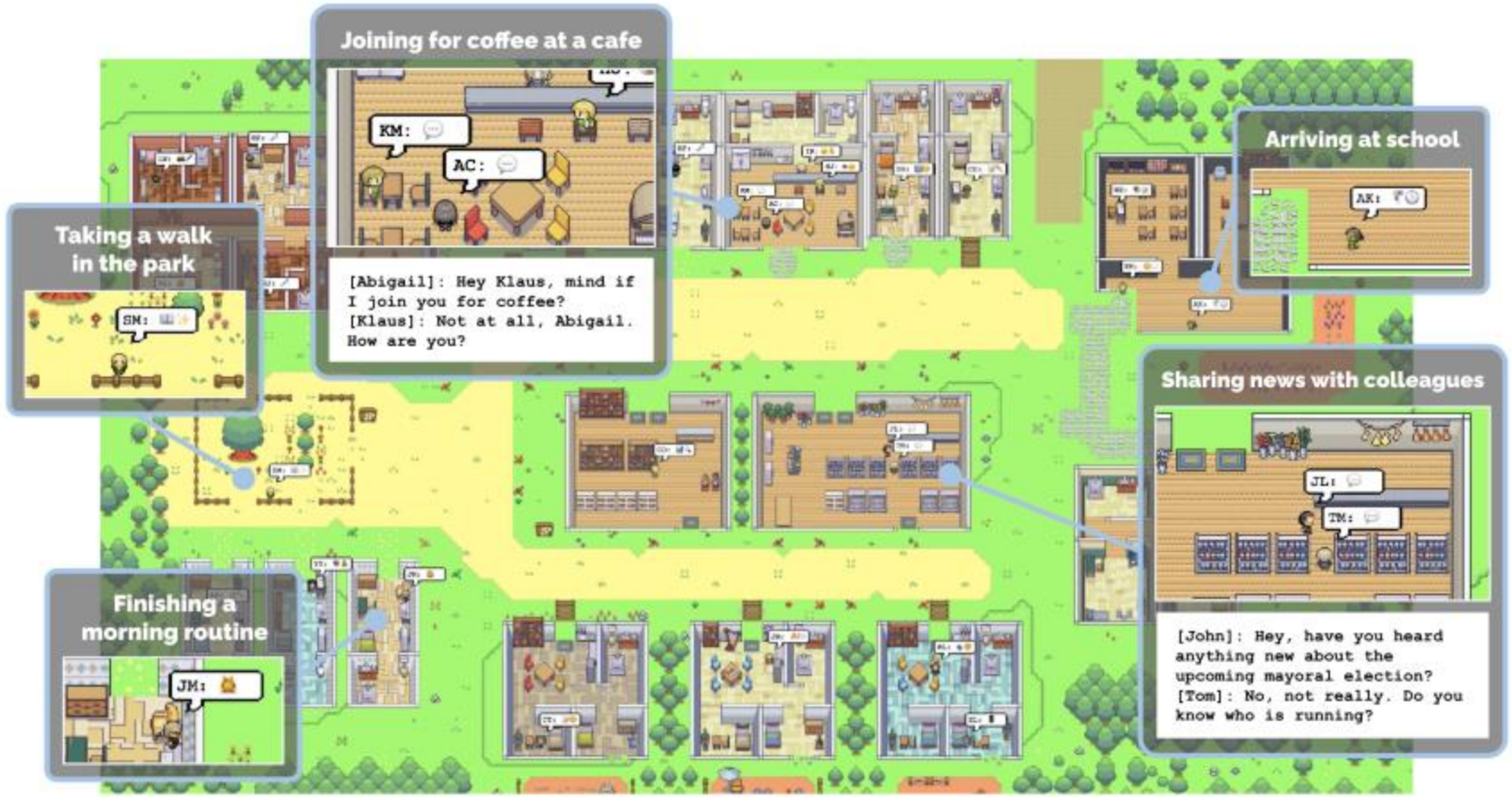


Figure 1: Generative agents are believable simulacra of human behavior for interactive applications. In this work, we demonstrate generative agents by populating a sandbox environment, reminiscent of The Sims, with twenty-five agents. Users can observe and intervene as agents plan their days, share news, form relationships, and coordinate group activities.

Interacting with Human Agents

- Can AI conduct semi-structured interviews with real respondents?

- Benefits
  - Could increase sample sizes for studies
  - Allow graduate students to carry out more robust interview-based research

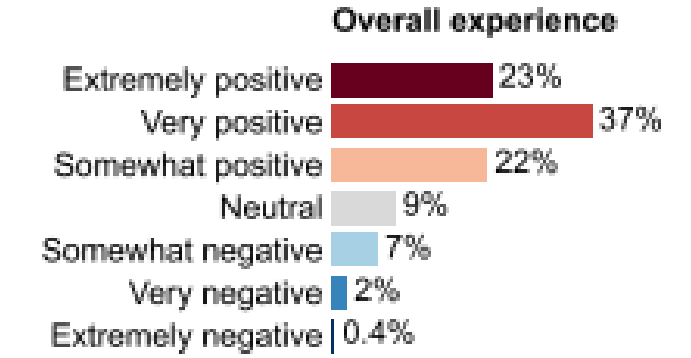
CESifo WORKING PAPERS

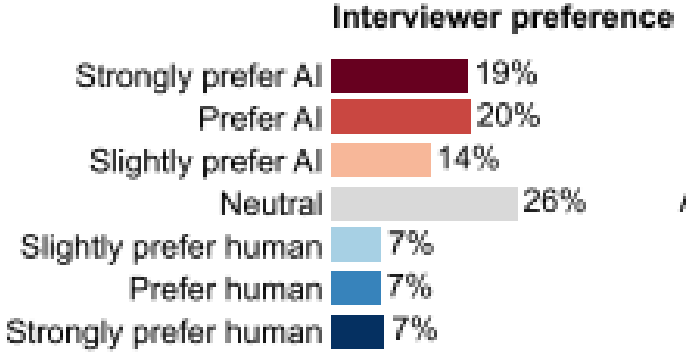
10666
2023

Original Version: September 2023
This Version: September 2024

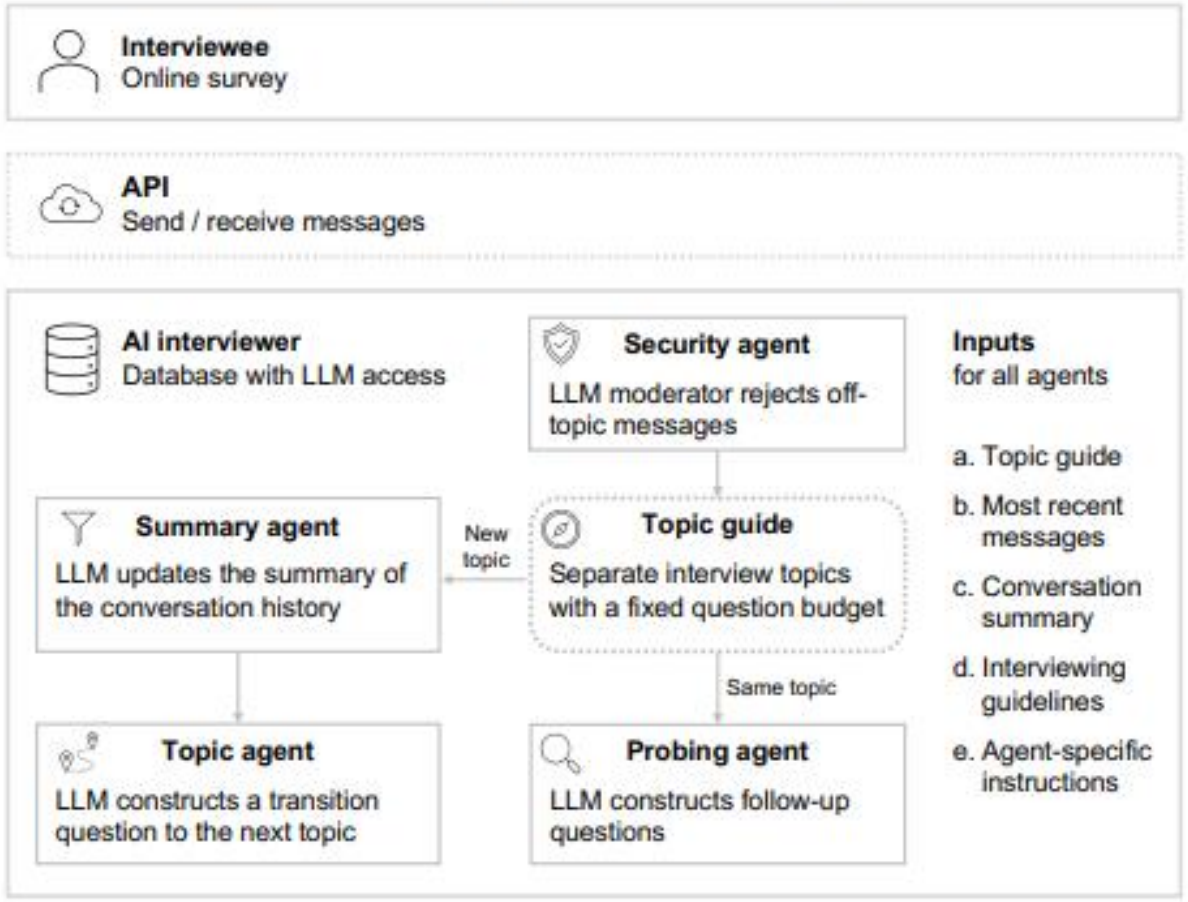
Conducting Qualitative Interviews with AI

Felix Chopra, Ingar Haaland





(b) Workflow to generate an interview question



Note: Panel A provides a screenshot of the chat interface used to conduct the interviews during our main survey. Panel B provides an overview of the workflow of how the AI interviewer generates the next interview question. For a detailed description see Section 2.3.

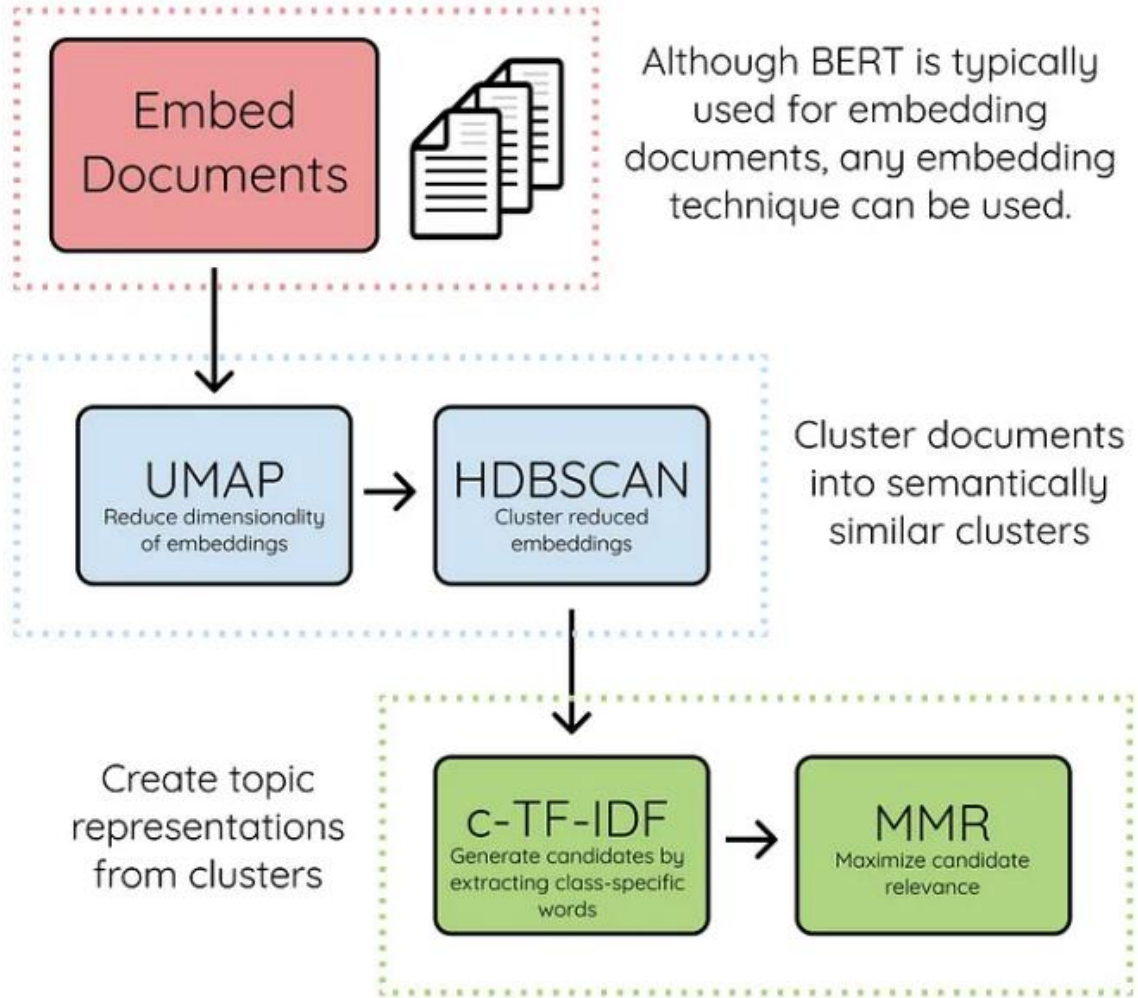
Text Annotation and Text/Metadata Extraction

- Significant overlap
- Most common use of LLMs in our field
- Includes topic modeling, sentiment analysis, data extraction
- Can be used upstream to create a large training dataset

BERTopic: Neural topic modeling with a class-based TF-IDF procedure

Maarten Grootendorst
maartengrootendorst@gmail.com

	20 NewsGroups		BBC News		Trump	
	TC	TD	TC	TD	TC	TD
LDA	.058	.749	.014	.577	-.011	.502
NMF	.089	.663	.012	.549	.009	.379
T2V-*MPNET*	.068	.718	-.027	.540	-.213	.698
T2V-*Doc2Vec*	.192	.823	.171	.792	-.169	.658
CTM	.096	.886	.094	.819	.009	.855
BERTopic-*MPNET*	.166	.851	.167	.794	.066	.663



BERTopic algorithm

Basic Examples with Code (Python)

- BERTopic example
- Finetuning an LLM example
- One-shot approach example

Example Data

- https://huggingface.co/datasets/SetFit/bbc-news

- Record of 2,225 articles published by the BBC between 2004 and 2005

- Each record is labeled as one of five categories:
  - Business, Entertainment, Politics, Sport, or Tech

Example 1: BERTopic

```python
from datasets import load_dataset
import pandas as pd

dataset = load_dataset("SetFit/bbc-news")
df = pd.DataFrame(dataset["train"])

df.head()
```

	text	label	label_text
0	wales want rugby league training wales could f...	2	sport
1	china aviation seeks rescue deal scandal-hit j...	1	business
2	rock band u2 break ticket record u2 have smash...	3	entertainment
3	markets signal brazilian recovery the brazilia...	1	business
4	tough rules for ringtone sellers firms that fl...	0	tech

```python
import hdbscan
from bertopic import BERTopic

df.columns
docs = df["text"].tolist()

hdbscan_model = hdbscan.HDBSCAN(
    min_cluster_size=70,      # ↑ bigger = fewer topics
    min_samples=2,
    metric="euclidean",
    cluster_selection_method="eom"
)

topic_model = BERTopic(
    embedding_model="all-MiniLM-L6-v2",
    hdbscan_model=hdbscan_model,
    verbose=True
)

topics, probs = topic_model.fit_transform(docs)
```

- Min_cluster_size: Smallest number of documents that can make up a separate topic
- Min_samples: determines how strict the cluster formation process is.
- Metric: Method for calculating distance in high dimensional space
- Cluster_selection_metric: Selection method for identifying clusters

https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

```python
import hdbscan
from bertopic import BERTopic

df.columns
docs = df["text"].tolist()

hdbscan_model = hdbscan.HDBSC
    min_cluster_size=70,
    min_samples=2,
    metric="euclidean",
    cluster_selection_method=
)

topic_model = BERTopic(
    embedding_model="all-Mini
    hdbscan_model=hdbscan_mod
    verbose=True
)

topics, probs = topic_model.f
```

```
2026-01-06 20:58:47,640 - BERTopic - Embedding - Transforming documents to embeddings.
modules.json: 100% 349/349 [00:00<00:00, 39.2kB/s]
config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 6.96kB/s]
README.md: 10.5k/? [00:00<00:00, 999kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 6.80kB/s]
config.json: 100% 612/612 [00:00<00:00, 74.2kB/s]
model.safetensors: 100% 90.9M/90.9M [00:02<00:00, 37.8MB/s]
tokenizer_config.json: 100% 350/350 [00:00<00:00, 22.8kB/s]
vocab.txt: 232k/? [00:00<00:00, 7.61MB/s]
tokenizer.json: 466k/? [00:00<00:00, 18.3MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 6.85kB/s]
config.json: 100% 190/190 [00:00<00:00, 19.9kB/s]
Batches: 100% 39/39 [00:04<00:00, 13.32it/s]
2026-01-06 20:58:57,647 - BERTopic - Embedding - Completed ✓
2026-01-06 20:58:57,648 - BERTopic - Dimensionality - Fitting the dimensionality reduction algorithm
2026-01-06 20:58:57,735 - BERTopic - Dimensionality - Completed ✓
2026-01-06 20:58:57,738 - BERTopic - Cluster - Start clustering the reduced embeddings
2026-01-06 20:58:57,827 - BERTopic - Cluster - Completed ✓
2026-01-06 20:58:57,834 - BERTopic - Representation - Fine-tuning topics using representation models.
2026-01-06 20:58:58,157 - BERTopic - Representation - Completed ✓
```

```python
df["bertopic_topic"] = topics
pd.crosstab(df["label_text"], df["bertopic_topic"])
```

bertopic_topic	-1	0	1	2	3
label_text					
business	46	7	0	223	10
entertainment	58	151	0	0	1
politics	35	3	0	3	201
sport	31	0	244	0	0
tech	48	159	0	5	0

BERTopic Pros and Cons

- Pros
  - Easy-to-use code
  - Highly customizable

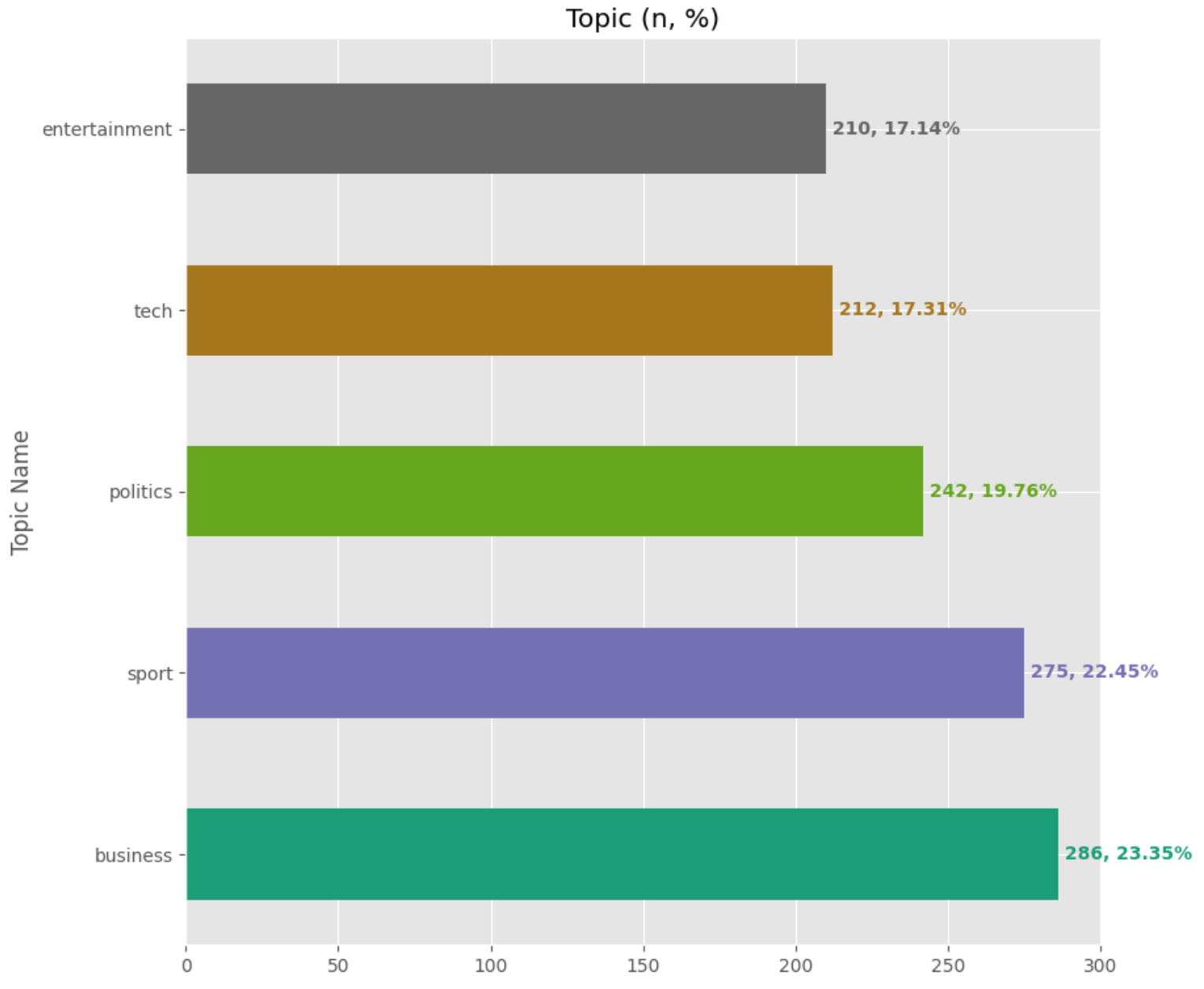
- Cons
  - Can struggle with longer text depending on embedding model used
    - One used here is limited to 128 tokens
  - Works best as unsupervised, which may not suit research needs

Example 2: Finetuning an LLM

- Finetuning an LLM involves using new data to update a model's parameters to your specific task
- Finetuning an entire model is not always the best option
  - Computationally expensive
  - Catastrophic forgetting
- Parameter Efficient Fine Tuning
  - Freeze most of the parameters
  - Train the classifier head that is added to the frozen LLM

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')


num_classes = len(df["label_text"].value_counts())

colors = plt.cm.Dark2(np.linspace(0, 1, num_classes))
iter_color = iter(colors)

df['label_text'].value_counts().plot.barh(title="Topic (n, %)",
                                          ylabel="Topic Name",
                                          color=colors,
                                          figsize=(9,9))

for i, v in enumerate(df['label_text'].value_counts()):
  c = next(iter_color)
  plt.text(v, i,
           " "+str(v)+", "+str(round(v*100/df.shape[0],2))+"%",
           color=c,
           va='center',
           fontweight='bold')
```

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')


num_classes = len(df["label_text"].value_

colors = plt.cm.Dark2(np.linspace(0, 1, n
iter_color = iter(colors)

df['label_text'].value_counts().plot.barh




for i, v in enumerate(df['label_text'].va
  c = next(iter_color)
  plt.text(v, i,
           " "+str(v)+", "+str(round(v*10
           color=c,
           va='center',
           fontweight='bold')
```



```
from sklearn.model_selection import train_test_split
import tensorflow as tf
import keras_nlp
# Add an index column to track original positions
df["original_index"] = df.index
y = tf.keras.utils.to_categorical(df["label"].values, num_classes=5)

x_train, x_test, y_train, y_test = train_test_split(df['text'], y, test_size=0.25, random_state=42, shuffle=True)
```

```
sequence_length = 512     # Sets the maximum number of tokens per input sequence

# Preprocessor (Creates text preprocessing pipeline)
bert_preprocess = keras_nlp.models.BertPreprocessor.from_preset(
    "bert_base_en_uncased",
    sequence_length=sequence_length
)

# encoder (brains of BERT)
bert_encoder = keras_nlp.models.BertBackbone.from_preset(
    "bert_base_en_uncased")


# Model inputs
input_ids = tf.keras.Input(shape=(sequence_length,), dtype=tf.int32, name="input_ids")
padding_mask = tf.keras.Input(shape=(sequence_length,), dtype=tf.int32, name="padding_mask")
segment_ids = tf.keras.Input(shape=(sequence_length,), dtype=tf.int32, name="segment_ids")

bert_encoder.trainable = False    # Specifies that the bert encoder is frozen and not fine tuned

x = bert_encoder({
    "token_ids": input_ids,
    "padding_mask": padding_mask,
    "segment_ids": segment_ids
})["pooled_output"]
x = tf.keras.layers.Dropout(0.2)(x)   # specifies dropout regularization to reduce chance of overfitting
outputs = tf.keras.layers.Dense(5, activation="softmax")(x) # adds the classifier heads that will assign input to
one of the five categories

model = tf.keras.Model([input_ids, padding_mask, segment_ids], outputs)
model.summary()
```

```
x_train = tf.convert_to_tensor(x_train, dtype=tf.string)
x_test  = tf.convert_to_tensor(x_test, dtype=tf.string)
y_train = tf.convert_to_tensor(y_train, dtype=tf.float32)
y_test  = tf.convert_to_tensor(y_test, dtype=tf.float32)
# Convert raw strings into numeric token IDs + masks
x_train_tokens = bert_preprocess(x_train)  # returns dict of tensors
x_test_tokens  = bert_preprocess(x_test)
y_train_int = np.argmax(y_train, axis=1)
y_test_int  = np.argmax(y_test, axis=1)
```

```
# Number of passes over the training data
n_epochs = 10

# Stops the model from continuing if not improving
earlystop_callback = tf.keras.callbacks.EarlyStopping(monitor = "val_loss",
                                                      patience = 3,
                                                      restore_best_weights = True)

# How to train the model
model.compile(optimizer = "adam",
              loss = "sparse_categorical_crossentropy",
              metrics = ["accuracy"])

# Actual training call
model_fit = model.fit(
    [x_train_tokens["token_ids"], x_train_tokens["padding_mask"], x_train_tokens["segment_ids"]],
    y_train_int,
    validation_data=(
        [x_test_tokens["token_ids"], x_test_tokens["padding_mask"], x_test_tokens["segment_ids"]],
        y_test_int
    ),
    epochs=n_epochs,
    batch_size=8,
    callbacks=[earlystop_callback]
)
```

```
# Number of passes over the training data
n_epochs = 10

# Stops the model from continuing if not improving
```

```
Epoch 1/10
115/115 [==============================] - 84s 595ms/step - loss: 1.4889 - accuracy: 0.3736 - val_loss: 1.2351 - val_accuracy: 0.5147
Epoch 2/10
115/115 [==============================] - 59s 511ms/step - loss: 1.1568 - accuracy: 0.5719 - val_loss: 0.9969 - val_accuracy: 0.6873
Epoch 3/10
115/115 [==============================] - 59s 513ms/step - loss: 1.0262 - accuracy: 0.6078 - val_loss: 0.9271 - val_accuracy: 0.6873
Epoch 4/10
115/115 [==============================] - 59s 512ms/step - loss: 0.9022 - accuracy: 0.6961 - val_loss: 0.7854 - val_accuracy: 0.8371
Epoch 5/10
115/115 [==============================] - 59s 511ms/step - loss: 0.8270 - accuracy: 0.7179 - val_loss: 0.7697 - val_accuracy: 0.7459
Epoch 6/10
115/115 [==============================] - 59s 511ms/step - loss: 0.7531 - accuracy: 0.7560 - val_loss: 0.6732 - val_accuracy: 0.8078
Epoch 7/10
115/115 [==============================] - 59s 511ms/step - loss: 0.6812 - accuracy: 0.7898 - val_loss: 0.6020 - val_accuracy: 0.8502
Epoch 8/10
115/115 [==============================] - 59s 511ms/step - loss: 0.6527 - accuracy: 0.7876 - val_loss: 0.5740 - val_accuracy: 0.8762
Epoch 9/10
115/115 [==============================] - 59s 510ms/step - loss: 0.6205 - accuracy: 0.7919 - val_loss: 0.5205 - val_accuracy: 0.8827
Epoch 10/10
115/115 [==============================] - 59s 512ms/step - loss: 0.5546 - accuracy: 0.8420 - val_loss: 0.4826 - val_accuracy: 0.8893
```

```
    epochs=n_epochs,
    batch_size=8,
    callbacks=[earlystop_callback]
)
```

```python
from sklearn.metrics import classification_report

y_pred = model.predict([x_test_tokens["token_ids"], x_test_tokens["padding_mask"], x_test_tokens["segment_ids"]])
y_pred_int = y_pred.argmax(axis=1)

print(classification_report(y_test_int, y_pred_int))
```

```
•••  10/10 [==============================] - 18s 2s/step
              precision    recall  f1-score   support

           0       0.85      0.90      0.88        51
           1       0.85      0.85      0.85        61
           2       0.94      0.95      0.95        66
           3       0.87      0.84      0.85        56
           4       0.92      0.89      0.90        73

    accuracy                           0.89       307
   macro avg       0.89      0.89      0.89       307
weighted avg       0.89      0.89      0.89       307
```

```
import numpy as np

# Predict probabilities on test set
y_test_probs = model.predict([
    x_test_tokens["token_ids"],
    x_test_tokens["padding_mask"],
    x_test_tokens["segment_ids"]
])

# Convert to predicted class integers
y_test_pred = np.argmax(y_test_probs, axis=1)

# Predict on training set
y_train_probs = model.predict([
    x_train_tokens["token_ids"],
    x_train_tokens["padding_mask"],
    x_train_tokens["segment_ids"]
])
y_train_pred = np.argmax(y_train_probs, axis=1)

df_all = pd.DataFrame({
    "text": np.concatenate([x_train, x_test]),
    "actual": np.concatenate([y_train_int, y_test_int]),
    "predicted": np.concatenate([y_train_pred, y_test_pred])
})
```

```
import numpy as np

# Predict probabiliti
y_test_probs = model.
    x_test_tokens["to
    x_test_tokens["pa
    x_test_tokens["se
])

# Convert to predicte
y_test_pred = np.argm

# Predict on training
y_train_probs = model
    x_train_tokens["t
    x_train_tokens["padding_mask"],
    x_train_tokens["segment_ids"]
])
y_train_pred = np.argmax(y_train_probs, axis=1)

df_all = pd.DataFrame({
    "text": np.concatenate([x_train, x_test]),
    "actual": np.concatenate([y_train_int, y_test_int]),
    "predicted": np.concatenate([y_train_pred, y_test_pred])
})
```

```
10/10 [==============================] - 14s 2s/step
29/29 [==============================] - 44s 2s/step
```

	text	actual	predicted
0	b'what really divides the parties so what is t...	4	4
1	b'prince crowned top music earner prince ear...	3	3
2	b'fuming robinson blasts officials england coa...	2	2
3	b'cup holders man utd visit everton holders ma...	2	2
4	b'sydney return for henin-hardenne olympic cha...	2	2

0 – Tech
1 – Business
2 – Sport
3 – Entertainment
4 - Politics

Finetuning Pros and Cons

- Pros
  - Can create a powerful model uniquely suited to your needs
  - Replicable and reusable once created
  - Based on your own pre-determined topics

- Cons
  - Expensive/Computationally intensive
  - Requires training data

Example 3: One-Shot Approach

- Bert is a much weaker model than even old OpenAI models
- Depending on your category, you may not need to finetune a model

```
# Step 2: Import libraries
import openai
import getpass

# Enter API key securely
openai.api_key = getpass.getpass("Enter your OpenAI API key: ")
```

```
response = openai.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "user", "content": "respond with one word. Is the following text about entertainment, tech,
politics, sport, or business? text: wales want rugby league training wales could follow england s lead by training
with a rugby league club.  england have already had a three-day session with leeds rhinos  and wales are thought to
be interested in a similar clinic with rivals st helens. saints coach ian millward has given his approval  but if it
does happen it is unlikely to be this season. saints have a week s training in portugal next week  while wales will
play england in the opening six nations match on 5 february.  we have had an approach from wales  confirmed a
saints spokesman.  it s in the very early stages but it is something we are giving serious consideration to.  st
helens  who are proud of their welsh connections  are obvious partners for the welsh rugby union  despite a spat in
2001 over the collapse of kieron cunningham s proposed £500 000 move to union side swansea. a similar cross-code
deal that took iestyn harris from leeds to cardiff in 2001 did go through  before the talented stand-off returned to
the 13-man code with bradford bulls. kel coslett  who famously moved from wales to league in the 1960s  is currently
saints  football manager  while clive griffiths - wales  defensive coach - is a former st helens player and is
thought to be the man behind the latest initiative. scott gibbs  the former wales and lions centre  played for st
helens from 1994-96 and was in the challenge cup-winning team at wembley in 1996."}
    ],
    max_tokens=50
)

print(response.choices[0].message.content)
```

```python
# Fixed prompt to prepend
fixed_prompt = "respond with one word. Is the following text about entertainment, tech, politics, sport, or
business? text: "

# Loop through the first 10 rows and submit to OpenAI
for i, text in enumerate(df['text'][:10]):
    prompt = fixed_prompt + text
    response = openai.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}],
        max_tokens=50
    )
    summary = response.choices[0].message.content
    print(f"Row {i}:")
    print(f"Input: {text}")
    print(f"Output: {summary}")
    print("-----")
```

```
Row 0:
Input: wales want rugby league training wales could follow england s lead by training with a rugby league club.  england have already had a three-day se
Output: sport
-----
Row 1:
Input: china aviation seeks rescue deal scandal-hit jet fuel supplier china aviation oil has offered to repay its creditors $220m (£117m) of the $550m i
Output: business
-----
Row 2:
Input: rock band u2 break ticket record u2 have smashed irish box office records with ticket sales for their dublin concerts  after more than 150 000 we
Output: entertainment
-----
Row 3:
Input: markets signal brazilian recovery the brazilian stock market has risen to a record high as investors display growing confidence in the durability
Output: business
-----
Row 4:
Input: tough rules for ringtone sellers firms that flout rules on how ringtones and other mobile extras are sold could be cut off from all uk phone netw
Output: business
-----
Row 5:
Input: iraq advice claim sparks new row the tories say ministers must respond in parliament to claims that the legal advice used to justify the iraq war
Output: politics
-----
Row 6:
Input: brits debate over  urban  music joss stone  a 17-year-old soul singer from devon  beat dizzee rascal  jamelia  lemar and the streets to win best
Output: entertainment
-----
Row 7:
Input: dirty den s demise seen by 14m more than 14 million people saw  dirty  den watts killed off on friday  marking eastenders  20th anniversary  acco
Output: entertainment
-----
Row 8:
Input: lib dems  new election pr chief the lib dems have appointed a senior figure from bt to be the party s new communications chief for their next gen
Output: politics
-----
Row 9:
Input: bbc poll indicates economic gloom citizens in a majority of nations surveyed in a bbc world service poll believe the world economy is worsening.
Output: business
-----
```

A quick aside – Batch API

- A new way to access LLMs
- Rather than asking individual queries, many queries are put in a single batch and uploaded together
- They are executed asynchronously in the next 24 hours
- Ideal for when results are not needed immediately (i.e. research)
  - 50% cost reduction
  - Higher rate limits

One-Shot Approach Pros and Cons

- Pros
  - Quick
  - Easy
  - Requires almost no coding

- Cons
  - Expensive
  - Opaque
  - Not fully replicable

Main takeaways

- Many ways to use LLMs for the same task
- Which one to use depends on the task
- Each come with costs and benefits
- It's useful to understand how an LLM works to better understand when it can be useful

The End

Jonathan Colner

Jcolner@american.edu