

Hands-on NLP workshop

James Lloyd
May 17 2022



Agenda

- An introduction to deep learning for natural language processing
 - Word vectors
 - RNNs
 - Transformers
- Build and deploy state of the art models with Abacus.AI
 - Emotion detection
 - Custom classification
 - Named entity recognition
- Perform an analysis using multiple models in a notebook

Sign up to Abacus to follow along

https://bit.ly/abacus_nlp

Other useful links

- Presentation
 - https://bit.ly/abacus_nlp_slides
- Notebook
 - https://bit.ly/abacus_nlp_notebook

Create a first project

The image shows a dark-themed welcome screen for Abacus.AI. At the top, the text 'WELCOME TO ABACUS.AI' is displayed in white, bold, uppercase letters. Below this, a message reads 'The first step is to create a project. It's really simple, give it a try!'. Underneath the message is a prominent blue button labeled 'Create Project'. In the bottom right corner, there is a link 'Want to use our API?' followed by a button labeled 'Create an API Key'. The background features faint concentric circles and small, colorful geometric shapes.

WELCOME TO ABACUS.AI

The first step is to create a project. It's really simple, give it a try!

Create Project

Want to use our API?

Create an API Key

Create a sentiment analysis project

Choose a Solution Use-Case



Personalization AI

- ☐ Personalized Recommendations
- ☐ Related Items
- ☐ Personalized Search



Forecasting and Planning

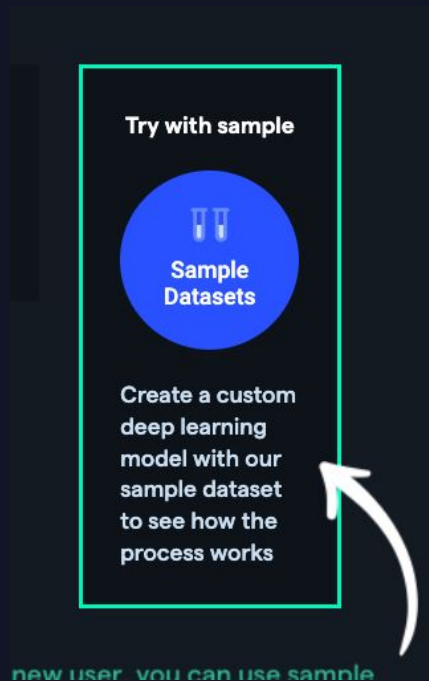
- ☐ Demand Forecasting
- ☐ Real-Time Forecasting



Natural Language Processing

- ☐ Named entity recognition
- ☐ NLP Powered Search
- ☐ Questions & Answers
- ☒ Sentiment Analysis
- ☐ Language Detection
- ☐ Text classification

Click to use the sample dataset



Wait for data to import



Dataset Processing

This process takes a few minutes . The system is inspecting the data formats , identifying the data types , mapping the columns to a system recognizable feature mapping , calculating meta data values , and doing tons of cool stuff to make your life easier . Please think of something calming while you wait .

While we wait, an intro to word vectors

We are going to build a sentiment analysis model which can classify text as positive or negative or detect emotions e.g.

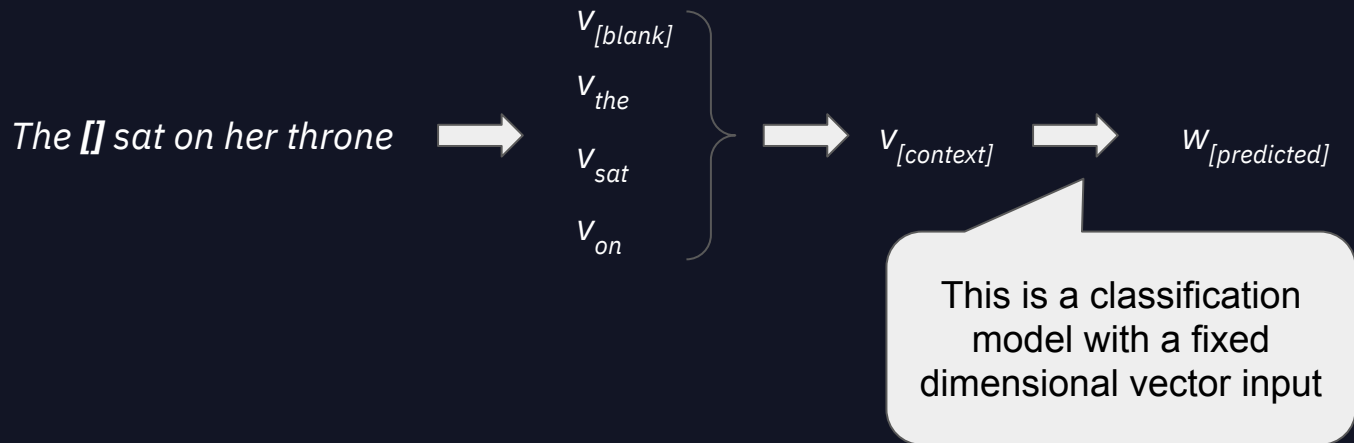
“The fish was excellent” -> positive

The vast majority of machine learning algorithms assume that model inputs are fixed dimensional vectors (a list of numbers of a fixed length)

Can we turn words and sentences into vectors?

A recipe to turn words into vectors

- Assign a vector to every word used in a dataset
- Predict hidden words based on the other words near it
- Optimise vectors and a model to make the best predictions



Word vectors capture the meaning of words

e.g. word algebra

$$\begin{array}{ccccccc}
 \textit{Queen} & - & \textit{woman} & + & \textit{man} & = & \textit{King} \\
 [.2, \dots, .8] & & [0, \dots, .8] & & [0, \dots, -.7] & & [.2, \dots, -.7]
 \end{array}$$

Not all word vectors have this property,
it's just an eye catching way to
demonstrate how some word vectors
capture meaning

To be continued...

Train an emotion detection model

Explore + Train Models

Train models

Training Configuration

Name

sentiment Model



TRAINING FEATURE GROUPS

* List of documents

sample_data_edmunds_car_reviews_1651859939



SENTIMENT TYPE

emotion



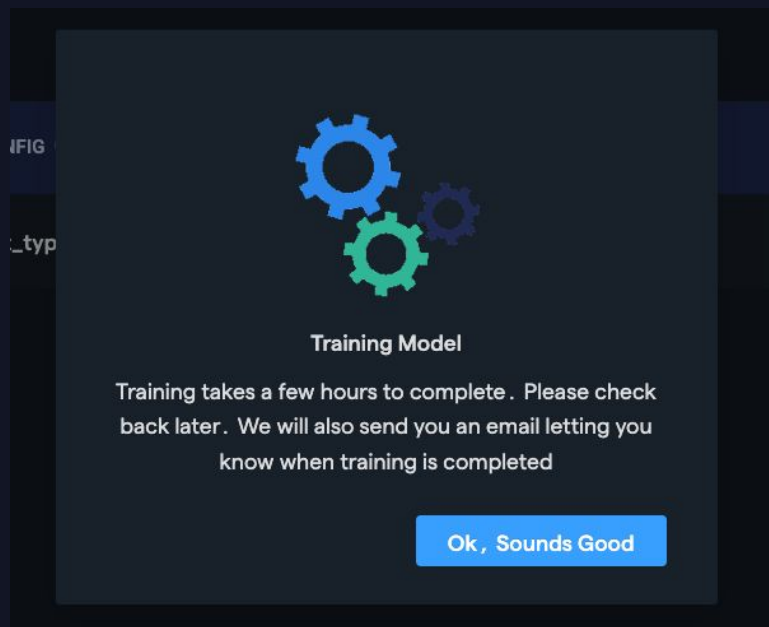
> Advanced Options

Set Refresh Schedule UTC (optional): ?

Enter cron expression

Train Model

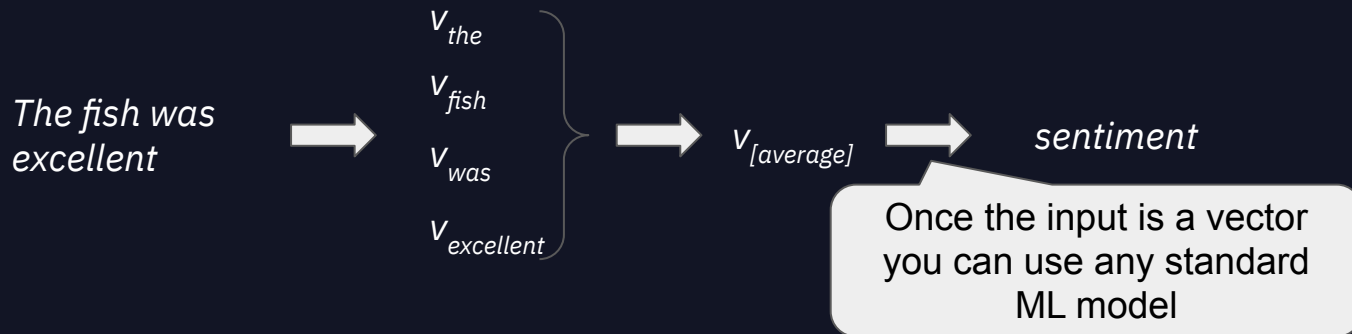
Wait for training



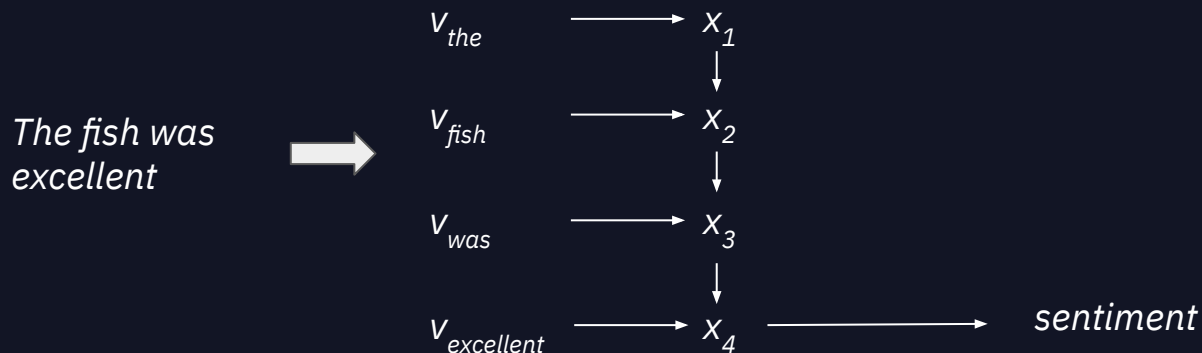
Using word vectors to build a model

We can now turn individual words into vectors but to turn a sentence into a vector we need to combine the individual word vectors

A simple method is to first take the average of all the word vectors - an example of a “continuous bag of words”

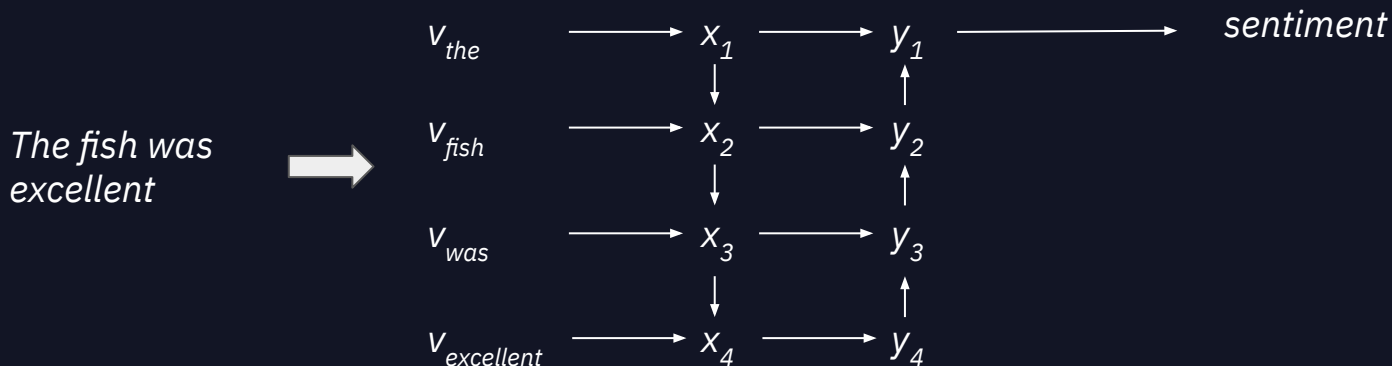


Improving on continuous bag of words - processing data sequentially



The same model is used at each step of processing, allowing the model to work with different lengths of text

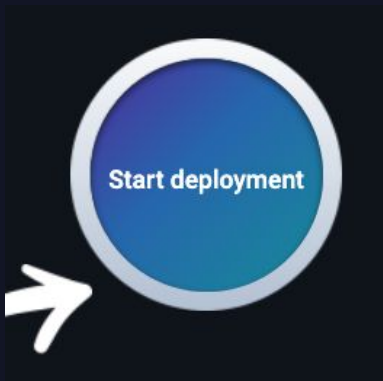
Can process data in both directions as well (e.g. bidirectional LSTM)



This can help with understanding of more complex sentences where words at the end of sentences influence the meaning of earlier words

Deploy the model

| STATUS ? | ACTIONS |
|----------|---|
| Complete | <button>Deploy</button> <button>Re-Train</button> |



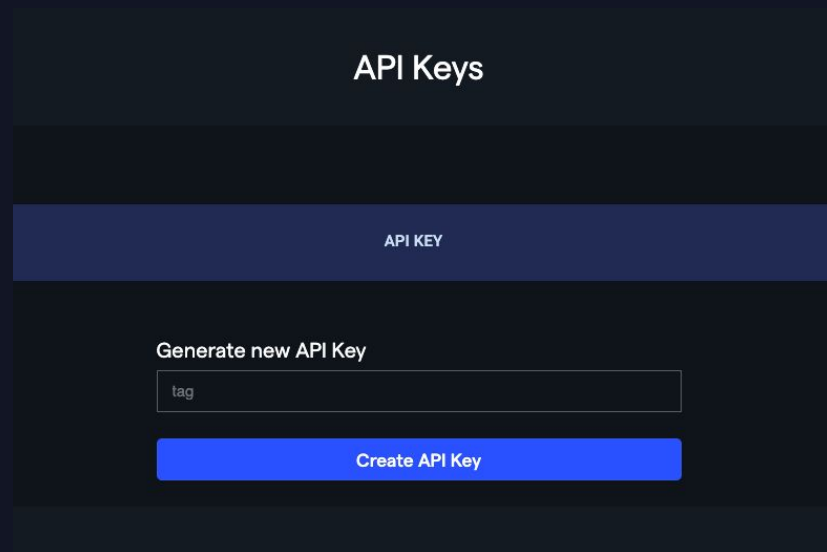
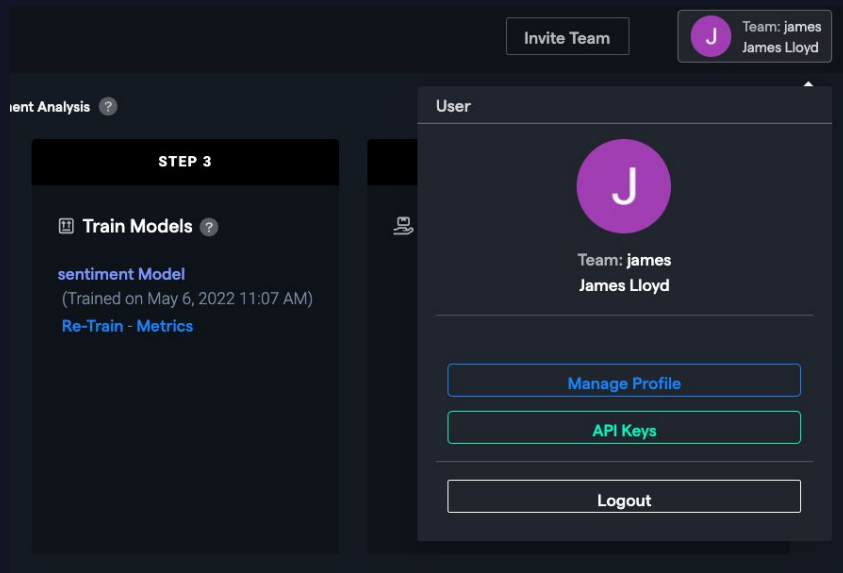
Deployment Name:

Deployment Type:
☐ Offline Batch ☒ Offline Batch + RealTime


* Estimated Number of Calls Per Second:


Deploy Model

While we wait - create an API key







Get feature group ID







 Projects



sentiment



  Datasets

  Feature Groups

  Models

  Deployments

  Batch Predictions

  Monitoring

Feature Groups

Use Type

All 

ATTACHED AT

FEATURE GROUP ID 



May 6, 2022 10:59 AM

34e81c741 

Load feature group in a notebook

Set up API client

```
[4]: api_key = 'api key goes here'
     server = 'https://abacus.ai'
```

```
[5]: api_client = abacusai.ApiClient(api_key=api_key, server=server)
     api_client
```

```
[5]: <abacusai.client.ApiClient at 0x7f0108466430>
```

```
[6]: data = api_client.describe_feature_group('feature group id goes here').load_as_pandas()
     data.head()
```

```
[6]:
```

| | vehicle_title | review_title | review |
|---|---|---|--|
| 0 | 1997 Toyota Previa Minivan LE All-Trac 3dr Min... | my 4th previa, best van ever made! | 1st 95 went over 300k before being totalled b... |
| 1 | 1997 Toyota Previa Minivan LE 3dr Minivan | Mom's Taxi Babies Ride | Sold 86 Toyota Van 285K miles to be replaced ... |
| 2 | 1997 Toyota Previa Minivan LE All-Trac 3dr Min... | Best Minivan ever | My 1997 AWD Previa is the third one that I ha... |
| 3 | 1997 Toyota Previa Minivan LE All-Trac 3dr Min... | Final model year of the great Previa | An amazing vehicle: mid-engine, supercharged,... |
| 4 | 2007 Toyota Avalon Sedan XLS 4dr Sedan (3.5L 6... | I'll drive this car until they take it away fr... | Bought this Avy in 2007 used with 1200 miles ... |

Navigate to predictions dashboard

Search Projects

Projects

sentiment

> Datasets

> Feature Groups

> Models

✓ Deployments

Detail

Predictions Dash

Predictions API

> Batch Predictions

> Monitoring

Predictions Dashboard for Deployment:

sentiment Model Deployment

Need help analyzing predictions? go to [model eval doc](#)

Toyotas will run forever and hybrid design is solid.The car is quiet and the seats are great.The interior build quality is terrible however.My headliner warped in the first 4K miles, the dash creaks and pops when you hit a bump and the door panels also creak constantly.A happy customer tells four people about their experience and unhappy customer tells 10 people.The crappy part is I have to sell it at a loss to buy the ford fusion.My gut told me to go with the ford and I should have followed it.

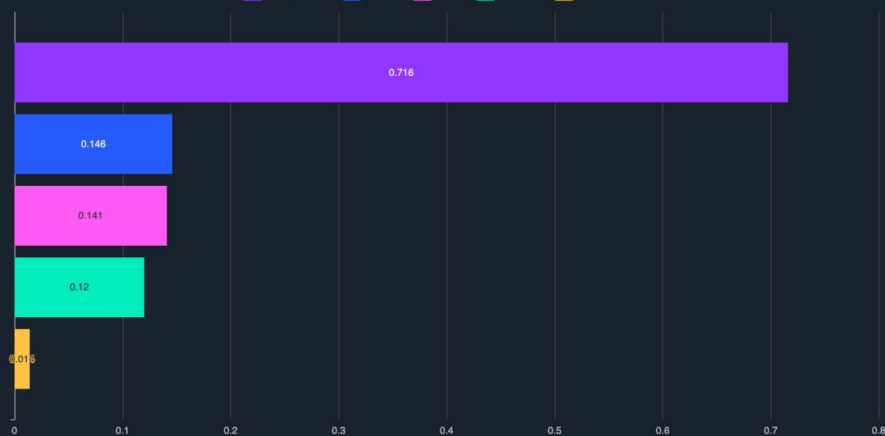
Predict

Experiment
with Test Data

Prediction
API

Predicted

disappointment approval neutral admiration annoyance



Navigate to predictions API and create token

Search Projects

Projects

sentiment

> Datasets

> Feature Groups

> Models

> Deployments

Detail

Predictions Dash

Predictions API

> Batch Predictions

> Monitoring

Predictions API - sentiment Model Deployment

python

Token: ad31ece13f444b87b60f1961bdbc4cf7

Sample Input Data:

* Toyotas will run forever and hybrid design is solid.The car is quiet and the seats are great.The interior build quality is terrible however.I experience and unhappy customer tells 10 people.The crappy part is I have to sell it at a loss to buy the ford fusion.My gut told me to go

Reset data to default

Request: ?

```
from abacusai import ClientOptions, PredictionClient
client_options = ClientOptions(server='https://james-temp-isolated-dev.internalreal.com')
client = PredictionClient(client_options)
client.get_sentiment(deployment_token='ad31ece13f444b87b60f1961bdbc4cf7', deployment_id='83404f41', document=" To
warped in the first 4K miles, the dash creaks and pops when you hit a dump and the door panels also creak constan
the ford fusion.My gut told me to go with the ford and I should have followed it.")
```

Run

Response: ?

```
{
  "admiration": 0.1769905984401703,
  "amusement": 0.00015590489783789963,
  "anger": 0.0008044916903600097,
  "annoyance": 0.049987565726041794,
  "approval": 0.2829807698726654,
```

Copy code into notebook and sort data by strength of different emotions

```
[8]: from abacusai import ClientOptions, PredictionClient
client_options = ClientOptions(server='https://james-tem
client = PredictionClient(client_options)
client.get_sentiment(deployment_token='ad31ece13f444b87b'
```

```
[8]: {'admiration': 0.18734973669052124,
'amusement': 0.0001582959375809878,
'anger': 0.0011313806753605604,
'anoyance': 0.09705094248056412,
'approval': 0.2514726519584656,
'caring': 0.0007489270064979792,
'confusion': 0.0007848410168662667,
'curiosity': 0.0003932887047994882,
'desire': 0.015303357504308224,
'disappointment': 0.8165181875228882,
'disapproval': 0.007338229101151228,
'disgust': 0.0005704679642803967,
'embarrassment': 0.00011633825488388538,
'excitement': 0.0012685491237789392,
'fear': 0.00027635175501927733,
'gratitude': 0.00023449238506145775,
'grief': 0.00071542157093063,
'joy': 0.0019889362156391144,
'love': 0.0002572837402112782,
'nervousness': 0.0010029805125668645,
'optimism': 0.008917922154068947,
'pride': 0.006275418680161238,
'realization': 0.01938549615442753,
'relief': 0.0024816414806991816,
'remorse': 0.002811080776154995,
'sadness': 0.0072595225647091866,
'surprise': 0.00025386386550962925,
'neutral': 0.10354920476675034}
```

```
[11]: sent_deployment_token = 'copy deployment token here'
sent_deployment_id = 'copy deployment ID here'
```

```
[12]: tqdm._instances.clear()
sample_texts = data['review'][:100]
sentiments = [
    client.get_sentiment(
        deployment_token=sent_deployment_token,
        deployment_id=sent_deployment_id,
        document=text
    )
    for text in tqdm(sample_texts)
]
```

100% |██████████| 100/100 [00:21<00:00, 4.72it/s]

```
[15]: query = 'joy'
print(f'Top scoring texts for: "{query}"\n')
scores = [s[query] for s in sentiments]
arg_sort = np.argsort(-np.array(scores))
for i in arg_sort[:5]:
    print(scores[i])
    print(sample_texts[i])
    print('')
```

Top scoring texts for: "joy"

0.9765579104423523

Sold 86 Toyota Van 285K miles to be replaced with 97 Previa mostly for reserve weekend duty. Did not change remote bat 6 le seat & folding 3rd row. Kids love this because they can l: X, MA, CO. With this economy we are glad to have Toyotas in c

```
[16]: query = 'annoyance'
print(f'Top scoring texts for: "{query}"\n')
scores = [s[query] for s in sentiments]
arg_sort = np.argsort(-np.array(scores))
for i in arg_sort[:5]:
    print(scores[i])
    print(sample_texts[i])
    print('')
```

Top scoring texts for: "annoyance"

0.9876571893692017

I bought my 2011 Avalon Limited new in Dec.2011. It is no ew one. Wind comes in window, I'm told it's normal due to e.Not a quiet car at all. Rear passenger window broke (fi) adults and 2 teens with 4 bags packed (not heavy..trunk to

0.9837900996208191

Overall our '13 Avalon is a compromise in meeting our ex ibility afforded to urban driving conditions. Mechanical nnoying is the mediocre workmanship within the cabin's int ble.. Headliner at the top edge of rear window and sewing/ de is a tad light - meaning not stout/solid and stable.

Now let's create a NLP classification model using the same dataset using the API

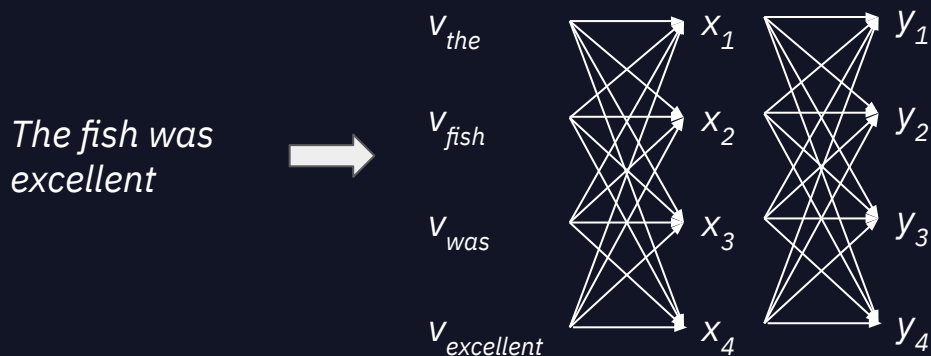
```
[*]: class_project = api_client.create_project('nlp_classification', use_case='NLP_CLASSIFICATION')

api_client.add_feature_group_to_project(
    feature_group_id=feature_group, project_id=class_project, feature_group_type='DOCUMENTS'
)

class_project.set_feature_mapping(
    feature_group_id=feature_group,
    feature_name='review',
    feature_mapping='DOCUMENT',
)

class_model = class_project.train_model(
    name='classification_model_1',
    training_config={
        'zero_shot_hypotheses': [
            'This text is about car speed / acceleration / slowness',
            'This text is about car price / cost / value for money',
            'This text is about car build quality',
            'This text is about car seats',
        ]
    }
)
```



While we wait - an introduction to transformers for NLP




As the data is processed, each word can depend on any other word through an architecture known as “self-attention”

Self attention can allow for nuanced understanding of relationships between words

I poured water from the bottle into the **cup** until **it** was **full**.

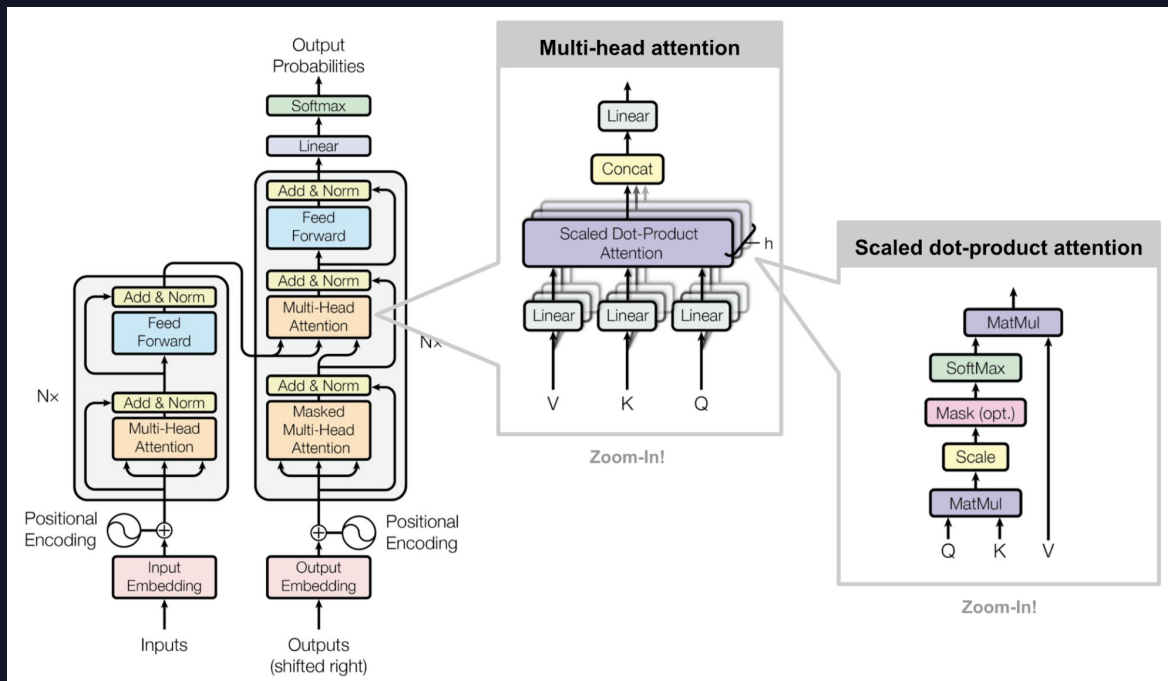


I poured water from the **bottle** into the cup until **it** was **empty**.



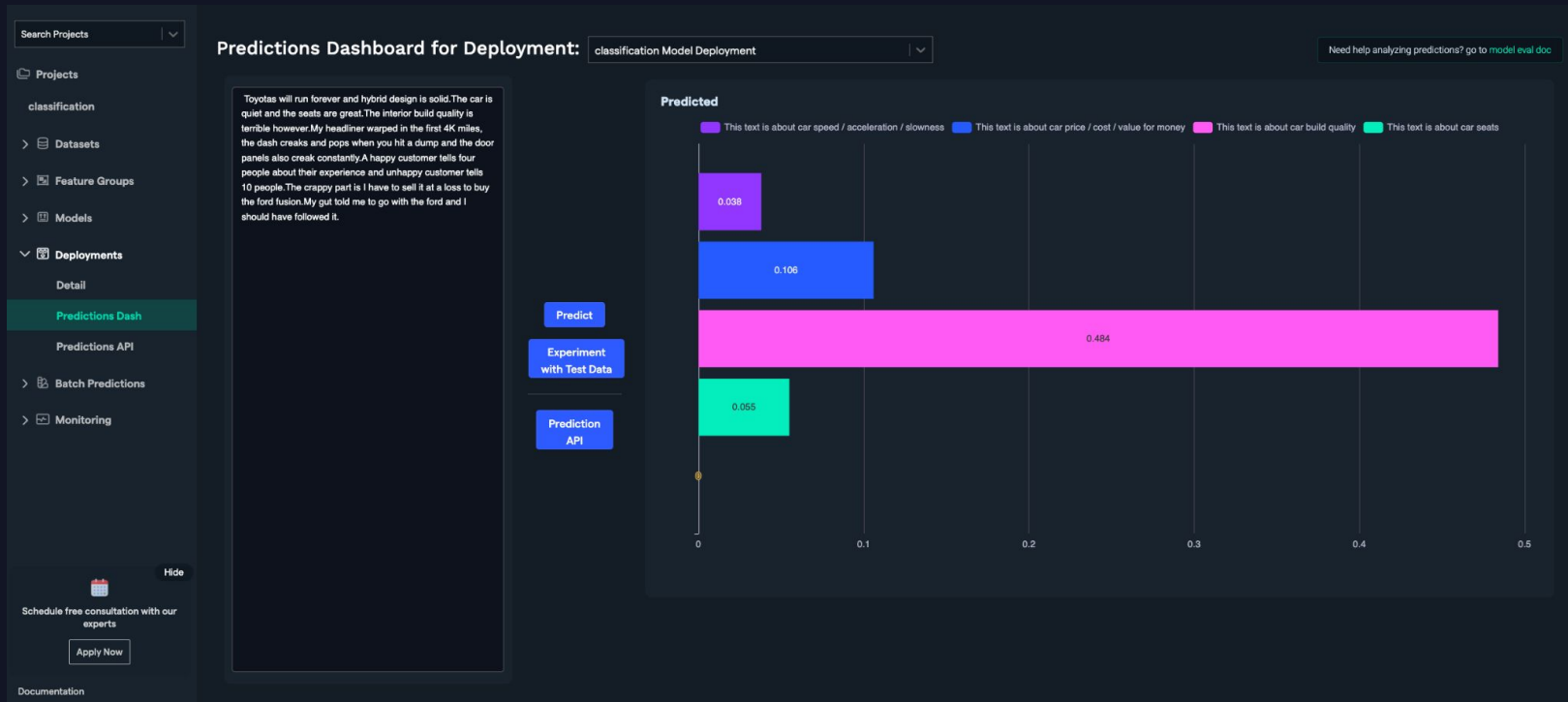
“It” changes meaning depending on full/empty. This is very relevant for tasks such as translation.

Attention - the details



To be continued...

Predictions dashboard



Predictions API

Search Projects

Projects

classification

> Datasets

> Feature Groups

> Models

> Deployments

Detail

Predictions Dash

Predictions API

> Batch Predictions

> Monitoring

Hide

Schedule free consultation with our experts

Predictions API - classification Model Deployment

python

Token: 009e101b09214ff2867a66e67f9364f0

Sample Input Data: ?

* Toyotas will run forever and hybrid design is solid.The car is quiet and the seats are great.The interior build quality is terrible however.My headlir experience and unhappy customer tells 10 people.The crappy part is I have to sell it at a loss to buy the ford fusion.My gut told me to go with the I

Reset data to default

Request: ?

```
from abacusai import ClientOptions, PredictionClient
client_options = ClientOptions(server='https://james-temp-isolated-dev.internalreal.com')
client = PredictionClient(client_options)
client.get_classification(deployment_token='009e101b09214ff2867a66e67f9364f0', deployment_id='78b6eb682', document=" Toyota
headliner warped in the first 4K miles, the dash creaks and pops when you hit a bump and the door panels also creak const
loss to buy the ford fusion.My gut told me to go with the ford and I should have followed it.")
```

Run

Response: ?

```
{
  "This text is about car speed / acceleration / slowness": 0.038277365267276764,
  "This text is about car price / cost / value for money": 0.10617555677890778,
  "This text is about car build quality": 0.4843114912509918,
  "This text is about car seats": 0.055070266127586365,
  "": 0.0002185008197557181
}
```

Quick predictions in code

```
[34]: predictions = []
      for text in sample_texts:
          if text not in text_to_prediction:
              break
          else:
              predictions.append(text_to_prediction[text])
      print(f'Predictions made so far: {len(predictions)}\n')
      query = list(predictions[0].keys())[1]
      print(f'Top scoring texts for: "{query}"\n')
      scores = [s[query] for s in predictions]
      arg_sort = np.argsort(-np.array(scores))
      for i in arg_sort[:5]:
          print(f'Score = {scores[i]}')
          print(sample_texts[i])
          print('')
```

Predictions made so far: 45

Top scoring texts for: "This text is about car price / cost / value for money"

Score = 0.7179281115531921

Fantastic vehicle for the price. Power and economy are much better than my predded on entry and exit. We have had a problem with the windshield washer reservoir. This is an extremely quiet vehicle that gets 27+ mpg on the road and 22+

Now create a named entity recognition (NER) model using the API

Build a named entity recognition (NER) model

```
[*]: ner_project = api_client.create_project('named_entity_recognition', use_case='NAMED_ENTITY_RECOGNITION')

api_client.add_feature_group_to_project(
    feature_group_id=feature_group, project_id=ner_project, feature_group_type='DOCUMENTS'
)

ner_project.set_feature_mapping(
    feature_group_id=feature_group,
    feature_name='review',
    feature_mapping='DOCUMENT',
)
```


Transformers architecture - why does it work so well?

A very tricky question to answer in full generality...

But, part of the answer is computational efficiency on GPUs and other hardware accelerators allowing more data to be processed

While we wait - investigate the classification model in more detail using a model I prepared earlier

NER predictions dashboard

Search Projects

▼

Projects

named_entity_recognition

>

Datasets

>

Feature Groups

>

Models

▼

Deployments

Detail

Predictions Dash

Predictions API

>

Batch Predictions

>

Monitoring

Predictions Dashboard for Deployment:

ner_model_1 Deployment

▼

Need help analyzing predictions? go to [model eval doc](#)

Experiment with Test Data

Prediction API

Decent vehicle as far as Minivans are concerned. But build quality is in question. Minor rattles (Dealer said it was normal). My sliding door motor crapped out after 6 months. Then when it was replaced I can hear it straining to close now. In 2008 I drove my cousin's Odyssey for a month and was very very impressed with it. It looked better inside, seemed to have functionality that suited my needs and definitely handled better. My husband actually enjoyed driving a van (he didn't like driving the Sienna) I'll be switching over to the Odyssey if I ever buy a Van again. I won't be buying any more Toyota due to the accelerator issue fiasco. I'm afraid the same issue will hit Siennas.

Result:

Decent vehicle as far as **Minivans** are concerned. But build quality is in question. Minor rattles (Dealer said it was normal). My sliding door motor crapped out after 6 months. Then when it was replaced I can hear it straining to close now. In 2008 I drove my cousin's **Odyssey** for a month and was very very impressed with it. It looked better inside, seemed to have functionality that suited my needs and definitely handled better. My husband actually enjoyed driving a van (he didn't like driving the **Sienna**) I'll be switching over to the **Odyssey** if I ever buy a **Van** again. I won't be buying any more **Toyota** due to the accelerator issue fiasco. I'm afraid the same issue will hit **Siennas**.

Predict

>

product

9

NER predictions in code - extract e.g. products

```
[40]: label_counts = collections.Counter([anno['displayName'] for anno_list in annotations for anno in anno_list])
label_counts.most_common(10)

[40]: [('product', 310),
      ('organization', 50),
      ('location', 29),
      ('cardinal number', 13),
      ('quantity', 8),
      ('group', 3),
      ('time', 3),
      ('geopolitical area', 3),
      ('date', 3),
      ('ordinal number', 3)]

[41]: product_counts = collections.Counter([anno['textExtraction']['textSegment']['phrase'].strip().lower()
                                           for anno_list in annotations for anno in anno_list if anno['displayName'] == 'product'])
product_counts.most_common(10)

[41]: [('avalon', 55),
      ('toyota', 24),
      ('lexus', 10),
      ('camry', 5),
      ('aval', 4),
      ('yo', 4),
      ('2013 avalon', 4),
      ('crosse', 4),
      ('', 4),
      ('la', 4)]
```

Use multiple models to perform a more detailed analysis

```
[42]: # Filter using NER

filtered_texts = [
    text
    for text, anno_list in zip(sample_texts, annotations)
    if any([anno['displayName'] == 'product' and 'avalon' in anno['textExtraction']] for anno in anno_list])
]
len(filtered_texts)
```

[42]: 55

```
[49]: # See top scores after the filter

predictions = []
for text in filtered_texts:
    if text not in text_to_prediction:
        break
    else:
        predictions.append(text_to_prediction[text])
query = list(predictions[0].keys())[1]
print(f'Top scoring texts for: "{query}"\n')
scores = [s[query] for s in predictions]
arg_sort = np.argsort(-np.array(scores))
for i in arg_sort[:5]:
    print(scores[i])
    print(filtered_texts[i])
    print('')
```

Top scoring texts for: "This text is about car price / cost / value for money"

0.6019570827484131

I just got the 2015 limited in year end clearance; I got the best price for the best car he top line in term of luxury, tech, stylish, smooth and quiet on free way on par with L inted a bid on free way, I don't much wind but tire sound with the road on the 18' wheel ies, regal, lacrosse, ATS are more power and torque but I did not feel much different. en Lacrosse on the to ride on freeway— smooth, quiet, and balance due to its heavy weig d bargain price at the end of the year make me to pick Avalon limited over the rest.

What next?

- Try these models on your own data
- Batch predictions to process large data efficiently
- Try other model types
- ...