Hands-on NLP workshop

James Lloyd
May 17 2022

ABACUS.AI

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

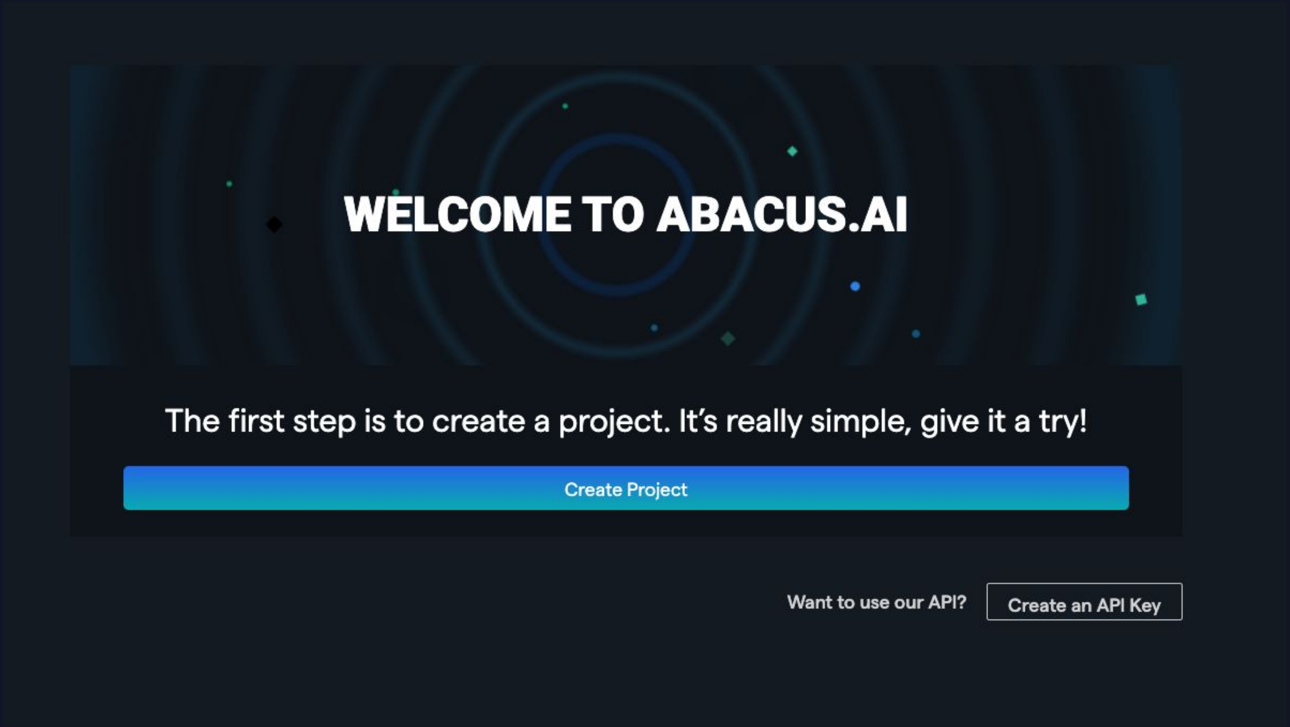
https://bit.ly/abacus_nlp

Other useful links

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

ABACUS.AI

Create a first project



ABACUS.AI

Create a sentiment analysis project

Choose a Solution Use-Case ?

Personalization AI

- Personalized Recommendations
- Related Items
- Personalized Search

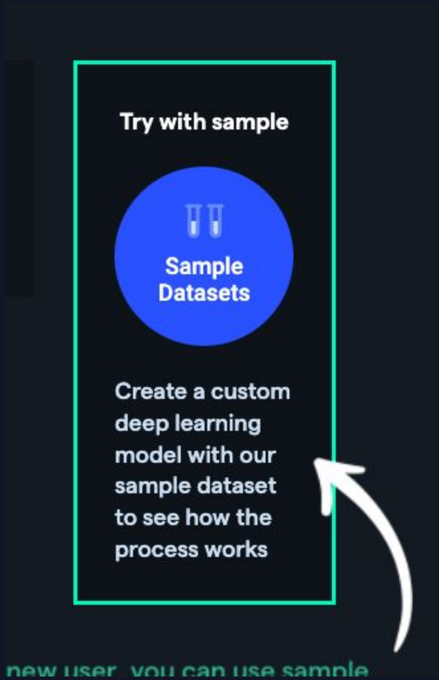
Forecasting and Planning

- Demand Forecasting
- Real-Time Forecasting

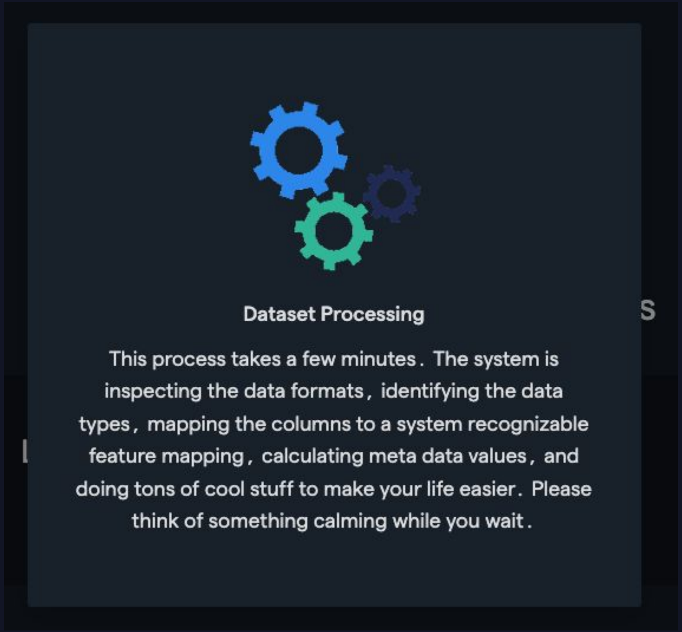
Natural Language Processing

- 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



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.

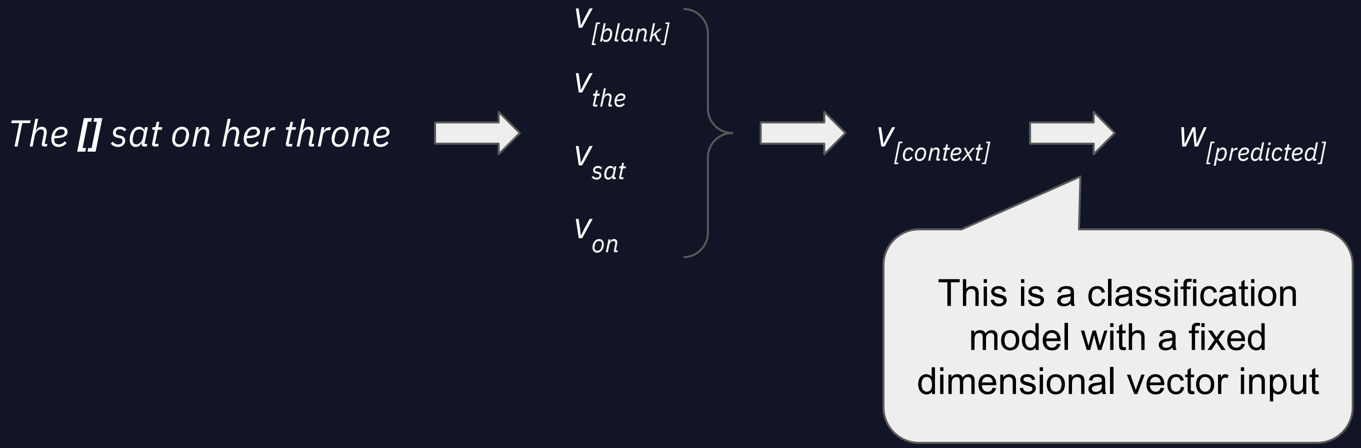
“The fish was excellent” -> positive

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

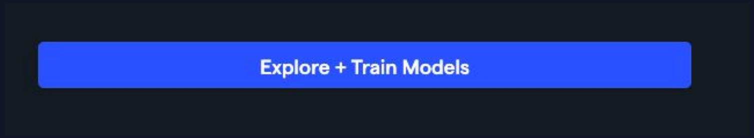
e.g. word algebra

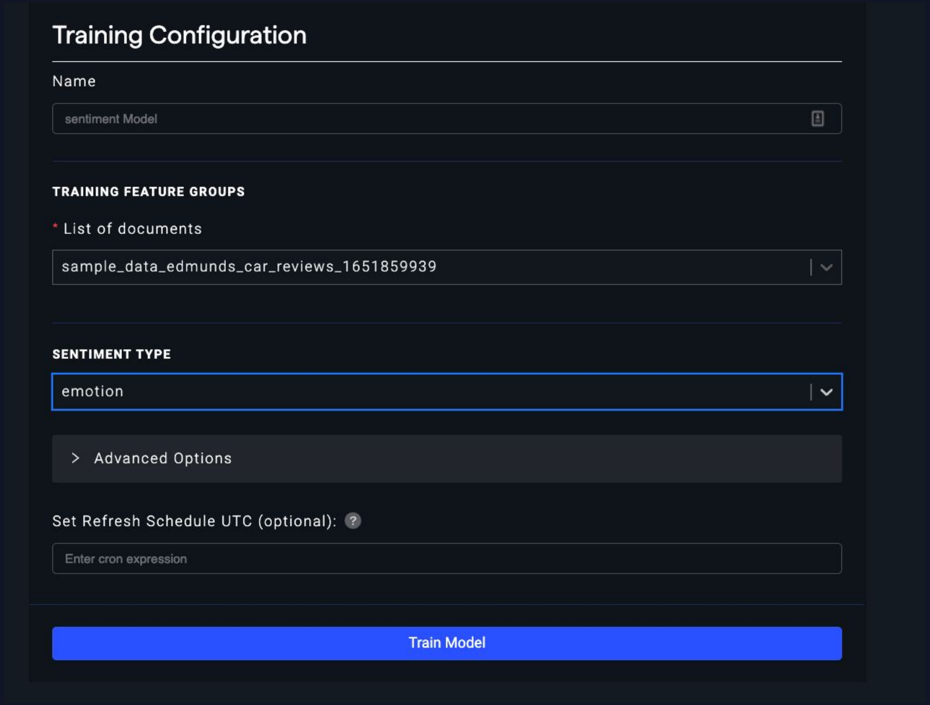
Queen	-	*woman*	+	*man*	=	*King*
[.2, ..., .8]		*[0, ..., .8]*		*[0, ..., -.7]*		*[.2, ..., -.7]*

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

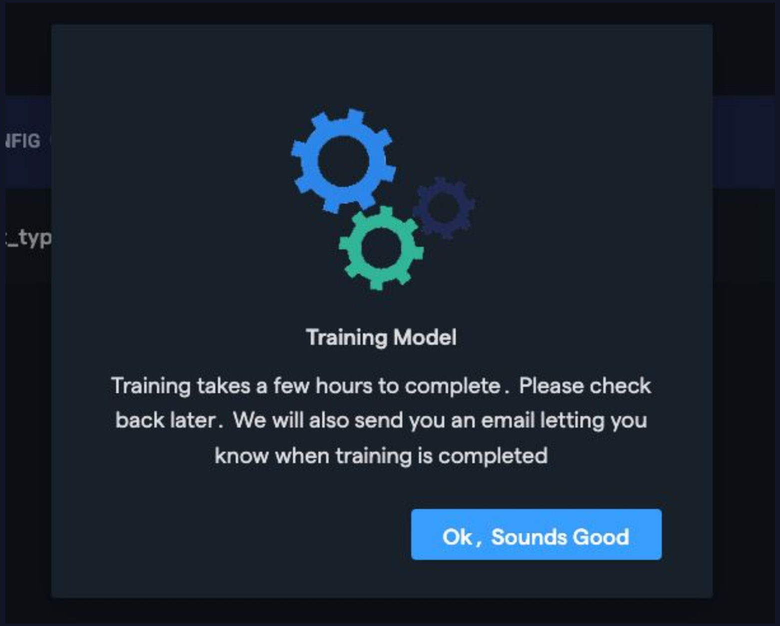
To be continued…

Train an emotion detection model





Wait for training



Using word vectors to build a model

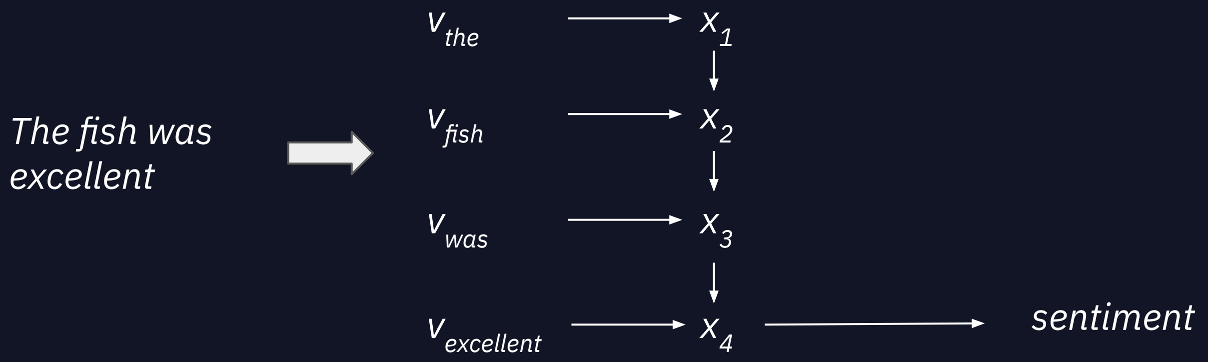
We can now turn individual words into vectors but to turn a sentence into a vector we need to combine the individual word vectors

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

The fish was excellent $\rightarrow$ $\left.\begin{matrix} v_{the} \\ v_{fish} \\ v_{was} \\ v_{excellent} \end{matrix}\right\}$ $\rightarrow$ $v_{[average]}$ $\rightarrow$ *sentiment*

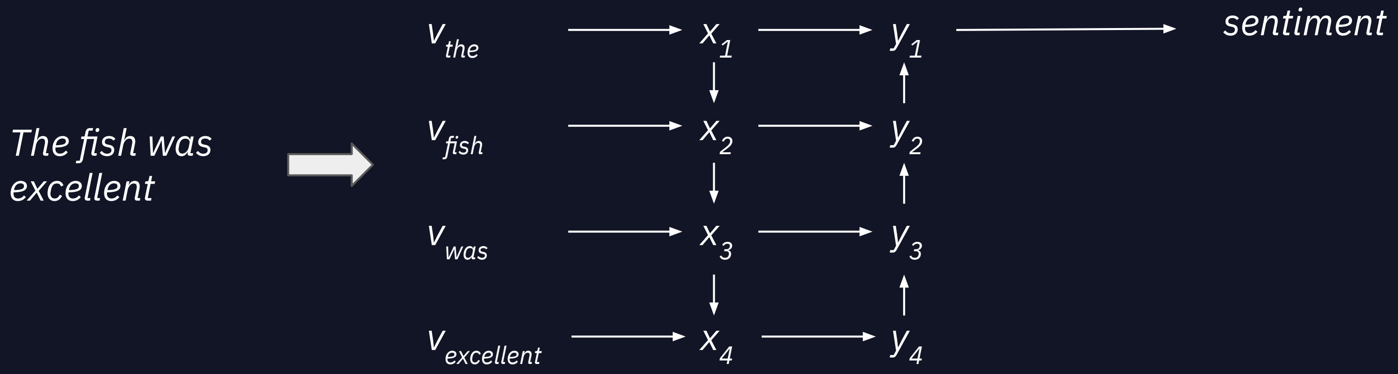
Once the input is a vector you can use any standard ML model

Improving on continuous bag of words - processing data sequentially



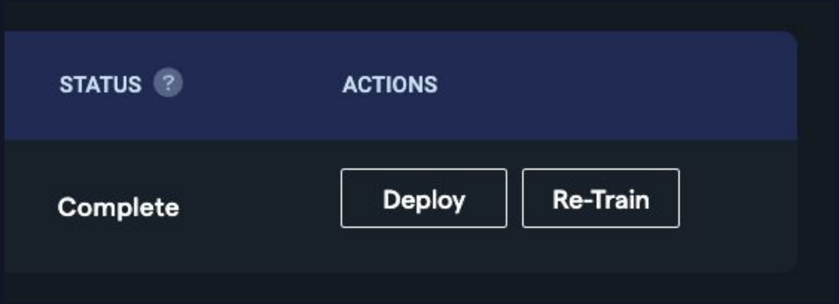
The same model is used at each step of processing, allowing the model to work with different lengths of text

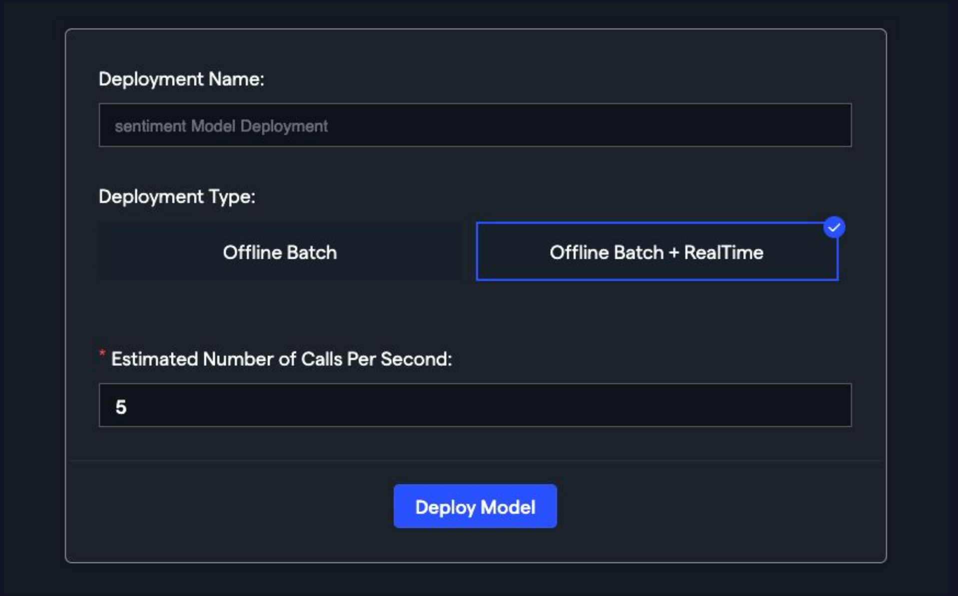
Can process data in both directions as well (e.g. bidirectional LSTM)



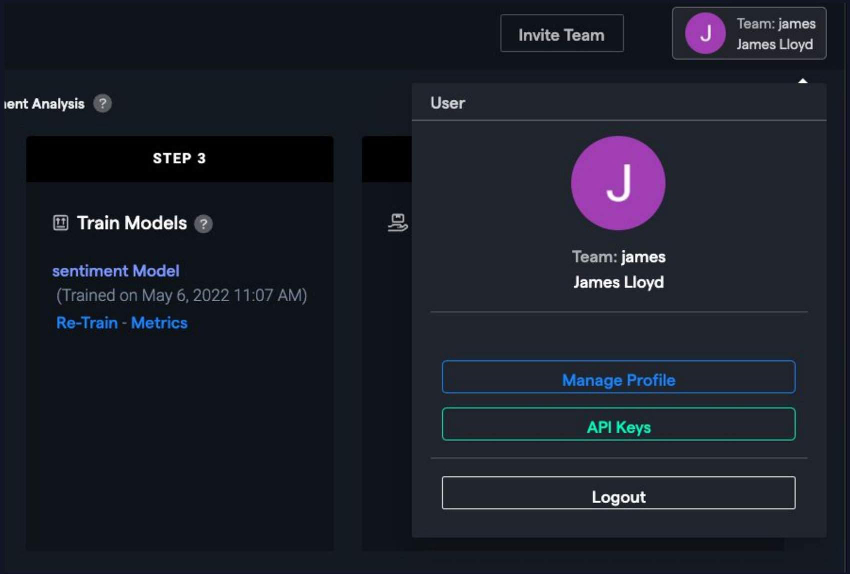
This can help with understanding of more complex sentences where words at the end of sentences influence the meaning of earlier words

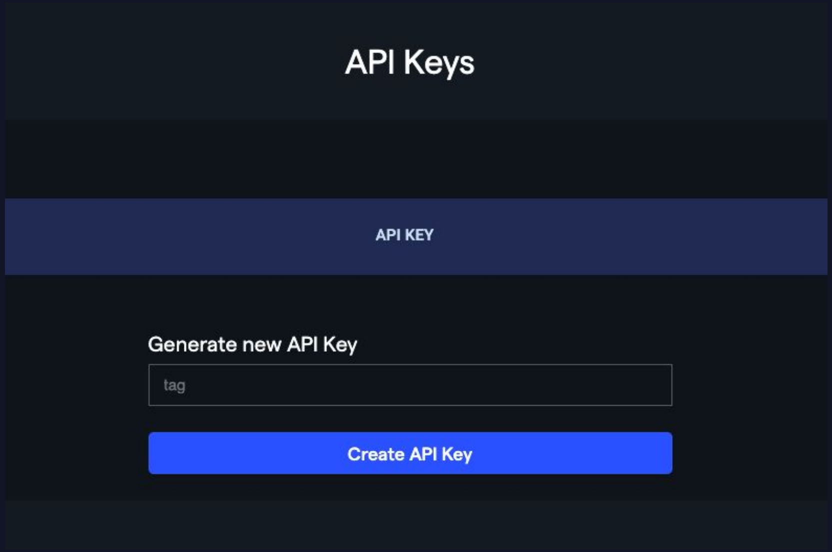
Deploy the model



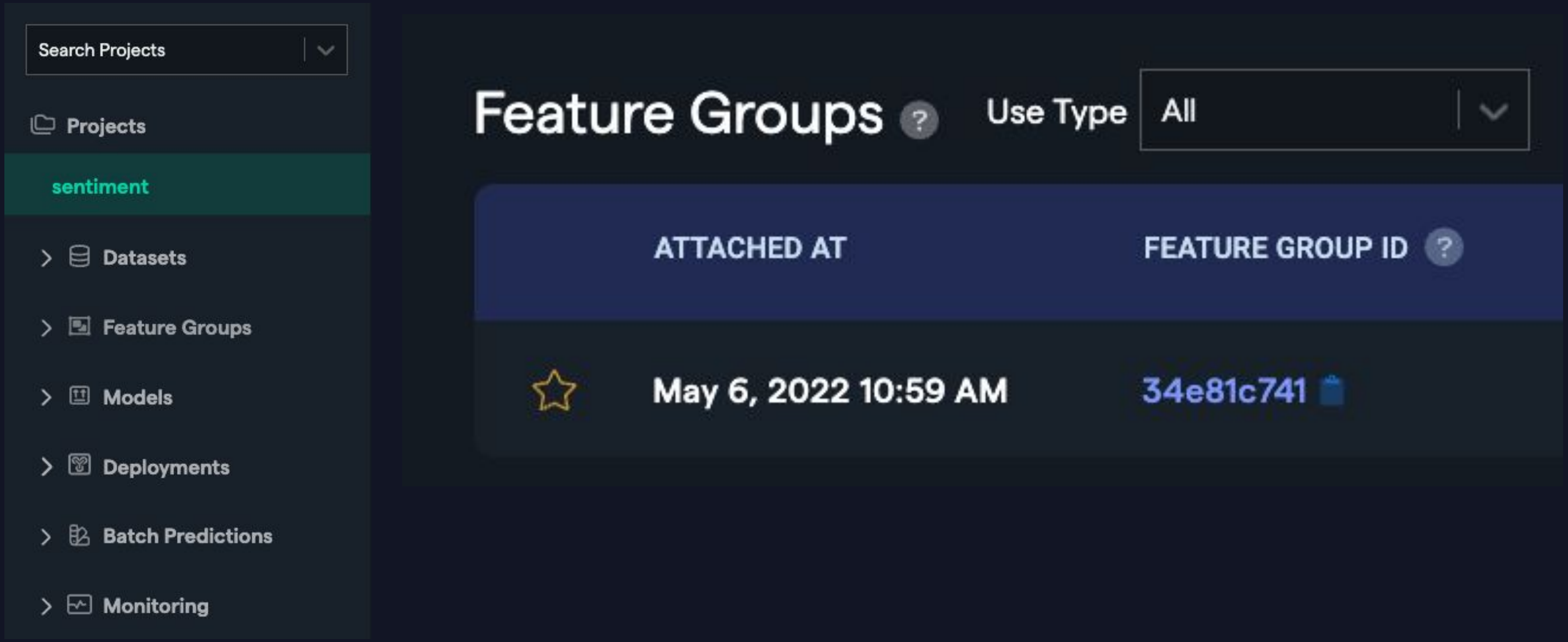


While we wait - create an API key





Get feature group ID



ABACUS.AI

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
```

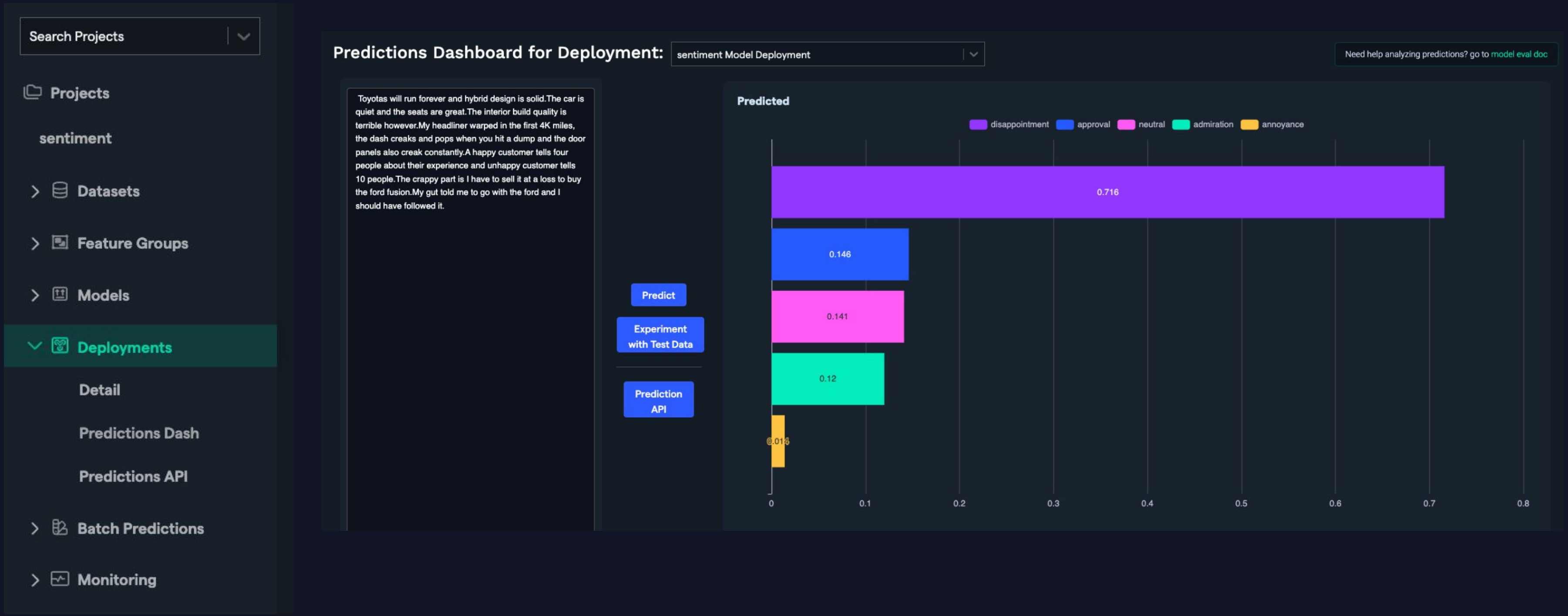
[5]:
```
<abacusai.client.ApiClient at 0x7f0108466430>
```

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

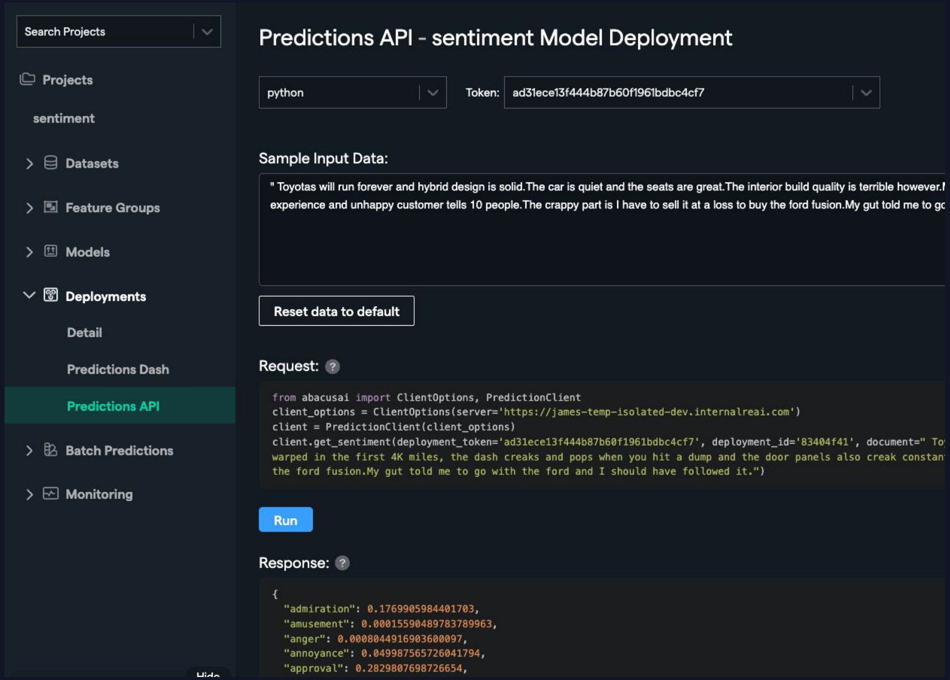
[6]:

	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



Navigate to predictions API and create token



ABACUS.AI

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': 0.00015829593375809878,
      'anger': 0.0011313806753605604,
      'annoyance': 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': 0.0002763517550192773,
      'gratitude': 0.00023449238506145775,
      'grief': 0.00071542157093063,
      'joy': 0.0019889362156391144,
      'love': 0.00025728374021127782,
      'nervousness': 0.0010029805125668645,
      'optimism': 0.008917922154068947,
      'pride': 0.006275418680161238,
      'realization': 0.01938549615442753,
      'relief': 0.0024816414806991816,
      'remorse': 0.002811080776154995,
      'sadness': 0.007259522564709186,
      '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
```

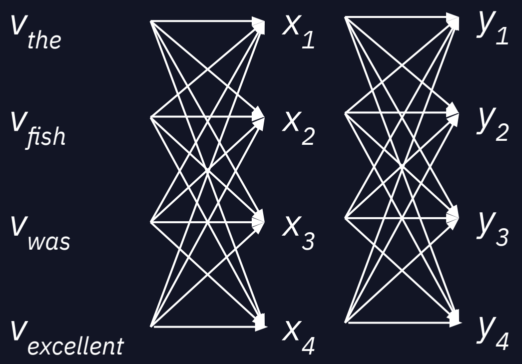
```
[16]: query = 'annoyance'
      print(f'Top scoring texts for: "{query}"\n')
      scores = [s[query] for s in sentiments]
      arg_sort = np.argsort(-np.array(scores))
      for i in arg_sort[:5]:
          print(scores[i])
          print(sample_texts[i])
          print('')

      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 e
      ibility afforded to urban driving conditions.  Mechanical
      nnoying is the mediocre workmanship within the cabin's in
      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

The fish was excellent



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

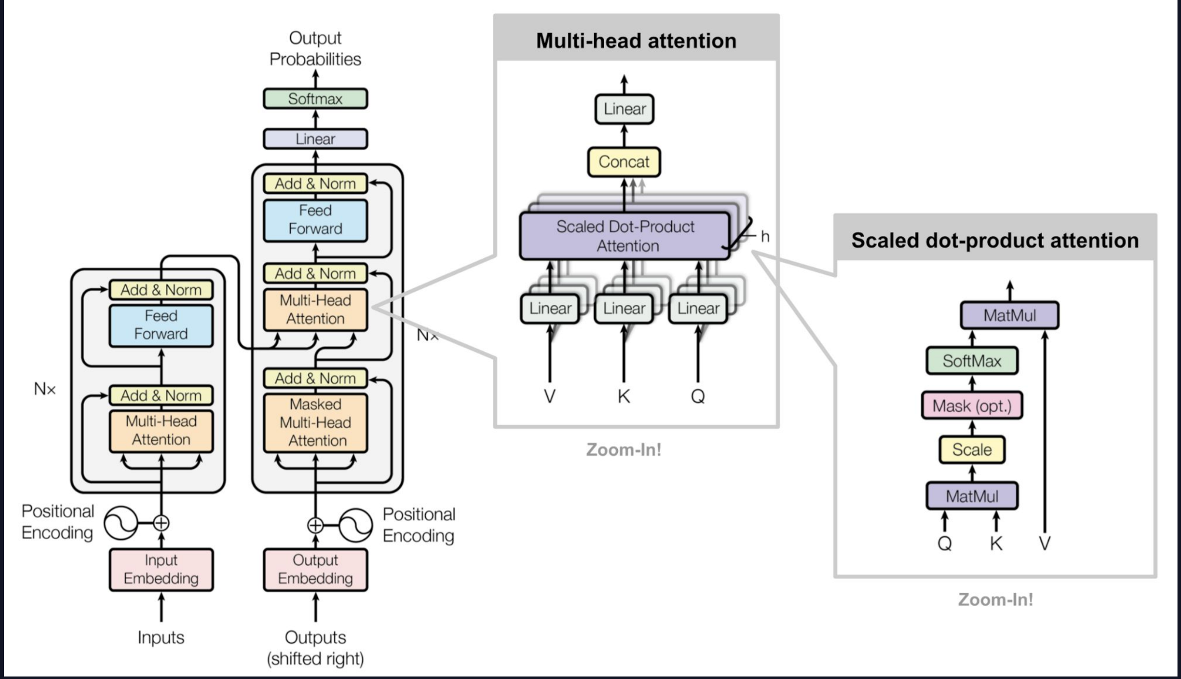
Self attention can allow for nuanced understanding of relationships between words

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.

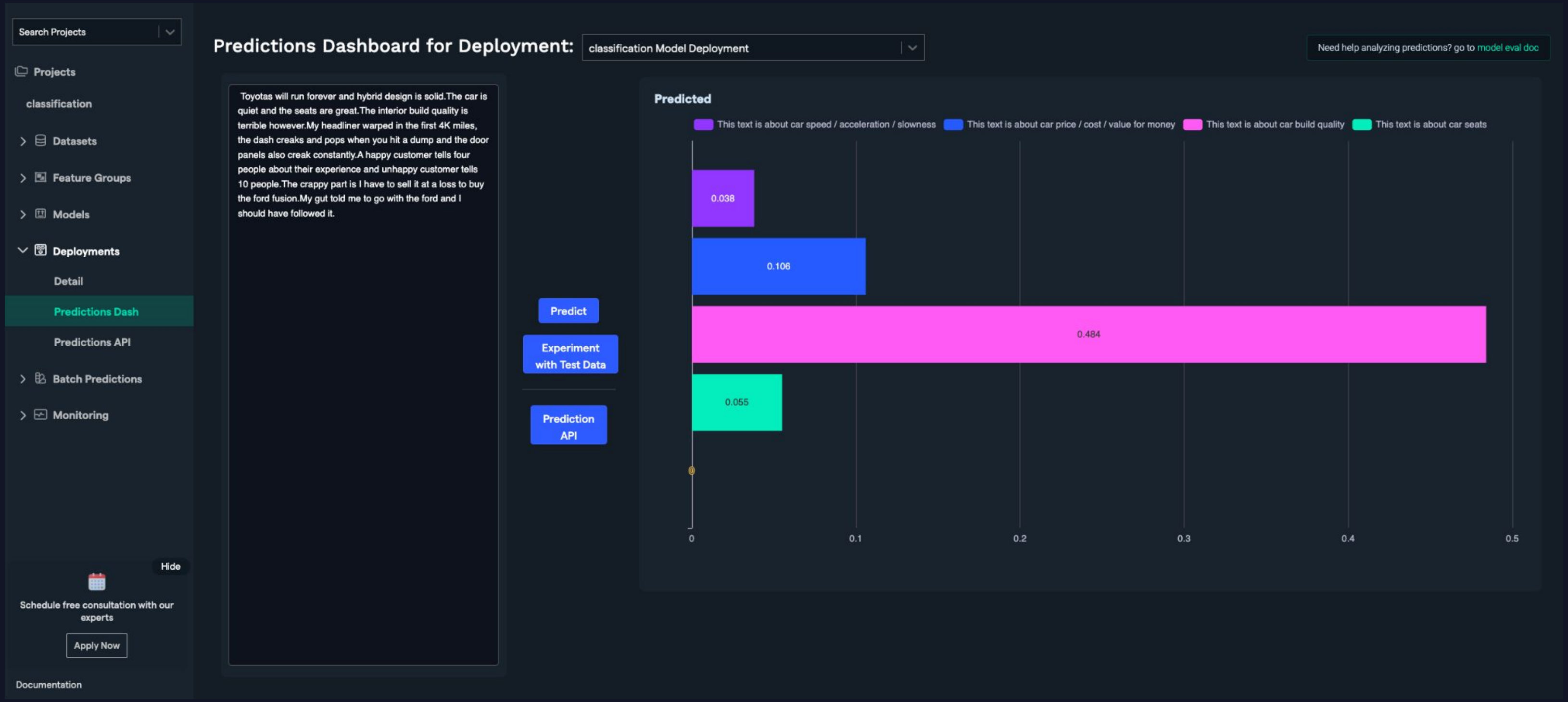
Attention - the details



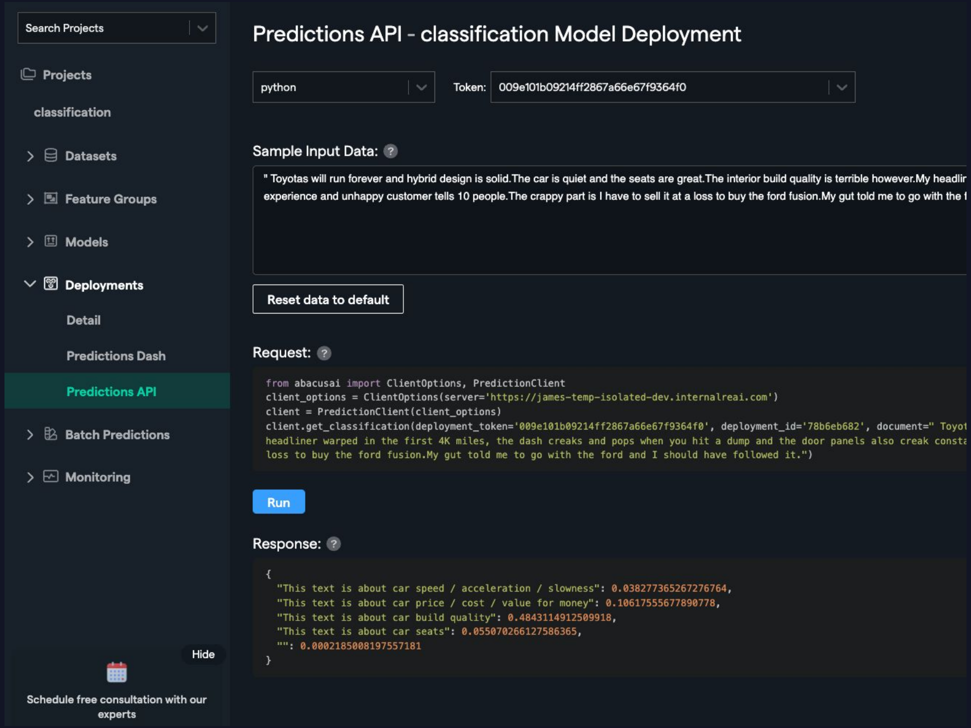
Source: https://lilianweng.github.io/posts/2018-06-24-attention/

To be continued…

Predictions dashboard



Predictions API



ABACUS.AI

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 prev
dded on entry and exit.  We have had a problem with the windshield washer reserv
rvoir.  This is an extremely quiet vehicle that gets 27+ mpg on the road and 22+
```

ABACUS.AI

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',
     )
```

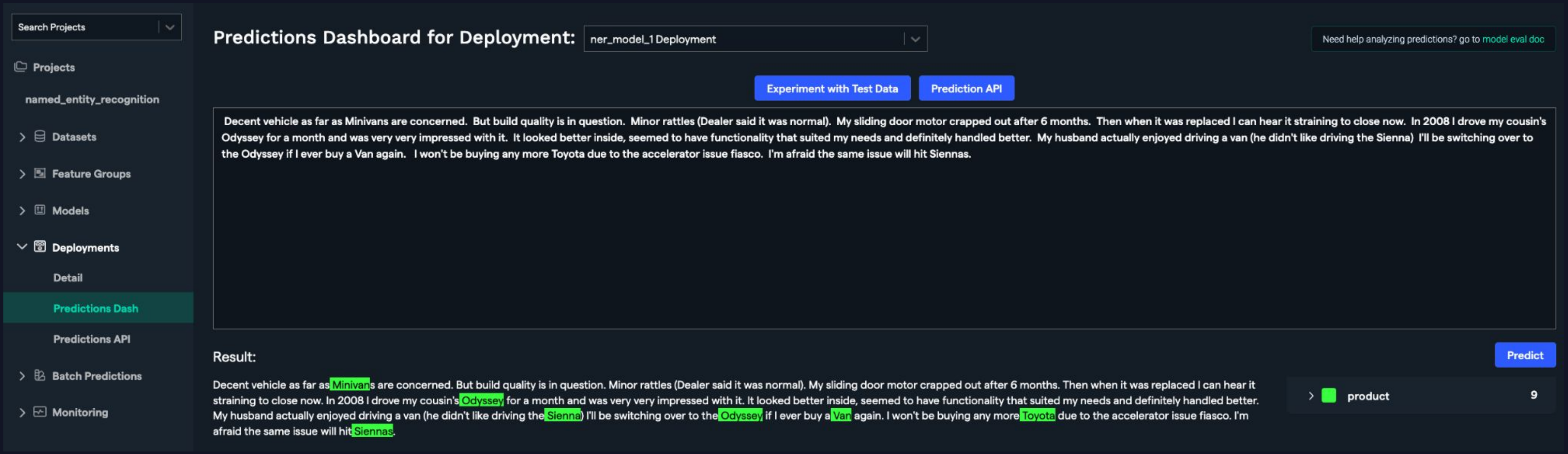
Transformers architecture - why does it work so well?

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



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']['tex
           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 ca
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
- ...