Semantic Search

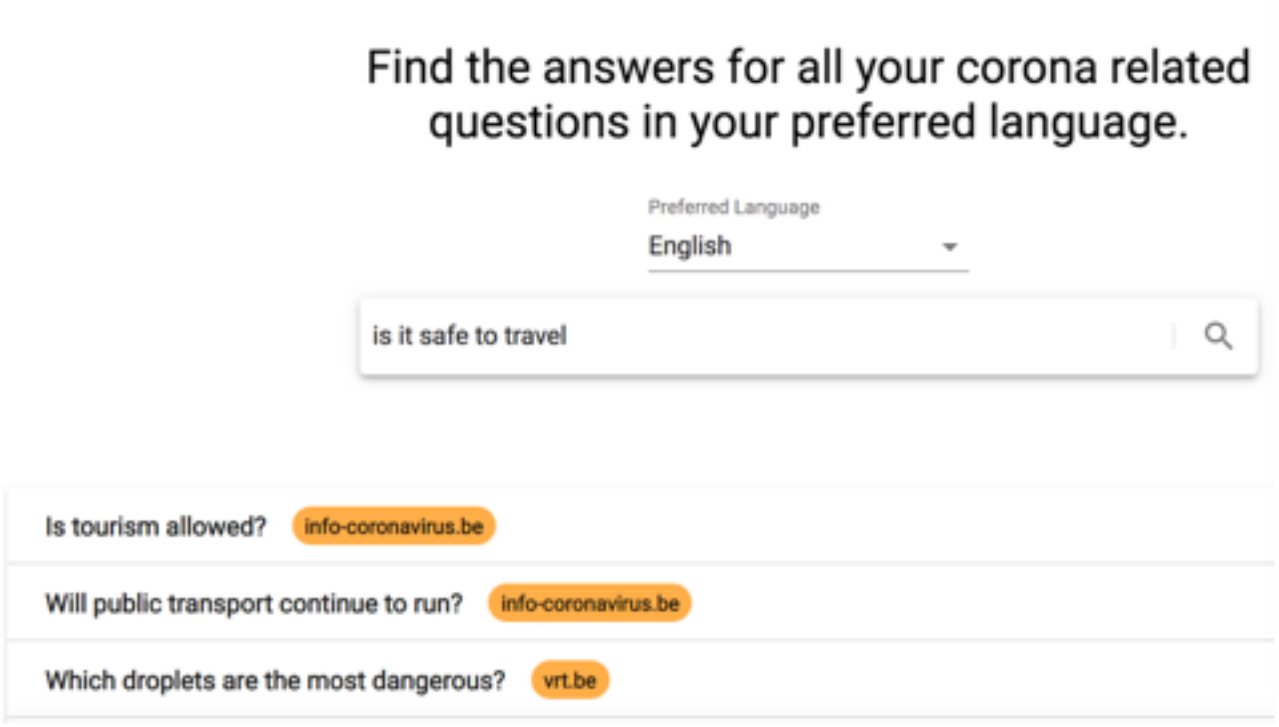
Built a COVID-19 Assistant Powered by NLP

Syllabus

- Context
- Search Engine Methodology
- Scraping the data
- Semantic Search using NLP
- Building the API

Context

The COVID-19 pandemic is changing the world rapidly and keeping people well-informed is an important step in flattening the curve. This is why ML6 decided to build a multilingual COVID-19 assistant powered by AI, more specifically Natural Language Processing (NLP).


Find the answers for all your corona related questions in your preferred language.

Preferred Language: English

is it safe to travel

Is tourism allowed? info-coronavirus.be

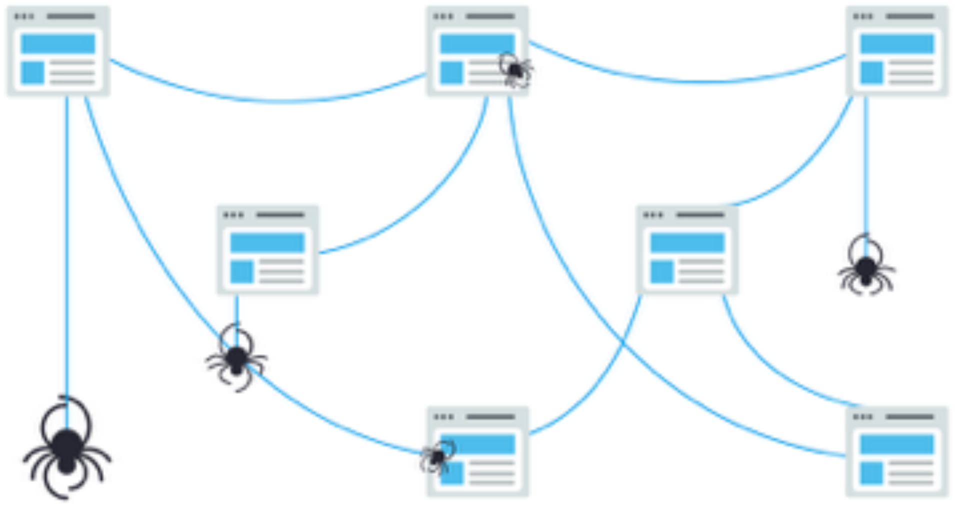
Will public transport continue to run? info-coronavirus.be

Which droplets are the most dangerous? vrt.be

Search Engine Methodology

How do search engines work?

- **Crawl**: Scour the Internet for content, looking over the code/content for each URL they find.
- **Index**: Store and organize the content found during the crawling process. Once a page is in the index, it's in the running to be displayed as a result to relevant queries. This type of index is callec an **inverted index**, because it inverts a page-centric data structure (page->words) to a keyword-centric data structure (word->pages).
- **Rank**: Provide the pieces of content that will best answer a searcher's query, which means that results are ordered by most relevant to least relevant.



Search Engine Solutions

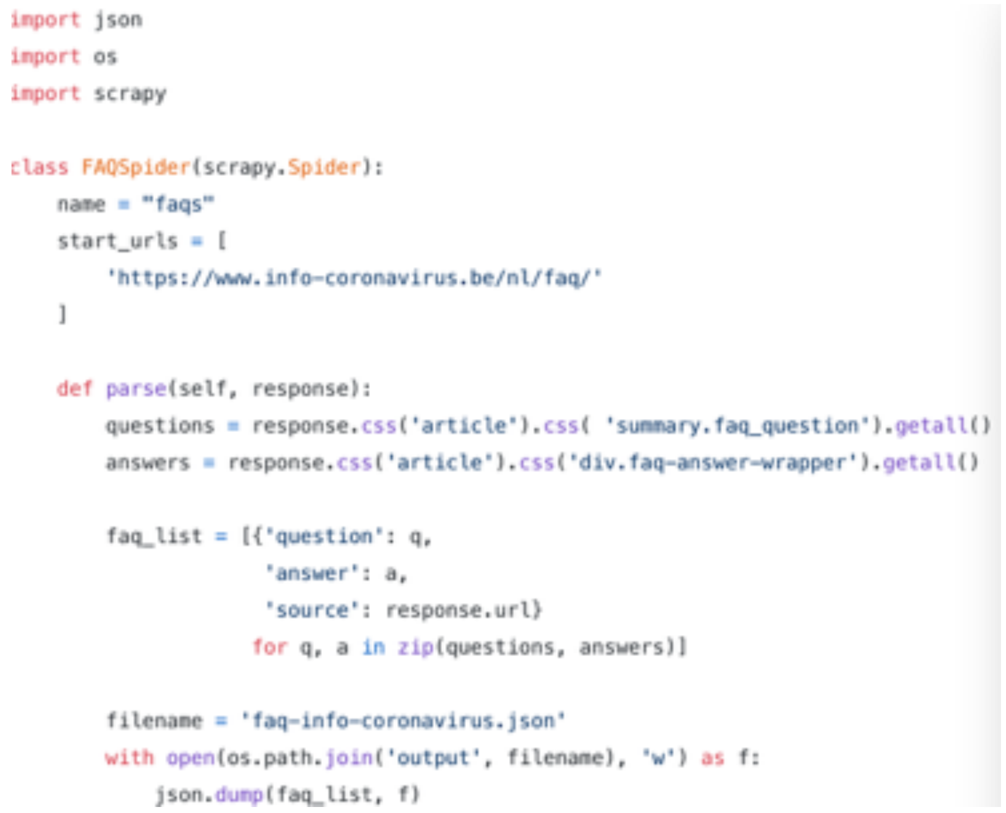
Soir VS ElasticSearch

Lucene Basic Concept

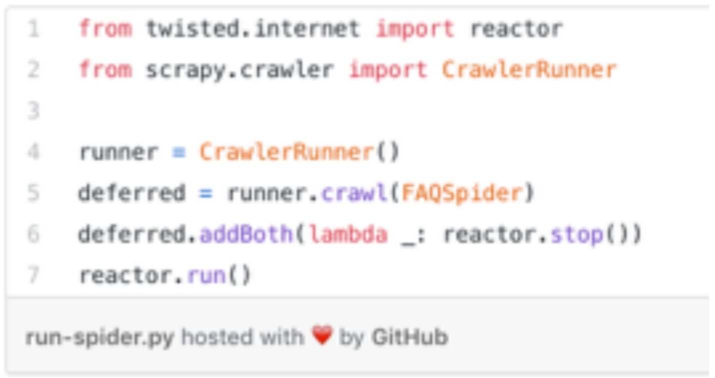
Scraping the data

- Selected a small set of trustworthy FAQs URLs that are being kept up to date by official organizations. Web scraping can easily be done using the python library scrapy. All you need to do is define a “spider” containing the URLs you want to scrape as well as the logic for parsing the underlying HTML.

- Around 500 question-answer pairs saved to local files in json format. Each of the json files contains a list with question-answer pairs along with their source URL.



```python
import json
import os
import scrapy

class FAQSpider(scrapy.Spider):
    name = "faqs"
    start_urls = [
        'https://www.info-coronavirus.be/nl/faq/'
    ]

    def parse(self, response):
        questions = response.css('article').css( 'summary.faq_question').getall()
        answers = response.css('article').css('div.faq-answer-wrapper').getall()

        faq_list = [{'question': q,
                     'answer': a,
                     'source': response.url}
                    for q, a in zip(questions, answers)]

        filename = 'faq-info-coronavirus.json'
        with open(os.path.join('output', filename), 'w') as f:
            json.dump(faq_list, f)
```



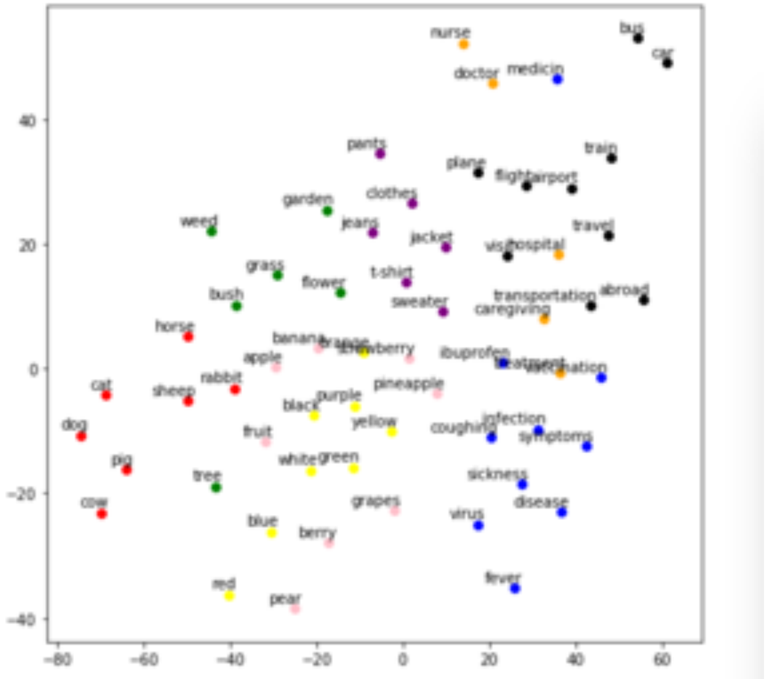
```python
from twisted.internet import reactor
from scrapy.crawler import CrawlerRunner

runner = CrawlerRunner()
deferred = runner.crawl(FAQSpider)
deferred.addBoth(lambda _: reactor.stop())
reactor.run()
```

run-spider.py hosted with ❤ by GitHub

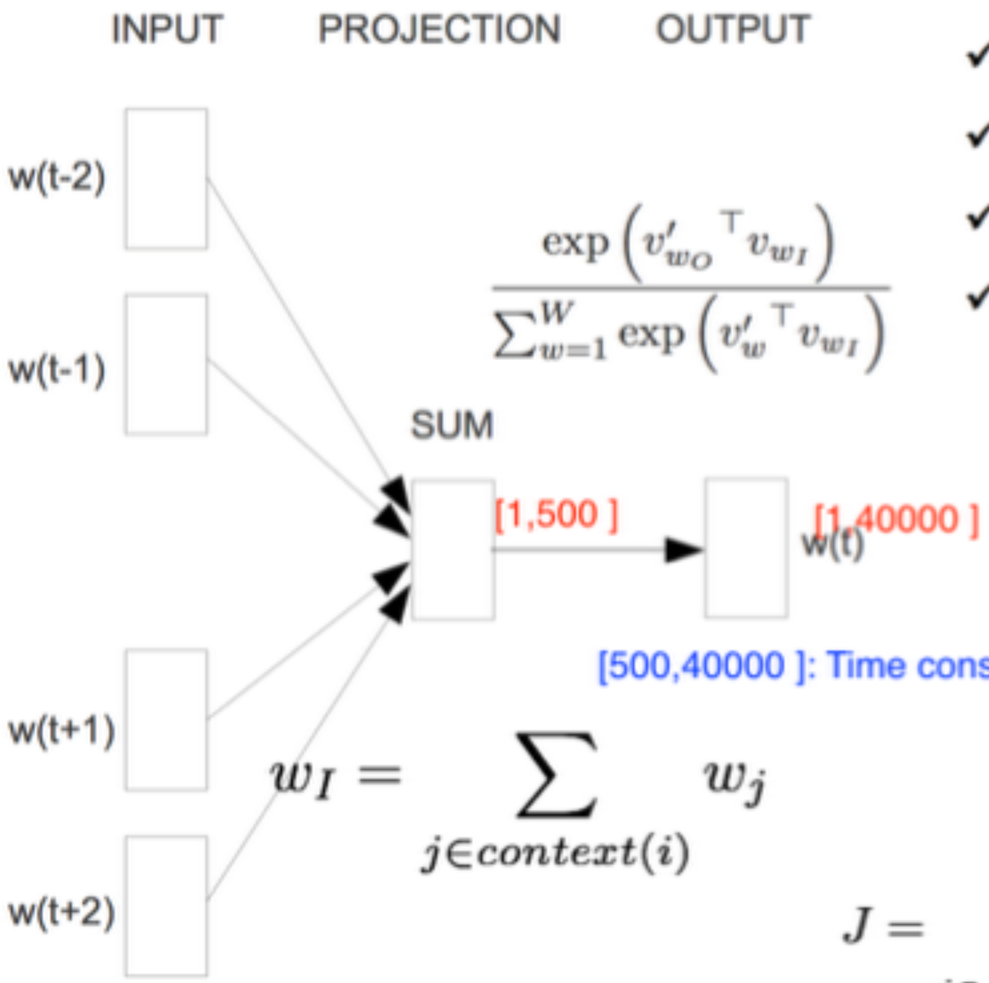
Traditional Search Engine VS **Semantic Search**

Traditionally, search engines are based on **keyword search** where you only get results matching the exact keyword that you typed.

However, you are probably also interested in results containing synonyms for your input keyword or just any related word. In modern NLP, text is usually represented as (high-dimensional) mathematical vectors, also called embeddings, that capture the semantics of the text.



Word2vec: CBOW



$$\frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

[500,40000]: Time consuming

$$w_I = \sum_{j \in context(i)} w_j$$

- ✓ No hidden layer
- ✓ Uses bi-directional contextual window
- ✓ No impacted by the orders of context words (BoW)
- ✓ Input layor: One-hot
- ✓ Projection layor: Sum of the vector

Objective Function

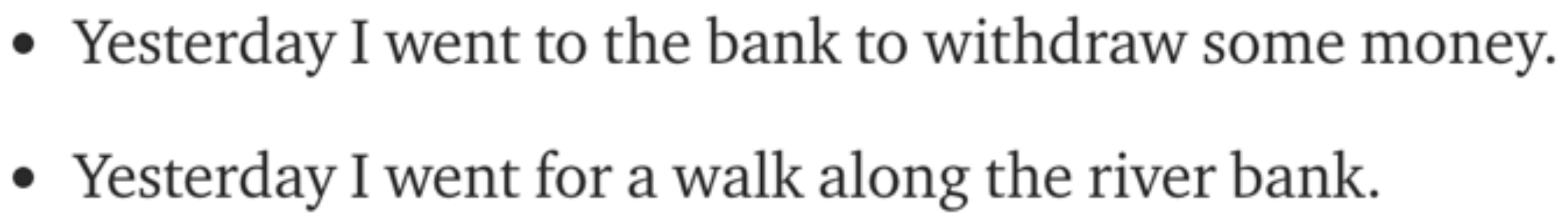
$$J = \sum_{w \in corpus} P(w|context(w))$$

$$J = \sum_{i \in corpus} log\left(\frac{exp(w_i^T \tilde{w}_I)}{\sum_{k=1}^{V} exp(w_i^T \tilde{w}_k)}\right)$$

$$J = \sum_{i \in corpus, j \in context(i)} log\left(\frac{exp(w_i^T \tilde{w}_j)}{\sum_{k=1}^{V} exp(w_i^T \tilde{w}_k)}\right)$$

Sentence Embeddings

- Word embeddings are great but we typically want to capture the meaning of a whole sentence, not just separate words.
- A recent trend in NLP is to learn contextualized embeddings, where the whole sentence (“context”) is taken into account before computing its embedding.



- Yesterday I went to the bank to withdraw some money.
- Yesterday I went for a walk along the river bank.

Universal Sentence Encoder

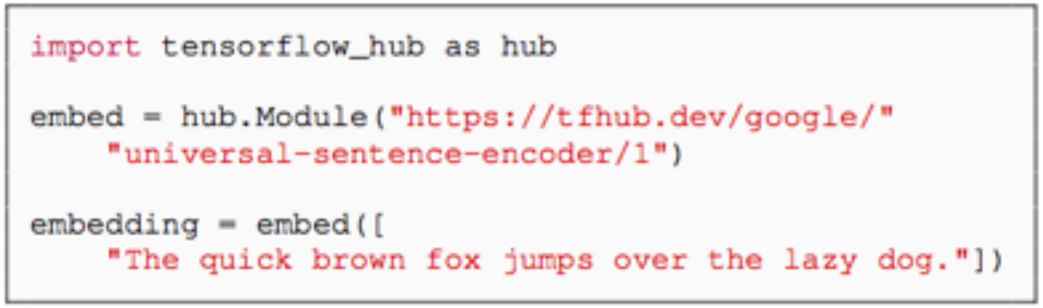
- The Universal Sentence Encoder developed by Google is a state-of-the-art model for computing contextualized sentence embeddings.

Encoders

- **Transformer architecture** targets high accuracy at the cost of greater model complexity and resource consumption.
- **Deep Averaging Network (DAN)** whereby input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network (DNN) to produce sentence embeddings.

Transfer Learning Models

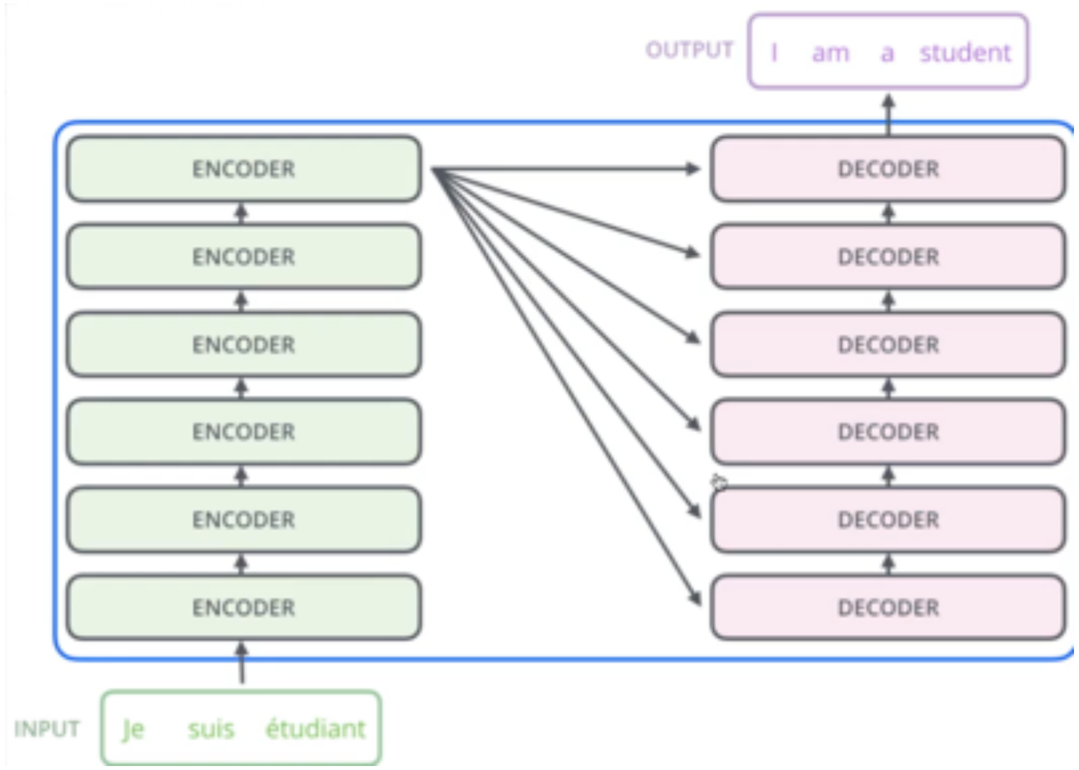
For the pairwise semantic similarity task, we directly assess
the similarity of the sentence embeddings produced by our two encoders. we
first compute the cosine similarity of the two sentence embeddings and then use cosine similarity into an angular distance.

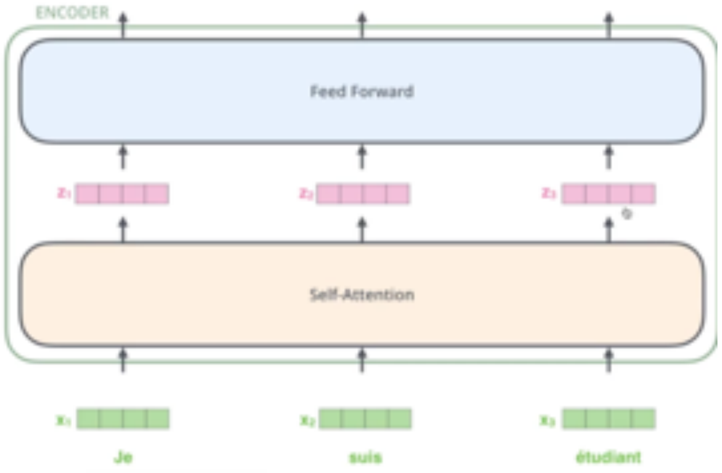


```
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder/1")

embedding = embed([
    "The quick brown fox jumps over the lazy dog."])
```

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \left(1 - \arccos\left(\frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||\ ||\mathbf{v}||}\right)/\pi\right)$$

Transformer Encoder





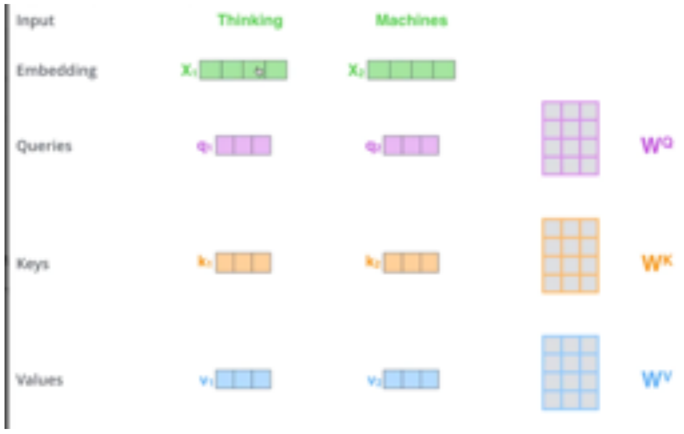
Transformer Encoder

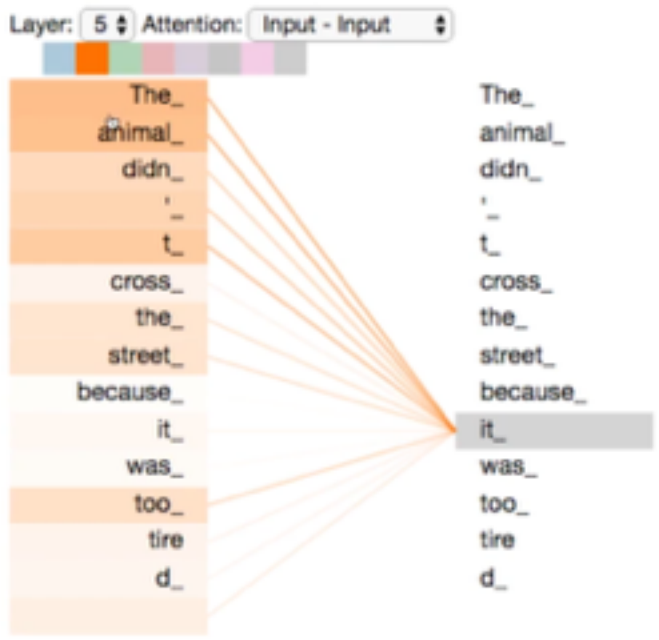
Self-attention的细节

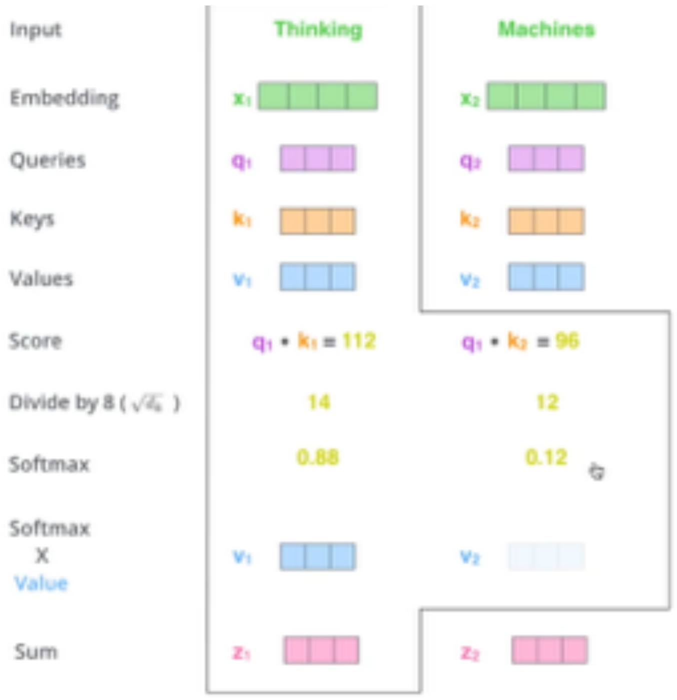
为了实现 self-attention，每个输入的位置需要产生三个向量，分别是 **Query 向量**，**Key 向量**和 **Value 向量**。这些向量都是由输入 embedding 通过三个 matrices （也就是线性变化）产生的。

注意到在Transformer架构中，这些新的向量比原来的输入向量要小，原来的向量是512维，转变后的三个向量都是64维。

第二步是**计算分数**。当我们在用self-attention encode某个位置上的某个单词的时候，我们希望知道这个单词对应的句子上其他单词的分数。其他单词所得到的分数表示了当我们encode当前单词的时候，应该放多少的关注度在其余的每个单词上。又或者说，其他单词和我当前的单词有多大的相关性或者相似性。



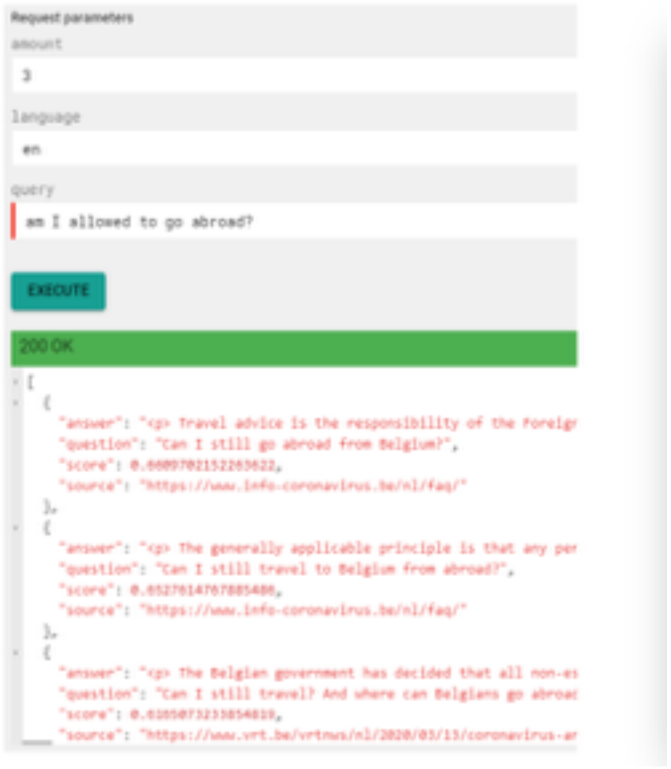




Building the API

- The API is implemented with the Connexion framework of Zalando, which is a Swagger/OpenAPI first framework for Python on top of Flask.

Why Connexion
With Connexion, you write the spec first. Connexion then calls your Python code, handling the mapping from the specification to the code. This incentivizes you to write the specification so that all of your developers can understand what your API does, even before you write a single line of code.



Request parameters

amount: 3

language: en

query: am I allowed to go abroad?

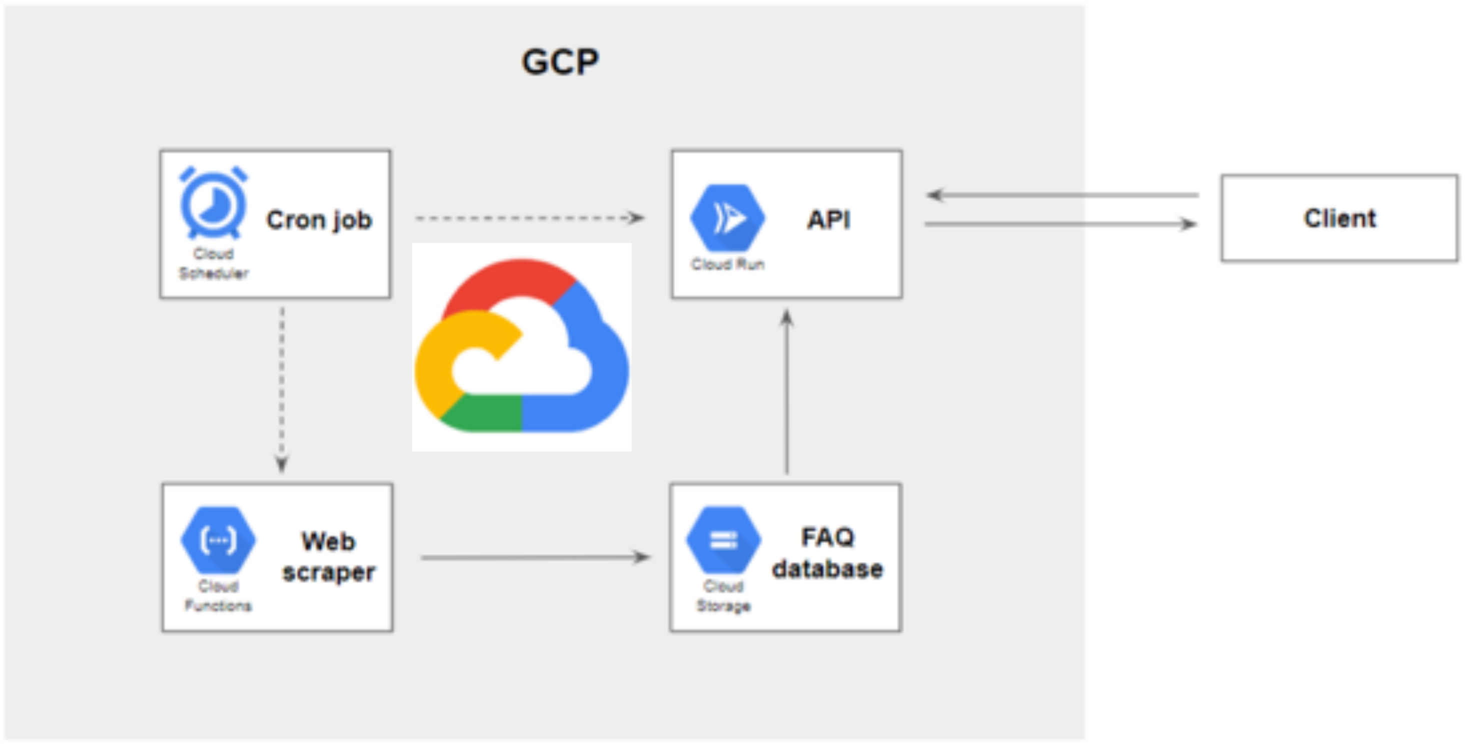
200 OK

```
[
  {
    "answer": "<p> Travel advice is the responsibility of the Foreig
    "question": "Can I still go abroad from Belgium?",
    "score": 0.6609702152263622,
    "source": "https://www.info-coronavirus.be/nl/faq/"
  },
  {
    "answer": "<p> The generally applicable principle is that any per
    "question": "Can I still travel to Belgium from abroad?",
    "score": 0.6527614767885486,
    "source": "https://www.info-coronavirus.be/nl/faq/"
  },
  {
    "answer": "<p> The Belgian government has decided that all non-es
    "question": "Can I still travel? And where can Belgians go abroac
    "score": 0.0185073233854819,
    "source": "https://www.vrt.be/vrtnws/nl/2020/03/13/coronavirus-ar
```

Building the API

Cloud Run, which is a fully managed compute platform for deploying containerized applications

How to keep the FAQ database up to date over time?


GCP
Cron job
Cloud Scheduler
API
Cloud Run
Client
Web scraper
Cloud Functions
FAQ database
Cloud Storage