tokens_and_embeddings

June 23, 2024

1 Tokenization and Embeddings

Import the libraries we are going to use.

[ ]:
```
# Preprocessing
from unstructured.partition.text import partition_text
from unstructured.cleaners.core import group_broken_paragraphs
from langchain.text_splitter import SentenceTransformersTokenTextSplitter

# tokenization and embedding
from sentence_transformers import SentenceTransformer

# Chroma
import chromadb
from chromadb.utils.embedding_functions import␣
 ↪SentenceTransformerEmbeddingFunction

# embedding projection
import umap.umap_ as umap
import numpy as np
from tqdm import tqdm

# visulalization
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

1.1 Pre-processing

It is a relatively large .txt file with impracitcal paragraph splitting. We group the broken parapraphs together into chunks of 1500 characters, which correspond to roughly 1 actual paragraph.

Additionally, we want the partition to later fit into the token splitter. The token splitter we will use has a max input length of 128 tokens. German has a token word ratio of roughly 2.1:1. The average German word has 6.3 characters.

128 / 2.1 * 6.3 = 384 characters

We are can increase

[ ]:
```
elements = partition_text('data/Stein.txt',␣
 ↪paragraph_grouper=group_broken_paragraphs, max_partition=384)
element_strings = [str(el) for el in elements]
print("\n\n".join([el for el in element_strings][:5]))
print("The book has been split into " + str(len(element_strings)) + " chunks.")
print("An element is " + str(len(str(element_strings[0]))) + " characters long.
 ↪")
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\dimit\AppData\Roaming\nltk_data…
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\dimit\AppData\Roaming\nltk_data…
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

```
Mr und Mrs Dursley im Ligusterweg Nummer 4 waren stolz darauf, ganz und gar
normal zu sein, sehr stolz sogar. Niemand wäre auf die Idee gekommen, sie
könnten sich in eine merkwürdige und geheimnisvolle Geschichte verstricken, denn
mit solchem Unsinn wollten sie nichts zu tun haben. Mr Dursley war Direktor
einer Firma namens Grunnings, die Bohrmaschinen herstellte.

Er war groß und bullig und hatte fast keinen Hals, dafür aber einen sehr großen
Schnurrbart. Mrs Dursley war dünn und blond und besaß doppelt so viel Hals, wie
notwendig gewesen wäre, was allerdings sehr nützlich war, denn so konnte sie den
Hals über den Gartenzaun recken und zu den Nachbarn hinüberspähen.

Die Dursleys hatten einen kleinen Sohn namens Dudley und in ihren Augen gab es
nirgendwo einen prächtigeren Jungen.

Die Dursleys besaßen alles, was sie wollten, doch sie hatten auch ein Geheimnis,
und dass es jemand aufdecken könnte, war ihre größte Sorge. Einfach unerträglich
wäre es, wenn die Sache mit den Potters herauskommen würde. Mrs Potter war die
Schwester von Mrs Dursley; doch die beiden hatten sich schon seit etlichen
Jahren nicht mehr gesehen.

Mrs Dursley behauptete sogar, dass sie gar keine Schwester hätte, denn diese und
deren Nichtsnutz von einem Mann waren so undursleyhaft, wie man es sich nur
denken konnte. Die Dursleys schauderten beim Gedanken daran, was die Nachbarn
sagen würden, sollten die Potters eines Tages in ihrer Straße aufkreuzen.
The book has been split into 1669 chunks.
An element is 366 characters long.
```

1.2 Chunk refinement

We now make sure that each chunk fits into the input lenght of the model we will use to embed our vector database.

[ ]:
```
token_splitter = SentenceTransformersTokenTextSplitter(chunk_overlap=0,␣
↪tokens_per_chunk=128, model_name="paraphrase-multilingual-MiniLM-L12-v2")

token_split_texts = []
for text in element_strings:
    token_split_texts += token_splitter.split_text(text)

print(f"\nTotal chunks: {len(token_split_texts)}")
print(token_split_texts[0])
```

```
modules.json:   0%|          | 0.00/229 [00:00<?, ?B/s]

c:\Users\dimit\.conda\envs\AppliedDataScience\Lib\site-
packages\huggingface_hub\file_download.py:157: UserWarning: `huggingface_hub`
cache-system uses symlinks by default to efficiently store duplicated files but
your machine does not support them in
C:\Users\dimit\.cache\huggingface\hub\models--sentence-transformers--paraphrase-
multilingual-MiniLM-L12-v2. Caching files will still work but in a degraded
version that might require more space on your disk. This warning can be disabled
by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more
details, see https://huggingface.co/docs/huggingface_hub/how-to-
cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to
run Python as an administrator. In order to see activate developer mode, see
this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-
your-device-for-development
  warnings.warn(message)

config_sentence_transformers.json:   0%|          | 0.00/122 [00:00<?, ?B/s]

README.md:   0%|          | 0.00/4.12k [00:00<?, ?B/s]

sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]

c:\Users\dimit\.conda\envs\AppliedDataScience\Lib\site-
packages\huggingface_hub\file_download.py:1132: FutureWarning: `resume_download`
is deprecated and will be removed in version 1.0.0. Downloads always resume when
possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(

config.json:   0%|          | 0.00/645 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/471M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/480 [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/9.08M [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/239 [00:00<?, ?B/s]

1_Pooling/config.json:   0%|          | 0.00/190 [00:00<?, ?B/s]


Total chunks: 1669
```

```
Mr und Mrs Dursley im Ligusterweg Nummer 4 waren stolz darauf, ganz und gar
normal zu sein, sehr stolz sogar. Niemand wäre auf die Idee gekommen, sie
könnten sich in eine merkwürdige und geheimnisvolle Geschichte verstricken, denn
mit solchem Unsinn wollten sie nichts zu tun haben. Mr Dursley war Direktor
einer Firma namens Grunnings, die Bohrmaschinen herstellte.
```

1.3 Language Model Analysis

1.3.1 Does it matter what language your text has when deciding for an embedding model?

We tokenize the chunks now with the tokenizer of the embedding model we will use.

- The model uses SentencePiece tokenisation, which is a bit different from WordPiece or Byte Pair Encoding.
- We still see sub words. White spaces are highlighted with underscores.
- Sentence boundaries are marked with <s>
- Subwords and single characters are recognizable
- Is more on the language-agnostic side, as it does not rely on white spaces to separate words.

[ ]:
```
model_name = "paraphrase-multilingual-MiniLM-L12-v2"
model = SentenceTransformer(model_name)
tokenized_chunks = []
for i, text in enumerate(token_split_texts[:10]):
    # Tokenize each chunk
    encoded_input = model.tokenizer(text, padding=True, truncation=True,␣
 ↪max_length=128, return_tensors='pt')
    # Convert token IDs back to tokens
    tokens = model.tokenizer.
 ↪convert_ids_to_tokens(encoded_input['input_ids'][0].tolist())
    tokenized_chunks.append(tokens)
    print(f"Chunk {i}: {tokens}")
```

```
Chunk 0: ['<s>', ' Mr', ' und', ' Mrs', ' Dur', 's', 'ley', ' im', ' Lig', 'u',
'ster', 'weg', ' Nummer', ' 4', ' waren', ' stolz', ' darauf', ',', ' ganz',
' und', ' gar', ' normal', ' zu', ' sein', ',', ' sehr', ' stolz', ' sogar',
'.', ' Niemand', ' wäre', ' auf', ' die', ' Idee', ' gekommen', ',', ' sie',
' könnten', ' sich', ' in', ' eine', ' merk', 'würdig', 'e', ' und', ' geheim',
'nis', 'volle', ' Geschichte', ' vers', 'trick', 'en', ',', ' denn', ' mit',
' solche', 'm', ' Un', 'sinn', ' wollte', 'n', ' sie', ' nichts', ' zu', ' tun',
' haben', '.', ' Mr', ' Dur', 's', 'ley', ' war', ' Direktor', ' einer',
' Firma', ' namen', 's', ' Grunn', 'ings', ',', ' die', ' Boh', 'r', 'maschine',
'n', ' herstel', 'lte', '.', '</s>']
Chunk 1: ['<s>', ' Er', ' war', ' groß', ' und', ' bul', 'lig', ' und',
' hatte', ' fast', ' keinen', ' Hal', 's', ',', ' dafür', ' aber', ' einen',
' sehr', ' großen', ' Sch', 'nur', 'r', 'bart', '.', ' Mrs', ' Dur', 's', 'ley',
' war', ' dün', 'n', ' und', ' blond', ' und', ' be', 'sa', 'ß', ' doppelt',
' so', ' viel', ' Hal', 's', ',', ' wie', ' notwendig', ' gewesen', ' wäre',
',', ' was', ' allerdings', ' sehr', ' nützlich', ' war', ',', ' denn', ' so',
```

' konnte', ' sie', ' den', ' Hal', 's', ' über', ' den', ' Garten', 'za', 'un',
' re', 'cken', ' und', ' zu', ' den', ' Nachbar', 'n', ' hin', 'über', 's',
'pä', 'hen', '.', '</s>']
Chunk 2: ['<s>', ' Die', ' Dur', 's', 'ley', 's', ' hatten', ' einen',
' kleinen', ' Sohn', ' namen', 's', ' Du', 'd', 'ley', ' und', ' in', ' ihren',
' Augen', ' gab', ' es', ' ni', 'r', 'gend', 'wo', ' einen', ' prä', 'chtige',
'ren', ' Jung', 'en', '.', '</s>']
Chunk 3: ['<s>', ' Die', ' Dur', 's', 'ley', 's', ' be', 'sa', 'ßen', ' alles',
',', ' was', ' sie', ' wollte', 'n', ',', ' doch', ' sie', ' hatten', ' auch',
' ein', ' Geheimnis', ',', ' und', ' dass', ' es', ' jemand', ' auf', 'de',
'cken', ' könnte', ',', ' war', ' ihre', ' größte', ' Sor', 'ge', '.',
' Einfach', ' un', 'er', 'träg', 'lich', ' wäre', ' es', ',', ' wenn', ' die',
' Sache', ' mit', ' den', ' Potter', 's', ' heraus', 'kommen', ' würde', '.',
' Mrs', ' Potter', ' war', ' die', ' Schwester', ' von', ' Mrs', ' Dur', 's',
'ley', ';', ' doch', ' die', ' beiden', ' hatten', ' sich', ' schon', ' seit',
' et', 'lichen', ' Jahren', ' nicht', ' mehr', ' gesehen', '.', '</s>']
Chunk 4: ['<s>', ' Mrs', ' Dur', 's', 'ley', ' behauptet', 'e', ' sogar', ',',
' dass', ' sie', ' gar', ' keine', ' Schwester', ' hätte', ',', ' denn',
' diese', ' und', ' deren', ' Nicht', 's', 'nut', 'z', ' von', ' einem',
' Mann', ' waren', ' so', ' und', 'urs', 'ley', 'haft', ',', ' wie', ' man',
' es', ' sich', ' nur', ' denken', ' konnte', '.', ' Die', ' Dur', 's', 'ley',
's', ' ', 'schau', 'der', 'ten', ' beim', ' Gedanken', ' daran', ',', ' was',
' die', ' Nachbar', 'n', ' sagen', ' würden', ',', ' sollten', ' die',
' Potter', 's', ' eines', ' Tages', ' in', ' ihrer', ' Straße', ' auf', 'kre',
'u', 'zen', '.', '</s>']
Chunk 5: ['<s>', ' Die', ' Dur', 's', 'ley', 's', ' wusste', 'n', ',', ' dass',
' auch', ' die', ' Potter', 's', ' einen', ' kleinen', ' Sohn', ' hatten', ',',
' doch', ' den', ' hatten', ' sie', ' nie', ' gesehen', '.', ' Auch', ' dieser',
' Jung', 'e', ' war', ' ein', ' guter', ' Grund', ',', ' sich', ' von', ' den',
' Potter', 's', ' fer', 'n', 'zuhalten', ';', ' mit', ' einem', ' solchen',
' Kind', ' sollte', ' ihr', ' Du', 'd', 'ley', ' nicht', ' in', ' Ber', 'ü',
'hrung', ' kommen', '.', '</s>']
Chunk 6: ['<s>', ' Als', ' Mr', ' und', ' Mrs', ' Dur', 's', 'ley', ' an',
' dem', ' tr', 'üb', 'en', ' und', ' grau', 'en', ' Dienstag', ',', ' an',
' dem', ' unsere', ' Geschichte', ' beginnt', ',', ' die', ' Augen', ' auf',
'sch', 'lug', 'en', ',', ' war', ' an', ' dem', ' wol', 'ken', 'ver', 'hang',
'en', 'en', ' Himmel', ' draußen', ' kein', ' Vor', 'zeichen', ' der', ' merk',
'würdig', 'en', ' und', ' geheim', 'nis', 'volle', 'n', ' Dinge', ' zu',
' erkennen', ',', ' die', ' bald', ' überall', ' im', ' Land', ' gesch', 'e',
'hen', ' sollten', '.', '</s>']
Chunk 7: ['<s>', ' Mr', ' Dur', 's', 'ley', ' su', 'mm', 'te', ' vor', ' sich',
' hin', ' und', ' such', 'te', ' sich', ' für', ' die', ' Arbeit', ' seine',
' langweilig', 'ste', ' K', 'rawat', 'te', ' aus', ',', ' und', ' Mrs', ' Dur',
's', 'ley', ' schw', 'at', 'zte', ' munt', 'er', ' vor', ' sich', ' hin', ',',
' während', ' sie', ' mit', ' dem', ' sch', 'rei', 'enden', ' Du', 'd', 'ley',
' range', 'lte', ' und', ' ihn', ' in', ' seinen', ' Hoch', 'stuhl', ' z', 'w',
'äng', 'te', '.', ' Kein', 'er', ' von', ' ihnen', ' sah', ' den', ' riesige',
'n', ' Wald', 'ka', 'uz', ' am', ' Fenster', ' vorbei', 'f', 'liegen', '.',

```
'</s>']
Chunk 8: ['<s>', ' Um', ' halb', ' ne', 'un', ' ', 'griff', ' Mr', ' Dur', 's',
'ley', ' nach', ' der', ' Ak', 'tenta', 'sche', ',', ' gab', ' seiner', ' Frau',
' einen', ' Sch', 'mat', 'z', ' auf', ' die', ' Wang', 'e', ' und', ' versucht',
'e', ' es', ' auch', ' bei', ' Du', 'd', 'ley', ' mit', ' einem', ' Ab',
'schied', 'sku', 's', 's', '.', ' Der', ' ging', ' jedoch', ' dane', 'ben', ',',
' weil', ' Du', 'd', 'ley', ' gerade', ' einen', ' W', 'utan', 'fall', ' hatte',
' und', ' die', ' Wände', ' mit', ' seinem', ' Haf', 'er', 'bre', 'i', ' be',
'war', 'f', '.', ' »', 'K', 'lein', 'er', ' Sch', 'ling', 'el', '«,', ' glu',
'cks', 'te', ' Mr', ' Dur', 's', 'ley', ',', ' während', ' er', ' nach',
' draußen', ' ging', '.', '</s>']
Chunk 9: ['<s>', ' Er', ' setzt', 'e', ' sich', ' in', ' den', ' Wagen', ' und',
' ', 'fuhr', ' ', 'rück', 'wärts', ' die', ' Ein', 'fahrt', ' zu', ' Nummer',
' 4', ' hinaus', '.', ' An', ' der', ' Straßen', 'e', 'cke', ' fiel', ' ihm',
' zum', ' ersten', ' Mal', ' etwas', ' Merk', 'würdig', 'es', ' auf', ' -',
' eine', ' Kat', 'ze', ',', ' die', ' eine', ' Straßen', 'karte', ' studier',
'te', '.', ' Einen', ' Moment', ' war', ' Mr', ' Dur', 's', 'ley', ' nicht',
' klar', ',', ' was', ' er', ' gesehen', ' hatte', ' -', ' dann', ' ', 'wand',
'te', ' er', ' ras', 'ch', ' den', ' Kopf', ' zurück', ',', ' um', ' noch',
' einmal', ' hinzu', 'schau', 'en', '.', '</s>']
```

Try the same now with a model that uses a SentenciePiece tokenizer.

You should notice: - White spaces have been removed - Words that have been split into words can be connected with `##` - First tries to determine word boundaries like byte-pair encoding. - The start of a sentence is marked with `[CLS]`

[ ]:
```
model_name = "Sahajtomar/German-semantic"
```

1.4 From Token to Embedding

Notice: - our text snippet has 110 tokens - the embedding has 384 dimensions - When calculating the embedding, the embedding model first calculates the 384 dimensional embedding for each individual token - depending on the model the individual vectors are then averaged, maxed or they take the embedding for the sentence boundary marker. - This allows us to end up with just one rather than 110 384 dimensional vectors per chunk

[ ]:
```
# raw text
print(token_split_texts[10])

# tokens
model_name = "paraphrase-multilingual-MiniLM-L12-v2"
model = SentenceTransformer(model_name)
print("Number of tokens: ", len(model.tokenizer(token_split_texts[10],␣
 ↪padding=True, truncation=True, max_length=128, return_tensors='pt')[0]))

# Embedding
embedding_function =␣
 ↪SentenceTransformerEmbeddingFunction(model_name="paraphrase-multilingual-MiniLM-L12-v2")
```

```
print(embedding_function([token_split_texts[10]]))
print("Vector dimensions: ",␣
 ↪len(embedding_function([token_split_texts[10]])[0]))
```

```
An der Einbiegung zum Ligusterweg stand eine getigerte Katze, aber eine
Straßenkarte war nicht zu sehen. Woran er nur wieder gedacht hatte! Das musste
eine Sinnestäuschung gewesen sein. Mr Dursley blinzelte und starrte die Katze
an. Die Katze starrte zurück. Während Mr Dursley um die Ecke bog und die Straße
entlangfuhr, beobachtete er die Katze im Rückspiegel.
Number of tokens:  96
[[0.4442093074321747, -0.11239522695541382, -0.021843602880835533,
0.18790219724178314, 0.0733574852347374, -0.08271972090005875,
0.22475753724575043, -0.004552755039185286, -0.15135742723941803,
0.03984015807509422, 0.3971920907497406, 0.07182806730270386,
-0.07458589971065521, 0.15222476422786713, -0.13651028275489807,
-0.007156126201152801 5, 0.02212478034198284, -0.3487188518047333,
-0.09613367915153503, 0.1602179855108261, 0.08875303715467453,
-0.1491146832704544, -0.23509694635868073, -0.033389199525117874,
-0.06730195134878159, -0.23179171979427338, -0.3984537422657013,
0.08470582216978073, -0.38942578434944153, -0.026574520394206047,
-0.2306361198425293, -0.3309357464313507, -0.04117792844772339,
0.2494991272687912, -0.013703645206987858, -0.214259073138237,
0.2964957058429718, 0.03970691189169884, 0.07463973760604858,
0.036332469433546066, -0.004123234655708075, -0.3637259006500244,
0.060233235359191895, 0.009575878269970417, 0.17679546773433685,
-0.15109775960445404, 0.14806461334228516, -0.040711451321840286,
0.13134396076202393, -0.28984910249710083, -0.13684560358524323,
-0.01631281152367592, -0.3319548964500427, -0.30862709879875183,
-0.23156605660915375, 0.2759372889995575, 0.10974418371915817,
0.18736742436885834, 0.18461234867572784, -0.08211326599121094,
-0.2990492582321167, 0.02410164475440979, -0.1928328275680542,
0.04727610945701599, 0.24110008776187897, -0.1856653243303299,
-0.0006077839061617851, -0.4203832447528839, 0.17572547495365143,
0.15977983176708221, 0.19310976564884186, 0.09549807757139206,
-0.06690941005945206, -0.2307504415512085, 0.11203309148550034,
-0.060560375452041626, 0.11698991060256958, 0.3355703055858612,
-0.1531236171722412, -0.4496874511241913, -0.06532737612724304,
0.10167115181684494, -0.21835416555404663, 0.1764851212501526,
-0.07670817524194717, 0.12331197410821915, -0.052977096289396286,
-0.10834445804357529, -0.15764884650707245, -0.21712340414524078,
0.29644575715065, -0.24966125190258026, 0.21371889114379883,
0.05066882446408272, 0.022307664155960083, -0.02184872329235077,
0.3570626676082611, -0.17376381158828735, -0.04547819122672081,
-0.05577504634857178, 0.16219396889209747, 0.22843801975250244,
0.06705933064222336, 0.1367255002260208, -0.014920968562364578,
-0.07231545448303223, 0.018612653017044067, -0.02683965303003788,
-0.019379563629627228, -0.13539643585681915, 0.28408583998680115,
-0.0459739975631237, 0.2755991518497467, 0.05630357563495636,
```

0.041246358305215836, -0.010217429138720036, -0.22804109752178192,
0.11074649542570114, 0.04008074477314949, 0.032205186784267426,
0.03387409448623657, -0.006035750266164541, 0.07646726071834564,
0.020425239577889442, -0.2754240930080414, 0.029243653640151024,
0.4402446746826172, -0.008684985339641571, -0.011494108475744724,
0.05974060669541359, -0.13447849452495575, -0.14647690951824188,
-0.036596428602933884, -0.23514501750469208, -0.11782566457986832,
0.2100502997636795, 0.12617872655391693, 0.1939830780029297,
-0.19606952369213104, -0.052695389837026596, 0.26370230317115784,
-0.08648452162742615, -0.11128079146146774, 0.2911303639411926,
-0.008859974332153797, -0.024747543036937714, 0.004450606182217598,
-0.13790254294872284, 0.07762941718101501, 0.0765383169054985,
0.17195285856723785, -0.3691389262676239, -0.09507167339324951,
0.2678404748439789, -0.14080527424812317, 0.11477276682853699,
0.0610039196908474, 0.32552066445350647, -0.17462323606014252,
0.30339714884757996, 0.07645165175199509, 0.14626210927963257,
-0.19331218302249908, 0.1043158546090126, 0.12488213926553726,
0.24321453273296356, -0.0256069153547287, 0.0038662180304527283,
0.13715460896492004, 0.0021158221643418074, 0.08417817950248718,
-0.254315584897995, -0.07878384739160538, 0.0435127355158329,
-0.3796602487564087, -0.09919441491365433, -0.34754350781440735,
0.06763976067304611, 0.22582338750362396, -0.14662449061870575,
0.05683073773980141, 0.028181910514831543, -0.28160277009010315,
0.10074469447135925, 0.06497708708047867, 0.161836639046669, -0.107505202293396,
0.1675006002187729, 0.25197166204452515, 0.00931733287870884,
0.4472598135471344, -0.05382324382662773, 0.12129238247871399,
0.016817063093185425, -0.17530763149261475, 0.05139319971203804,
0.01444042194634676, 0.30354586243629456, 0.17887072265148163,
0.17695295810699463, -0.28927144408226013, -0.37741807103157043,
-0.004052742850035429, -0.09164746850728989, -0.3405623137950897,
-0.2782965898513794, 0.07947386056184769, -0.1250874549150467,
-0.021662600338459015, 0.27398112416267395, 0.04509760066866875,
-0.16061340272426605, 0.21170783042907715, -0.2592691481113434,
0.19524341821670532, -0.25016435980796814, -0.11335618048906326,
0.16004516184329987, -0.3216705024242401, 0.16347701847553253,
-0.09445489197969437, -0.2212555855512619, 0.1599484533071518,
-0.19827775657176971, -0.14961038529872894, 0.15830761194229126,
0.2534123957157135, -0.051083121448755264, -0.426722913980484,
-0.03772852197289467, -0.006885936949402094, -0.15611161291599274,
0.03843889385461807, 0.007386287208646536, -0.23328979313373566,
0.3737861216068268, 0.1605360507965088, 0.04781655594706535,
0.15935637056827545, 0.11798747628927231, -0.2996613681316376,
-0.3712824583053589, -0.16268320381641388, 0.10678284615278244,
-0.19033408164978027, -0.2617511451244354, -0.3410102427005768,
0.2486533671617508, 0.04968784376978874, 0.21287624537944794, 0.521833598613739,
0.04090217873454094, -0.14300252497196198, -0.13416273891925812,
-0.07179442793130875, -0.20225520431995392, 0.07623917609453201,
0.1484837681055069, 0.37096357345581055, 0.4777900278568268,

0.06070389971137047, 0.15740786492824554, -0.028570739552378654,
-0.051502469927072525, -0.23750530183315277, 0.03072565793991089,
0.0655737966299057, -0.11887698620557785, 0.05402815341949463,
0.028778458014130592, -0.0773126631975174, -0.010002643801271915,
-0.16169224679470062, -0.027662532404065132, 0.4182868003845215,
-0.037381771951913834, -0.0192659143358469, 0.09870156645774841,
-0.17133162915706635, 0.2502550184726715, 0.09207817167043686,
0.42265570163726807, 0.03837507218122482, 0.0654749646782875,
-0.1331387609243393, 0.07941035181283951, -0.0244241114705801,
0.13449998199939728, 0.18372970819473267, -0.16423599421977997,
-0.06013859435915947, -0.011708802543580532, 0.2591623365879059,
-0.16800576448440552, 0.1332463175058365, 0.3226630985736847,
-0.2338717132806778, -0.21755707263946533, -0.00012812810018658638,
0.2559112012386322, 0.0873045027256012, 0.03604608029127121,
-0.6438794136047363, 0.04037943854928017, 0.1412113904953003,
-0.05696997046470642, 0.1382497400045395, 0.12919612228870392,
-0.11493667215108871, -0.11225956678390503, -0.10554876923561096,
0.03210672736167908, 0.016930898651480675, 0.12355607748031616,
0.3744324743747711, 0.1942378431558609, 0.21548092365264893,
0.26324358582496643, 0.15926063060760498, -0.07880445569753647,
-0.3122994005680084, -0.10243353992700577, -0.3495723307132721,
0.22335965931415558, 0.19337286055088043, -0.027968456968665123,
0.10551360994577408, -0.2938343286514282, -0.4753105342388153,
0.06251654773950577, -0.34868237376213074, 0.0670929104089737,
-0.26149317622184753, 0.09822653979063034, 0.12847907841205597,
0.16260434687137604, -0.07933434844017029, -0.3707022964954376,
-0.1402813196182251, 0.1800599843263626, 0.05705629289150238,
0.00781711284071207, -0.10873774439096451, -0.14297713339328766,
0.04787926748394966, -0.18275868892669678, -0.2922225594520569,
0.15822000801563263, 0.06446581333875656, 0.009890434332191944,
0.08689210563898087, 0.12584207952022552, -0.4581066071987152,
0.13061030209064484, 0.21423006057739258, 0.22738580405712128,
0.28980782628059387, 0.21892714500427246, 0.4322487413883209,
-0.09107585996389389, 0.08084481954574585, -0.14250318706035614,
-0.12466876953840256, 0.18855242431163788, -0.2929902672767639,
0.00700552761554718, -0.04901757836341858, -0.1168389618396759,
0.15209980309009552, -0.0396592952311039, -0.3060619533061981,
0.047089219093322754, -0.16502155363559723, -0.16567756235599518,
0.019493699073791504, -0.21986953914165497, 0.0018969712546095252,
0.026521913707256317, -0.13097994029521942, 0.08353570848703384,
-0.04566323757171631, 0.20522983372211456, 0.04422856494784355,
0.19168035686016083]]
Vector dimensions: 384

1.5 Building our Vector Store

1.6

[ ]:
```
chroma_client = chromadb.Client()


embedding_function =␣
 ↪SentenceTransformerEmbeddingFunction(model_name="paraphrase-multilingual-MiniLM-L12-v2")
chroma_collection = chroma_client.create_collection("Steint.txt",␣
 ↪embedding_function=embedding_function)

ids = [str(i) for i in range(len(token_split_texts))]

chroma_collection.add(ids=ids, documents=token_split_texts)
chroma_collection.count()
```

[ ]:
```
1669
```

1.7 Embedding Projections

We retrieve all embeddings from our chroma collection.

- UMAP (Uniform Manifold Approximation and Projection): reduces dimensionanality of a vector to project into a lower dimensionality space. Tyipically 2D or 3D vor visualisations.

[ ]:
```
embeddings = chroma_collection.get(include=['embeddings'])['embeddings']
umap_transform = umap.UMAP(random_state=0, transform_seed=0).fit(embeddings)
```

```
c:\Users\dimit\.conda\envs\AppliedDataScience\Lib\site-
packages\umap\umap_.py:1945: UserWarning: n_jobs value 1 overridden to 1 by
setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
```

1.7.1 Function to apply the UMAP transformation to our data

We will need to tranform multiple vectors

[ ]:
```
def project_embeddings(embeddings, umap_transform):
    umap_embeddings = np.empty((len(embeddings),2))
    for i, embedding in enumerate(tqdm(embeddings)):
        umap_embeddings[i] = umap_transform.transform([embedding])
    return umap_embeddings
```

Create a query.

[ ]:
```
query = "Hogwarts"
```

Embed and project the query into a 2-dimensional space.

[ ]:
```
original_query_embedding = embedding_function(query)
project_original_query = project_embeddings(original_query_embedding,␣
↪umap_transform)
```

```
100%|      | 8/8 [00:14<00:00,  1.80s/it]
```

Query the chroma_collection for documents related to “Hogwarts” and retrieve the top 5 results

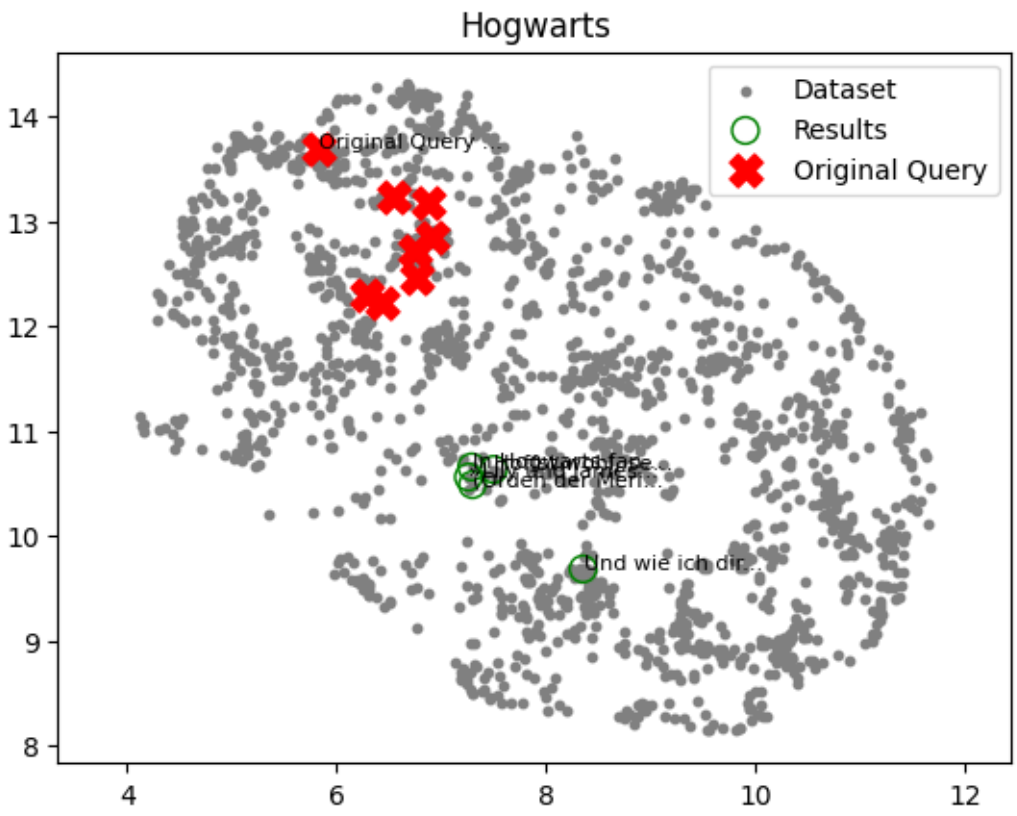
- Extract the embeddings from the results
- Flatten the list of embeddings
- Project the result embeddings using the umap_transform
- Project the dataset embeddings using the umap_transform

[ ]:
```
results = chroma_collection.query(query_texts=["Hogwarts"], n_results=5,␣
↪include=['documents', 'embeddings'])
print(results['documents'][0])
result_embeddings = results['embeddings']
result_embeddings = [item for sublist in result_embeddings for item in sublist]
projected_result_embeddings = project_embeddings(result_embeddings,␣
↪umap_transform)
projected_dataset_embeddings = project_embeddings(embeddings, umap_transform)
```

```
['In Hogwarts fangen sie alle ganz von vorne an, es wird dir sicher gut gehen.
Sei einfach du selbst. Ich weiß, es ist schwer. Du bist auserwählt worden und
das ist immer schwer. Aber du wirst eine tolle Zeit in Hogwarts verbringen - wie
ich damals - und heute noch, um genau zu sein.«', 'Hoffst wohl, selber Wildhüter
zu werden, wenn du mit Hogwarts fertig bist - diese Hütte von Hagrid muss dir
wie ein Palast vorkommen im Vergleich zu dem, was du von deiner Familie gewöhnt
bist.« Ron stürzte sich auf Malfoy und in diesem Moment kam Snape die Treppe
hoch. »WEASLEY!« Ron ließ Malfoys Umhang los.', '(Orden der Merlin, Erster
Klasse, Großz., Hexenmst. Ganz hohes Tier, Internationale Vereinig. d. Zauberer)
Sehr geehrter Mr Potter, wir freuen uns, Ihnen mitteilen zu können, dass Sie an
der Hogwarts-Schule für Hexerei und Zauberei aufgenommen sind. Beigelegt finden
Sie eine Liste aller benötigten Bücher und Ausrüstungsgegenstände. Das Schuljahr
beginnt am 1. September.', '»Lily und James Potters Sohn von Hogwarts
fernhalten! Du bist ja verrückt. Sein Name ist vorgemerkt, schon seit seiner
Geburt. Er geht bald auf die beste Schule für Hexerei und Zauberei auf der
ganzen Welt. Nach sieben Jahren dort wird er sich nicht mehr wiedererkennen.',
'Und wie ich dir schon gesagt hab, bin ich der Schlüsselhüter von Hogwarts -
über Hogwarts weißt du natürlich alles.« »Ähm - nein«, sagte Harry. Hagrid sah
schockiert aus. »Tut mir leid«, sagte Harry rasch. »Tut dir leid?«, bellte
Hagrid und wandte sich zu den Dursleys um mit einem Blick, der sie in die
Schatten zurückweichen ließ. »Denen sollte es leidtun.']
```

```
100%|      | 5/5 [00:05<00:00,  1.01s/it]
100%|      | 1669/1669 [26:26<00:00,  1.05it/s]
```

[ ]:
```
def shorten_text(text, max_length=15):
    """ Shortens text to max_length and adds an ellipsis if the text was␣
 ↪shortened. """
    return (text[:max_length] + '...') if len(text) > max_length else text

plt.figure()

# Scatter plots
plt.scatter(projected_dataset_embeddings[:, 0], projected_dataset_embeddings[:,␣
 ↪1], s=10, color='gray', label='Dataset')
plt.scatter(projected_result_embeddings[:, 0], projected_result_embeddings[:,␣
 ↪1], s=100, facecolors='none', edgecolors='g', label='Results')
plt.scatter(project_original_query[:, 0], project_original_query[:, 1], s=150,␣
 ↪marker='X', color='r', label='Original Query')

# Assuming result_texts is an array of texts for the results
# result_texts = ['text1', 'text2', ..., 'text5']

for i, text in enumerate(results['documents'][0]):
    if i < len(projected_result_embeddings):
        plt.annotate(shorten_text(text), (projected_result_embeddings[i, 0],␣
 ↪projected_result_embeddings[i, 1]), fontsize=8)

# Assuming you have text for the original query
original_query_text = 'Original Query Text'  # Replace with your actual text␣
 ↪for the original query
plt.annotate(shorten_text(original_query_text), (project_original_query[0, 0],␣
 ↪project_original_query[0, 1]), fontsize=8)

plt.gca().set_aspect('equal', 'datalim')
plt.title('Hogwarts')
plt.legend()
plt.show()
```



1.7.2 3D-Projection

[ ]:
```
# Adjusted UMAP transform for 3D projection
embeddings = chroma_collection.get(include=['embeddings'])['embeddings']
umap_transform_3d = umap.UMAP(n_components=3, random_state=0, transform_seed=0).
 ↪fit(embeddings)

def project_embeddings_3d(embeddings, umap_transform):
    umap_embeddings = np.empty((len(embeddings), 3))
    for i, embedding in enumerate(tqdm(embeddings)):
        umap_embeddings[i] = umap_transform.transform([embedding])[0]
    return umap_embeddings
```

```
c:\Users\dimit\.conda\envs\AppliedDataScience\Lib\site-
packages\umap\umap_.py:1945: UserWarning: n_jobs value 1 overridden to 1 by
setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
```

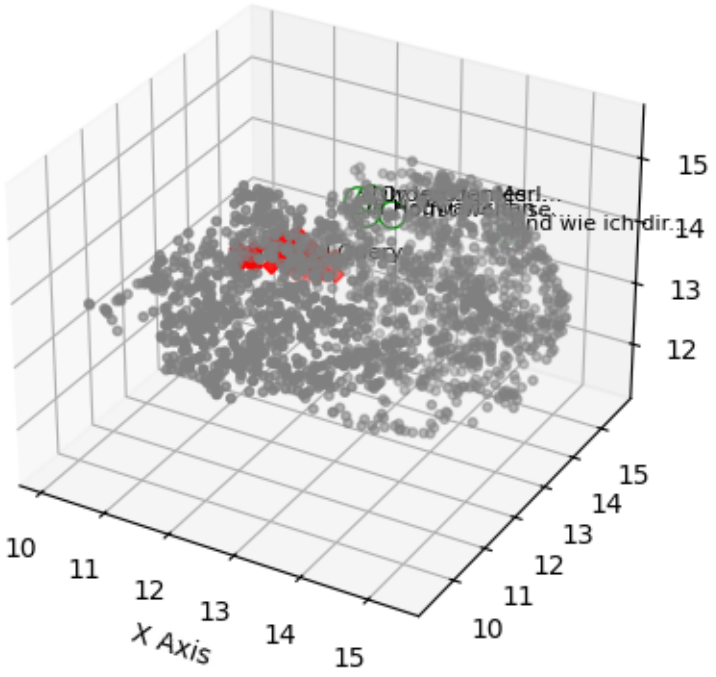
[ ]:
```
project_original_query = project_embeddings_3d(original_query_embedding,␣
 ↪umap_transform_3d)
```

```
projected_result_embeddings = project_embeddings_3d(result_embeddings,␣
 ↪umap_transform_3d)
projected_dataset_embeddings = project_embeddings_3d(embeddings,␣
 ↪umap_transform_3d)
```

```
100%|      | 8/8 [00:07<00:00,  1.07it/s]
100%|      | 5/5 [00:04<00:00,  1.23it/s]
100%|      | 1669/1669 [27:44<00:00,  1.00it/s]
```

[ ]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
projected_dataset_embeddings_3d = projected_dataset_embeddings
projected_result_embeddings_3d = projected_result_embeddings
project_original_query_3d = project_original_query

# Scatter plots
ax.scatter(projected_dataset_embeddings_3d[:, 0],␣
 ↪projected_dataset_embeddings_3d[:, 1], projected_dataset_embeddings_3d[:,␣
 ↪2], s=10, color='gray', label='Dataset')
ax.scatter(projected_result_embeddings_3d[:, 0],␣
 ↪projected_result_embeddings_3d[:, 1], projected_result_embeddings_3d[:, 2],␣
 ↪s=100, facecolors='none', edgecolors='g', label='Results')
ax.scatter(project_original_query_3d[:, 0], project_original_query_3d[:, 1],␣
 ↪project_original_query_3d[:, 2], s=150, marker='X', color='r',␣
 ↪label='Original Query')

# Annotations
for i, text in enumerate(results['documents'][0]):
    if i < len(projected_result_embeddings_3d):
        ax.text(projected_result_embeddings_3d[i, 0],␣
 ↪projected_result_embeddings_3d[i, 1], projected_result_embeddings_3d[i, 2],␣
 ↪shorten_text(text), fontsize=8)

ax.text(project_original_query_3d[0, 0], project_original_query_3d[0, 1],␣
 ↪project_original_query_3d[0, 2], shorten_text(original_query_text),␣
 ↪fontsize=8)

ax.set_xlabel('X Axis')
```

[ ]:
```
Text(0.5, 0, 'X Axis')
```


15
14
13
12
15
14
13
12
11
10
10
11
12
13
14
15
X Axis