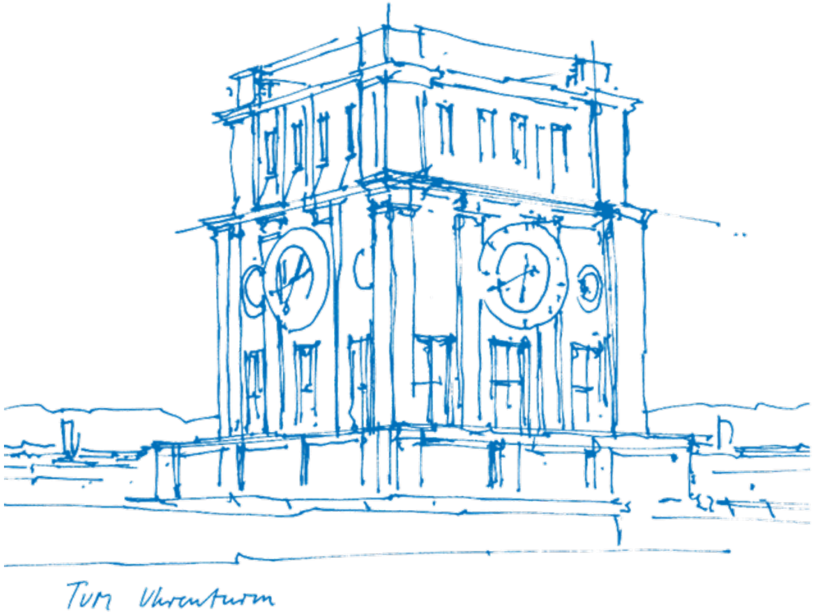
Technical University of Munich

Clustering-Based Sentiment Analysis for Media Agenda Setting

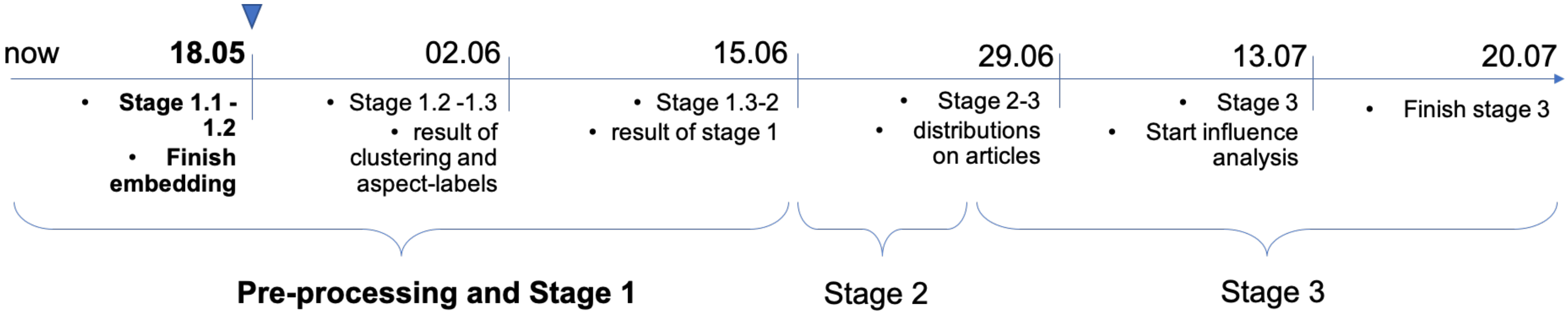
Opinion Lab Group 2.3

Wing Sheung Leung, Qiaoxi Liu

May 18, 2020



Milestones


now
18.05
02.06
15.06
29.06
13.07
20.07
• Stage 1.1 - 1.2
• Finish embedding
• Stage 1.2 -1.3
• result of clustering and aspect-labels
• Stage 1.3-2
• result of stage 1
• Stage 2-3
• distributions on articles
• Stage 3
• Start influence analysis
• Finish stage 3
Pre-processing and Stage 1
Stage 2
Stage 3

Overview

Stage 1: Generate sentence embeddings with our corpus

1.1 Embeddings

XLING sentence-level embeddings
Indexing sentences

1.2 Kmeans and Elbow Method

sklearn.cluster.MiniBatchKMeans
Elbow Method for determining optimal k

Future Plan

Stage 1.1: XLING sentence-level embeddings

Our encoder:

```
en_de_embed = hub.Module("https://tfhub.dev/google/universal-sentence-encoder-xling/en-de/1")
```

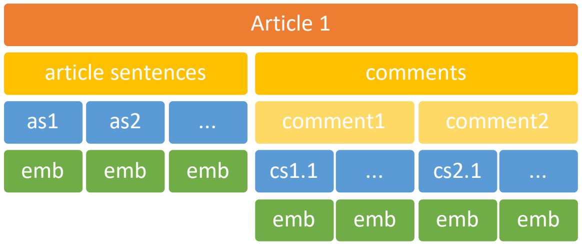


Figure: hierarchy of json

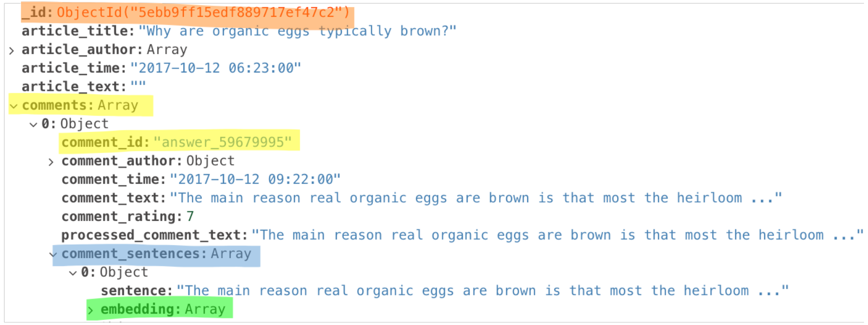
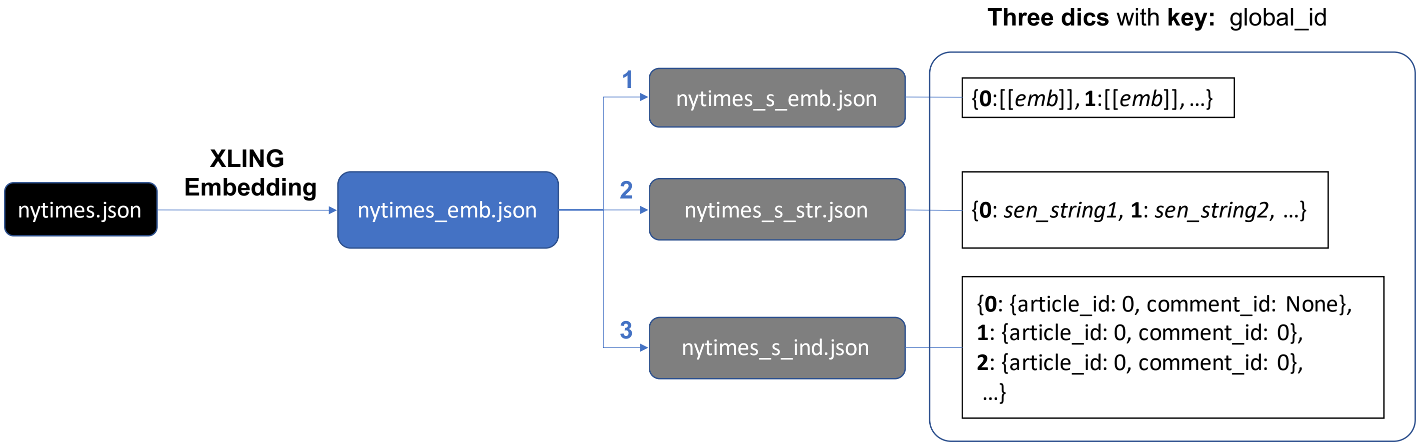


Figure: example from one article in quora

Stage 1.1: Reorganize sentences from articles

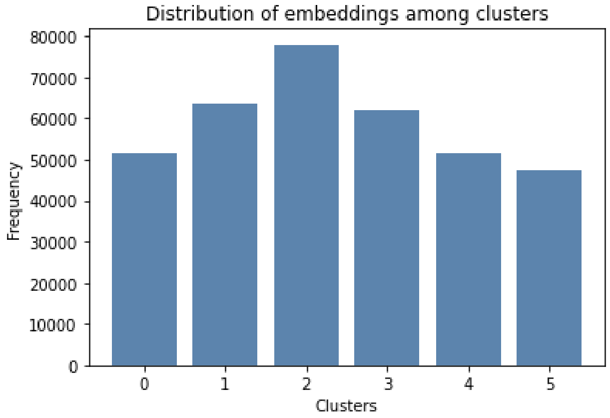
Extract embeddings from nested dict to get three separate files:

(i) embedding vectors → do clustering
(ii) strings → after clustering to generate word list, helping for define aspect labels
(iii) indexes → after sentence labeling, to do article (comments) level statistic



Stage 1.2: sklearn.cluster.MiniBatchKMeans

```
class KMeansClustering():
def __init__(self, k, X, is_mini_batch = True, plot_bar_chart = True):
  self.k = k
  self.X = np.array(X).reshape(len(X), 512)
  self.km = MiniBatchKMeans(n_clusters=k, init='k-means++', batch_size=3000, compute_labels=True).fit(self.X)
```



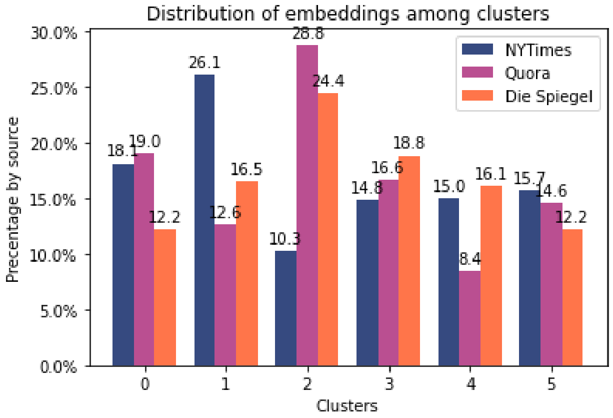


Figure: Example of distribution of embeddings of all tokenized sentences from the three sources among 6 clusters

Stage 1.2: sklearn.cluster.MiniBatchKMeans

Why MiniBatchKmeans instead of original sklearn.cluster.KMeans

XLING sentence level embeddings is generated in 512 dimensions for each tokenized sentence by NLTK.


```
_id: ObjectId("5ebe53b020438c599546a330")
embedding: Array
  0: Array
    0: -0.052148230373859406
    1: -0.054156072437763214
    2: -0.022018445655703545
    3: -0.06850385665893555
    4: -0.012877867557108402
    5: 0.053435664623975754
```

```
    502: 0.07619432359933853
    503: -0.012671503238379955
    504: -0.05270243063569069
    505: -0.012462617829442024
    506: 0.019090808928012848
    507: -0.005563048180192709
    508: 0.057824768126010895
    509: 0.043750520795583725
    510: 0.040863677859306335
    511: -0.015979250892996788
doc_id: 0
```


Figure: XLING embedding output for a sample sentence. Left: First 6 dimensions. Right: Last 10 dimensions

Source	Embedding JSON size	Original corpus size
New York Times	827 MB	55.9 MB
Quora	638 MB	15.9 MB
Die Speigel	2.3 GB	131 MB

Table: Embeddings generated are greatly larger then the original corpus size

Stage 1.2: sklearn.cluster.MiniBatchKMeans

Why MiniBatchKmeans instead of original sklearn.cluster.KMeans

Just loading all sentence embeddings in Google Colaboratory, 6.36 GB out of the given 12.72 GB RAM had already been used up.

MiniBatchKMeans is faster and helps to prevent the session from crushing, however, gives slightly different results.

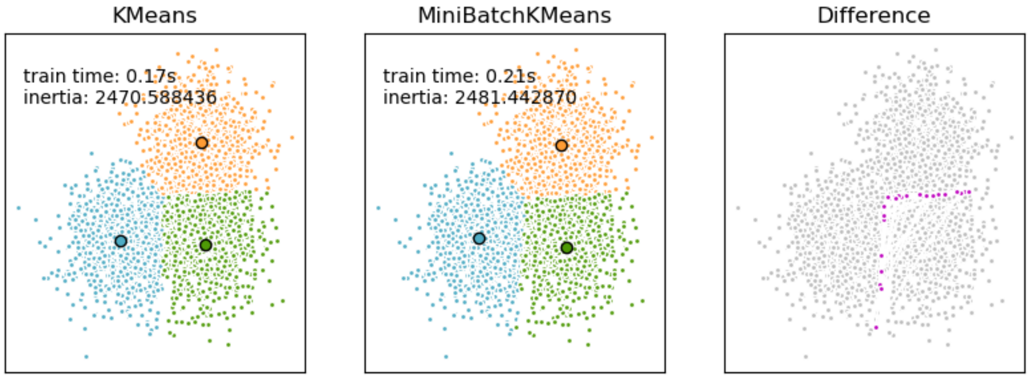


Figure: Extracted from scikit-learn; Data points classified differently are shown as purple points in 'Difference' block

`https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html`

Stage 1.2: Elbow Method for determining optimal k

```
K = range(2, 21)
for k in K:
  model = KMeansClustering(k, X)
  distortions.append(model.km.inertia_)
```

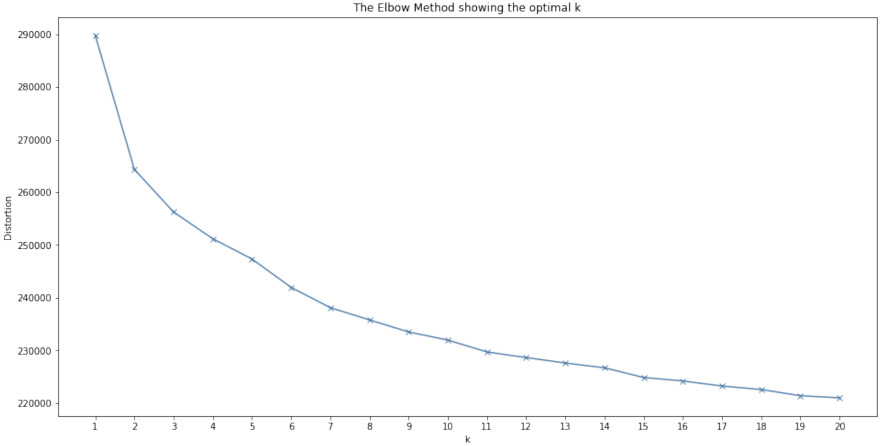
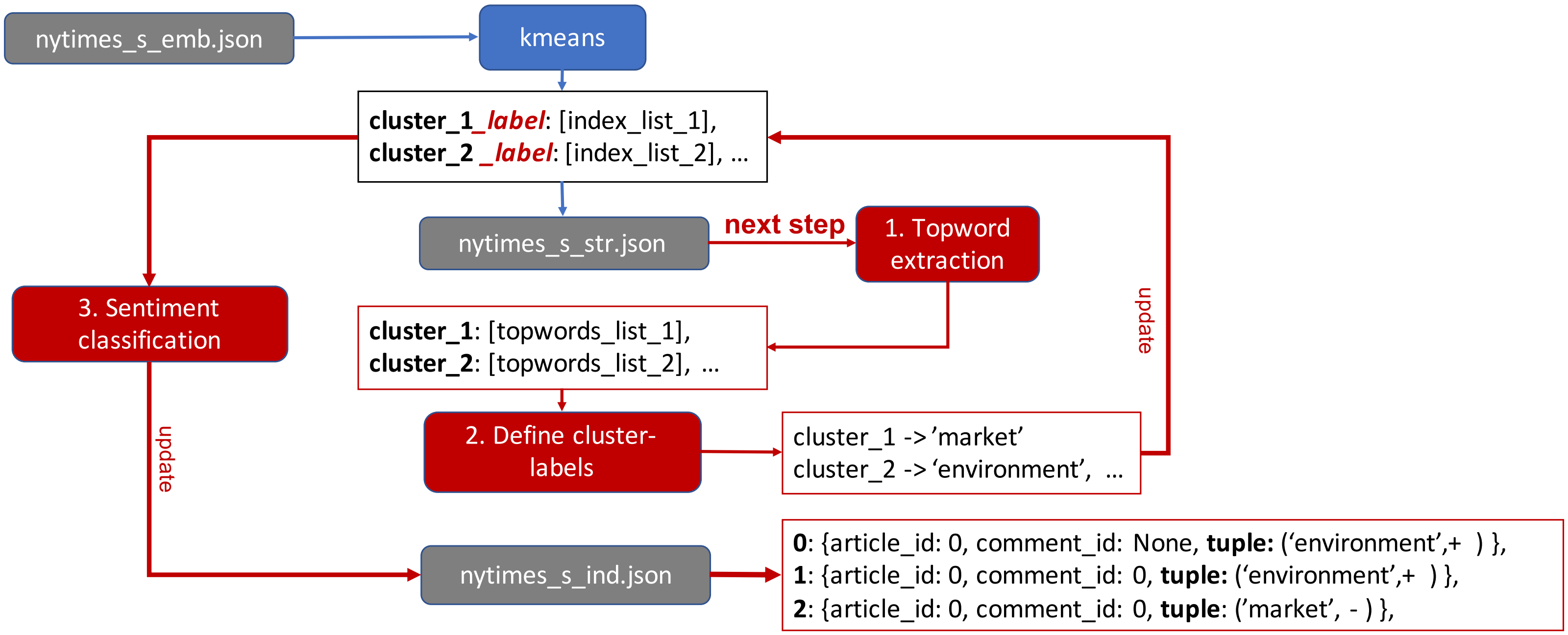


Figure: No distinguishable elbow of the curve for determination of optimal k

Future Plan


nytimes_s_emb.json
kmeans
cluster_1_label: [index_list_1],
cluster_2 _label: [index_list_2], …
nytimes_s_str.json
next step
1. Topword extraction
3. Sentiment classification
cluster_1: [topwords_list_1],
cluster_2: [topwords_list_2], …
update
2. Define cluster-labels
cluster_1 -> 'market'
cluster_2 -> 'environment', …
update
nytimes_s_ind.json
0: {article_id: 0, comment_id: None, tuple: ('environment',+) },
1: {article_id: 0, comment_id: 0, tuple: ('environment',+) },
2: {article_id: 0, comment_id: 0, tuple: ('market', -) },