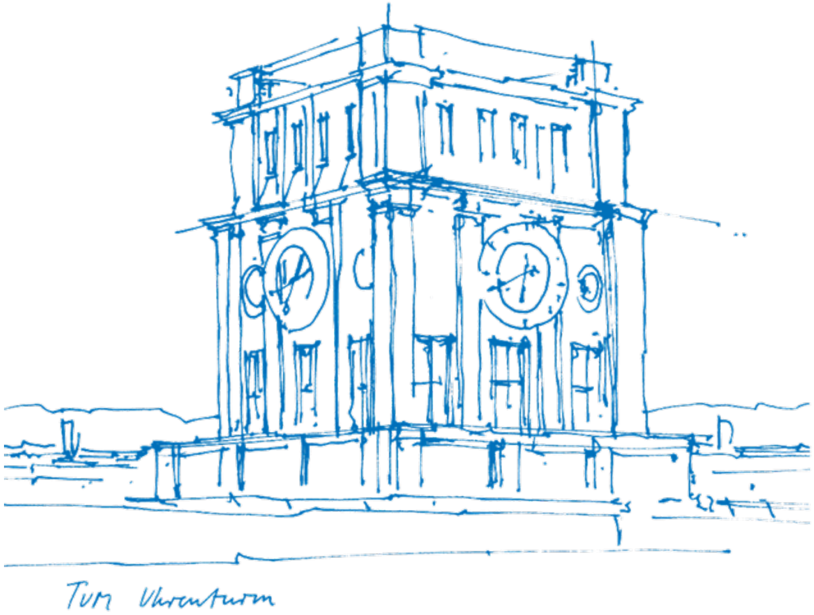
Technical University of Munich

Clustering-Based Sentiment Analysis for Media Agenda Setting

Opinion Lab Group 2.3

Wing Sheung Leung, Qiaoxi Liu

May 3, 2020



Outline

- Aims
- Dataset
- Expected outputs
- Todo
  - Stage 1
    - Generate word / token / sentence embeddings with our corpus
    - Build a k-mean clustering model for identifying sub-topics in organic dataset
    - Select a sentiment pre-trained model
  - Stage 2
    - Visualize distribution of samples among clusters and corresponding sentiment frequency
  - Stage 3
    - Investigate Media Agenda Setting
- Milestones

Aims

Measure the influence of two online newspapers onto the social media according to Agenda setting theory

Agenda setting theory:

Suggest the news item which is covered more frequently and prominently indicates that the audience will regard the issue as more important.

Dataset

Articles and respective comments on the domain of organic food with search terms *organic food* and *organic farming*

- Articles from two online newspapers, *New York Times* (English) and *Der Spiegel* (German)
- Direct response (bilingual): comments right under those articles
- Indirect response: posts in unrelated discussion forums, *Quora*

	Start	End	No. of articles
New York Times			
With comments	2006	2017	99
Without comments	1970	2017	228
Der Spiegel			
With comments	2007	2017	61
Without comments	2007	2017	91
Quora			
With comments	2009	2017	1304
Without comments	2010	2017	193

Table: Statistics for data crawled from New York Times, Der Spiegel and Quora with 'relevant' labelled as 1.0

Expected outputs (Processes)

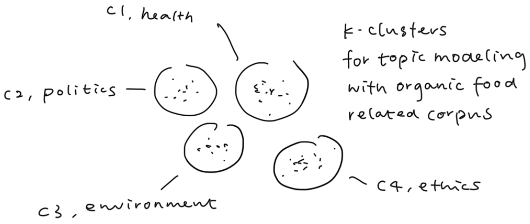


Figure: Stage 1: Cluster, tuple(aspect,sentiment)

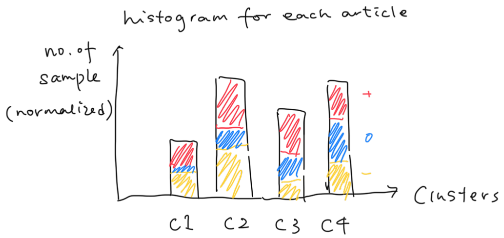


Figure: Stage 2: Distribution of clusters per articles

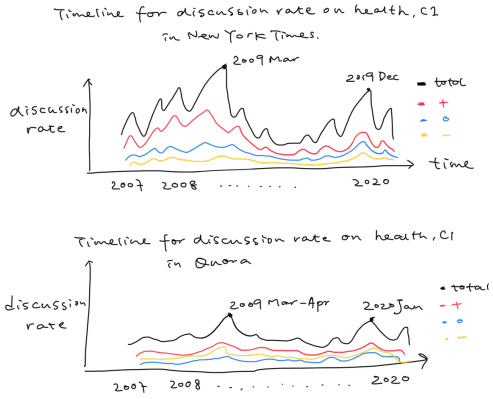


Figure: Stage 3: Timeline analysis, influence on social media

Todo Stage 1.1

Generate embeddings

text $\rightarrow$ *lines split* $\rightarrow$ list of sentences $\rightarrow$ *sentence embedding* $\rightarrow$ **vectors**

[1] Chidambaram et al. 2018.
[2] dalequark 2019.

Todo Stage 1.1

Generate embeddings

text $\rightarrow$ *lines split* $\rightarrow$ list of sentences $\rightarrow$ *sentence embedding* $\rightarrow$ **vectors**

Example: *sentence* ="You are what you eat."

[1] Chidambaram et al. 2018.
[2] dalequark 2019.

Todo Stage 1.1

Generate embeddings

text $\rightarrow$ *lines split* $\rightarrow$ list of sentences $\rightarrow$ *sentence embedding* $\rightarrow$ **vectors**

Example: *sentence* ="You are what you eat."

Possible embedding models (on TensorFlow Hub) to apply:

- Universal sentence encoder[1]
- Bert[2]

[1] Chidambaram et al. 2018.
[2] dalequark 2019.

Todo Stage 1.1

Generate embeddings

text $\rightarrow$ *lines split* $\rightarrow$ list of sentences $\rightarrow$ *sentence embedding* $\rightarrow$ **vectors**

Example: *sentence* ="You are what you eat."

Possible embedding models (on TensorFlow Hub) to apply:

- Universal sentence encoder[1]
- Bert[2]

```
embed = hub.Module("https://tfhub.dev/google/universal-sentence-encoder-xling/en-de/1")
with tf.Session() as session:
  session.run()
  print(session.run(embed(sentence)))
```

$\rightarrow [0.02, 0.01, ..., -0.12]$

[1] Chidambaram et al. 2018.
[2] dalequark 2019.

Todo Stage 1.2

Build a k-mean clustering model for identifying sub-topics

Now, let's start working with **vectors**!

$\rightarrow$ k-means cluster, optimize k with human observation (samples: sentences)

Todo Stage 1.2

Build a k-mean clustering model for identifying sub-topics

Now, let's start working with **vectors**!

$\rightarrow$ k-means cluster, optimize k with human observation (samples: sentences)

```
s_emb_matrix = [s0, s2, ..., s9] # s are embedding vectors of sentences
nclusters= 3
km = KMeans(nclusters)
km.fit(s_emb_matrix)
clusters = {} # key: label, values: index of sentence
for i, label in enumerate(km.labels_):
  clusters[label].append(i) # append the index to the corresponding label
```

Todo Stage 1.2

Build a k-mean clustering model for identifying sub-topics

Now, let's start working with **vectors**!

$\rightarrow$ k-means cluster, optimize k with human observation (samples: sentences)

```
s_emb_matrix = [s0, s2, ..., s9] # s are embedding vectors of sentences
nclusters= 3
km = KMeans(nclusters)
km.fit(s_emb_matrix)
clusters = {} # key: label, values: index of sentence
for i, label in enumerate(km.labels_):
  clusters[label].append(i) # append the index to the corresponding label
```

$\rightarrow$ get $clusters[0]$: s_0, s_4, s_5 with top-n words list $topWords[0]$: $w_0, w_1, ...w_n$

Todo Stage 1.2

Build a k-mean clustering model for identifying sub-topics

Now, let's start working with **vectors**!

$\rightarrow$ k-means cluster, optimize k with human observation (samples: sentences)

```
s_emb_matrix = [s0, s2, ..., s9] # s are embedding vectors of sentences
nclusters= 3
km = KMeans(nclusters)
km.fit(s_emb_matrix)
clusters = {} # key: label, values: index of sentence
for i, label in enumerate(km.labels_):
  clusters[label].append(i) # append the index to the corresponding label
```

$\rightarrow$ get $clusters[0]$: s_0, s_4, s_5 with top-n words list $topWords[0]$: $w_0, w_1, ...w_n$

$\rightarrow$ under each cluster, check similarities of words (samples: words)

Todo Stage 1.2

Build a k-mean clustering model for identifying sub-topics

Now, let's start working with **vectors**!

$\rightarrow$ k-means cluster, optimize k with human observation (samples: sentences)

```
s_emb_matrix = [s0, s2, ..., s9] # s are embedding vectors of sentences
nclusters= 3
km = KMeans(nclusters)
km.fit(s_emb_matrix)
clusters = {} # key: label, values: index of sentence
for i, label in enumerate(km.labels_):
  clusters[label].append(i) # append the index to the corresponding label
```

$\rightarrow$ get $clusters[0]$: s_0, s_4, s_5 with top-n words list $topWords[0]$: $w_0, w_1, ...w_n$

$\rightarrow$ under each cluster, check similarities of words (samples: words)

$\rightarrow$ interpret/extract $clusters[0]$ to an aspect (like environment).

$\rightarrow$ sentences s_0, s_4, s_5 are assigned to environment.

Todo Stage 1.3

Select a sentiment pre-trained model

Since we get $s_0 \rightarrow$ "environment", we use pre-trained VADER classifier[3] for each sentence, output a **2-tuple**.

[3] Hutto and Gilbert 2014.

Todo Stage 1.3

Select a sentiment pre-trained model

Since we get $s_0 \rightarrow$ "environment", we use pre-trained VADER classifier[3]
for each sentence, output a **2-tuple**.

Example

$s_0 =$"not only improves the fruit quality, but is a lot better on our environment as well."

[3] Hutto and Gilbert 2014.

Todo Stage 1.3

Select a sentiment pre-trained model

Since we get $s_0 \rightarrow$ "environment", we use pre-trained VADER classifier[3]
for each sentence, output a **2-tuple**.

Example

$s_0 =$"not only improves the fruit quality, but is a lot better on our environment as well."

$\rightarrow$ arr = [0.2,...,0.1] $\rightarrow$ *kmean clustering* $\rightarrow$ "Environment" (result from stage 1.2)

[3]Hutto and Gilbert 2014.

Todo Stage 1.3

Select a sentiment pre-trained model

Since we get $s_0 \rightarrow$ "environment", we use pre-trained VADER classifier[3]
for each sentence, output a **2-tuple**.

Example

$s_0 =$"not only improves the fruit quality, but is a lot better on our environment as well."

$\rightarrow$ arr = [0.2,...,0.1] $\rightarrow$ *kmean clustering* $\rightarrow$ "Environment" (result from stage 1.2)
$\rightarrow$ *VADER classifier*

```
analyzer = SentimentIntensityAnalyzer()
print(analyzer.polarity_scores(arr))
```

$\rightarrow$ {**'pos': 0.74**, 'neu': 0.26, 'neg': 0.0}

[3] Hutto and Gilbert 2014.

Todo Stage 1.3

Select a sentiment pre-trained model

Since we get $s_0 \rightarrow$ "environment", we use pre-trained VADER classifier[3]
for each sentence, output a **2-tuple**.

Example

$s_0 =$"not only improves the fruit quality, but is a lot better on our environment as well."

$\rightarrow$ arr = [0.2,...,0.1] $\rightarrow$ *kmean clustering* $\rightarrow$ "Environment" (result from stage 1.2)

$\rightarrow$ *VADER classifier*

```
analyzer = SentimentIntensityAnalyzer()
print(analyzer.polarity_scores(arr))
```

$\rightarrow$ {**'pos': 0.74**, 'neu': 0.26, 'neg': 0.0}

$\rightarrow$ **(Environment, pos)**

[3] Hutto and Gilbert 2014.

Todo Stage 2

Visualize distribution on single article (and its comments)

Now, the sentence s is converted into a tuple $t = (Aspect, +/-/o)$.

Todo Stage 2

Visualize distribution on single article (and its comments)

Now, the sentence s is converted into a tuple $t = (Aspect, +/-/o)$.

$\rightarrow$ Take all samples $[t_0, t_1, t_2, ...]$ from one article

$\rightarrow$ accumulate and normalize them

Todo Stage 2

Visualize distribution on single article (and its comments)

Now, the sentence s is converted into a tuple $t = (Aspect, +/-/o)$.

$\rightarrow$ Take all samples $[t_0, t_1, t_2, ...]$ from one article

$\rightarrow$ accumulate and normalize them

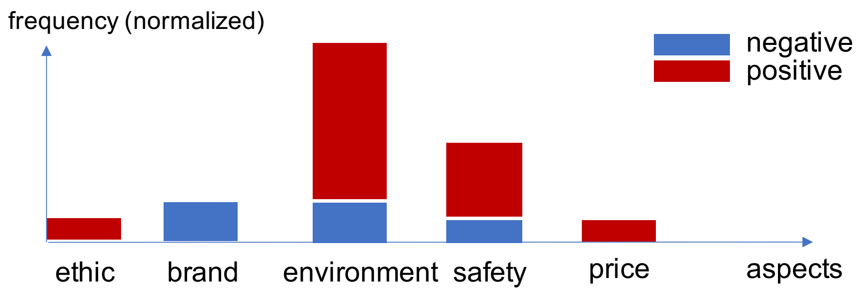


Figure: Aspects distribution on one text

Todo Stage 3

Investigate Media Agenda Setting

Does Y (news) has influence on X (comments/Quora)?

Todo Stage 3

Investigate Media Agenda Setting

Does Y (news) has influence on X (comments/Quora)?

$\rightarrow$ discover/measure the relationship between X,Y

- Pearson correlations (time not considered)
- Local Similarity Analysis (LSA) statistic identifies the existence of local and lagged relationships
- Granger causality (Does Y_t help to predict X_{t+1} ?)
- Lagged Correlation (response after a lapse of time, how strong correlation)

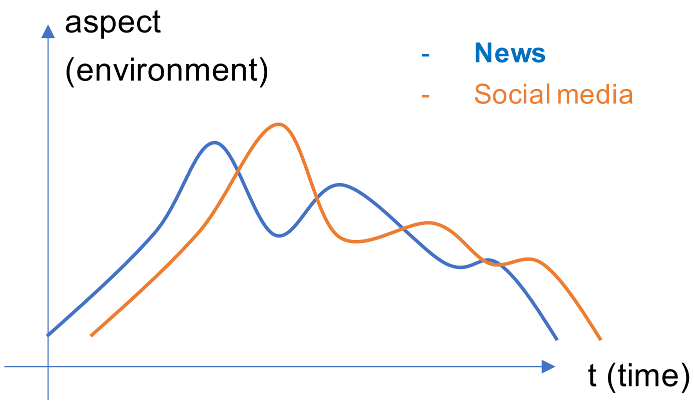
Todo Stage 3

Investigate Media Agenda Setting

Does Y (news) has influence on X (comments/Quora)?

$\rightarrow$ discover/measure the relationship between X,Y

- Pearson correlations (time not considered)
- Local Similarity Analysis (LSA) statistic identifies the existence of local and lagged relationships
- Granger causality (Does Y_t help to predict X_{t+1} ?)
- Lagged Correlation (response after a lapse of time, how strong correlation)



Milestones

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

Pre-processing and Stage 1 (now – 15.06)

Stage 2

Stage 3

References

- Chidambaram, M. et al. (2018). "Learning cross-lingual sentence representations via a multi-task dual-encoder model". In: *arXiv preprint arXiv:1810.12836*.
- dalequark (2019). *predicting movie reviews with bert on tf hub*. URL: https://github.com/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb (visited on 2019).
- Hutto, C. J. and E. Gilbert (2014). "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Eighth international AAAI conference on weblogs and social media*.