Social Topic Distributions

Hakan Akyürek

Mürüvvet Hasanbaşoğlu

June 29, 2020

Overview

1. **Sentence embeddings**
    - **Sentence preprocessing**
    - **Universal Sentence Encoder**
    - **Find optimal n_component**
2. **Experiments**
    - **Experiments where random users are selected from the same dataset**
    - **Experiments where random users are selected from other datasets**
    - **Analogy Experiments**
3. **Next Tasks -** for the next 2 weeks

1. Sentence Embeddings

1. **Retrieve the sentence vocabulary**

- Curated organic data = ~2.5M sentences
- Sentence preprocessing: remove only punctuation, one word and duplicate sentences and sentences that contain less than 15 characters. ➔ **~1.6M sentences**

1. Sentence Embeddings

1. **Retrieve the sentence vocabulary**

- Curated organic data = ~2.5M sentences
- Sentence preprocessing: remove only punctuation, one word and duplicate sentences and sentences that contain less than 15 characters. ➔ **~1.6M sentences**

2. **Compute sentence embeddings**

- Universal Sentence Encoder, 512dim

```
import tensorflow_hub as hub
universal_sentence_encoder = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")
train_embeddings = np.empty((0, 512), float)
for sentence, _ in sentence_vocabulary.items():
   embedding = universal_sentence_encoder([sentence])
   embedding = np.array(list(embedding)).astype(float)
   train_embeddings = np.append(train_embeddings, embedding, axis=0)
```

  ~6h/200k sentence ➔ too long!
- Training in sentence batches:
  ~1min/1000 sentence ➔ 1.6M sentence = 26h39min

1. Sentence Embeddings

1. **Retrieve the sentence vocabulary**

- Curated organic data = ~2.5M sentences
- Sentence preprocessing: remove only punctuation, one word and duplicate sentences and sentences that contain less than 15 characters. ➔ **~1.6M sentences**

2. **Compute sentence embeddings**

- Universal Sentence Encoder, 512dim

```
import tensorflow_hub as hub
universal_sentence_encoder = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")
train_embeddings = np.empty((0, 512), float)
for sentence, _ in sentence_vocabulary.items():
   embedding = universal_sentence_encoder([sentence])
   embedding = np.array(list(embedding)).astype(float)
   train_embeddings = np.append(train_embeddings, embedding, axis=0)
```

~6h/200k sentence ➔ too long!

- Training in sentence batches:
  ~1min/1000 sentence ➔ 1.6M sentence = 26h39min ➔ not sure if possible on CoLabs

1. Sentence Embeddings

3. **Train GMM with all sentences**

   Problem 1: More non-convergence issues

   Problem 2: Ram crashes

   Solutions:

   ➔ Maybe also training in batches using **warm_start()**

   or

   ➔ fit a random sample and predict all dataset

1. Sentence Embeddings

3. **Train GMM with all sentences**

   Problem 1: More non-convergence issues

   Problem 2: Ram crashes

   Solutions:

   ➔ Maybe also training in batches using **warm_start()**

   or

   ➔ fit a random sample and predict all dataset

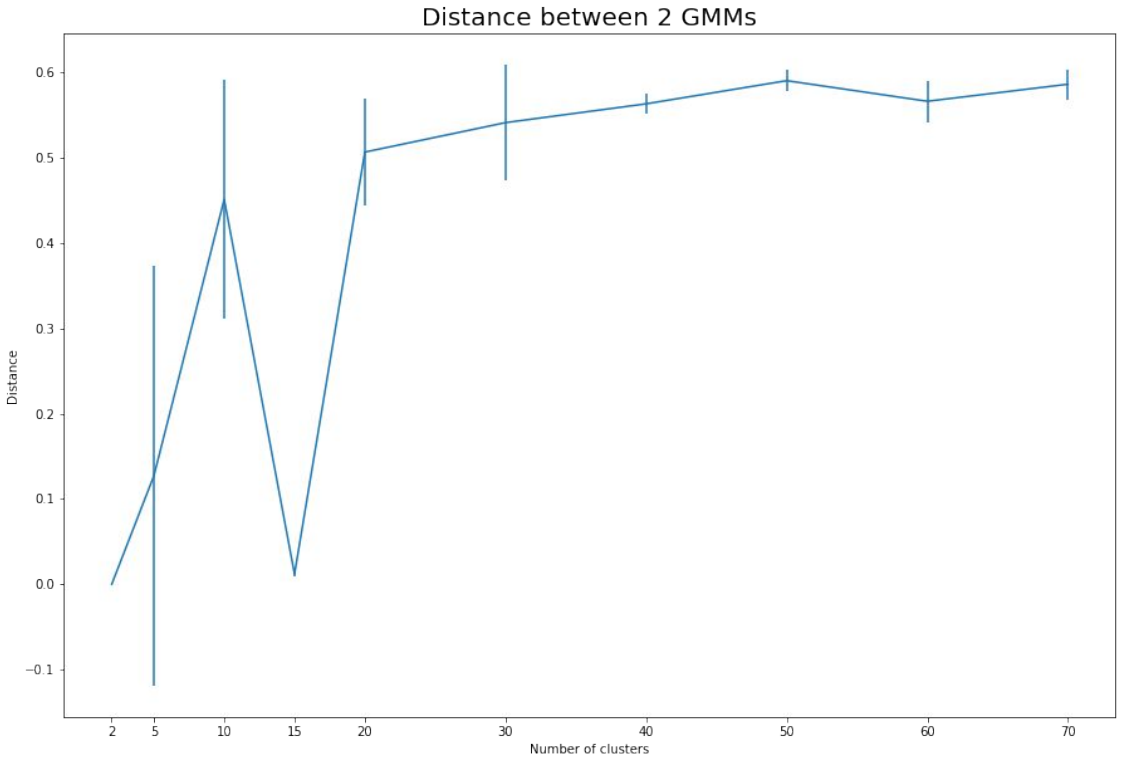
Stuck in 2nd step !

➔ Select optimal n_component using a randomly selected set of sentences

1. Sentence Embeddings

Optimal n_component for sentence embeddings:

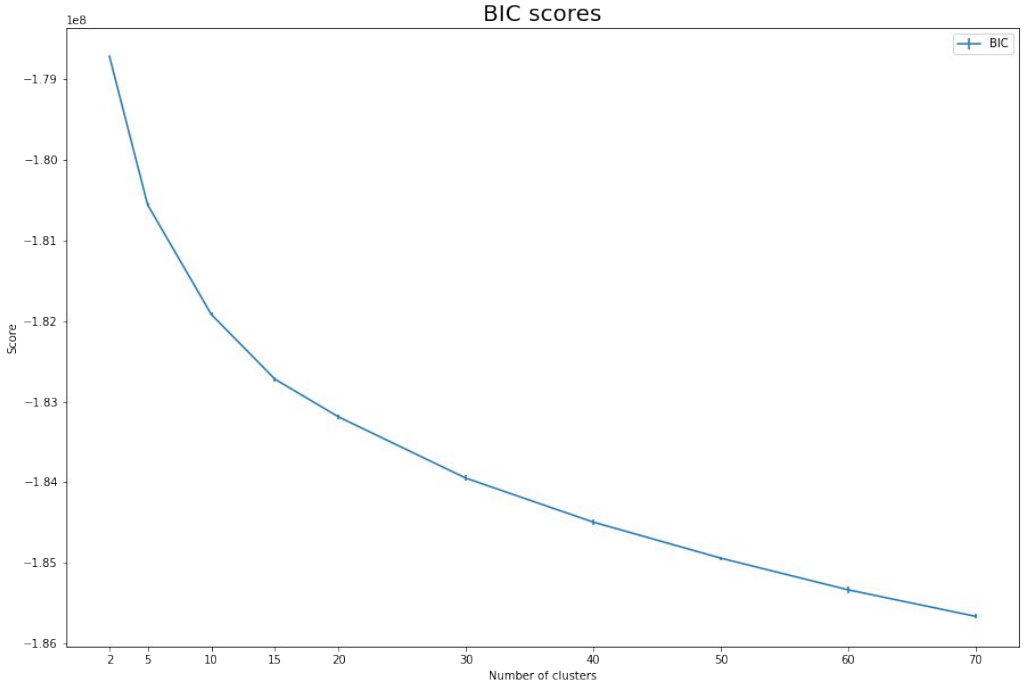
- Randomly selected 100k sentences
- range: 2, 5, 10, 20, …, 70

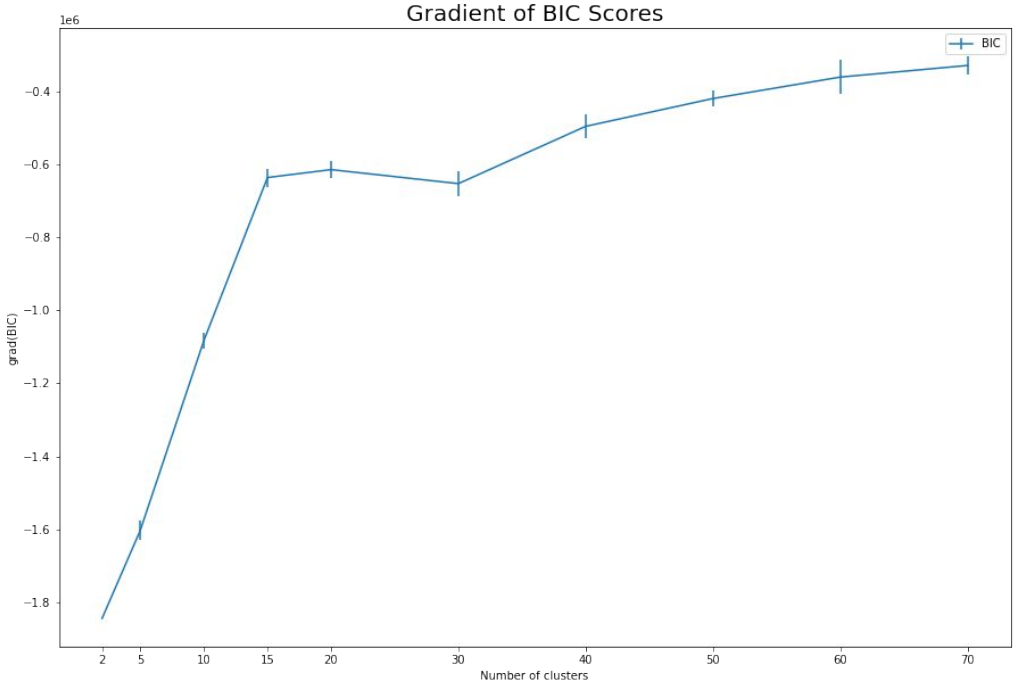


1. Sentence Embeddings

Optimal n_component for sentence embeddings:

- Randomly selected 100k sentences
- range: 2, 5, 10, 20, …, 70





2. Experiments

1. Experiments where random users are selected from the same dataset

- Got the random and answering users from the same dataset
  - Got 10 random comments from the article
  - Got 10 random comments from the rest of the comments in the dataset
- Done 10 times in order to avoid effects of randomness
- General purpose social media sites tend to not follow the general pattern

	Answering		Random	
	Mean	STD	Mean	STD
chicagotribune	0.384988575	0.1201952149	0.475516536	0.1156128185
latimes	0.4014499837	0.1102942562	0.4520089944	0.1169824563
reddit	0.3995765351	0.131873915	0.5002881615	0.1426158464
disqus	0.4026876656	0.1295383024	0.393727771	0.1040792169
foodbabe	0.3674852275	0.1108248364	0.5035238198	0.1331969733
foodrevolution	0.4438731436	0.1460208042	0.5325115038	0.09594465136
cafemom	0.4013961526	0.1603765152	0.467886504	0.1009873944
usatoday	0.4282095614	0.1648158043	0.4710413858	0.10318971
washingtonpost	0.3646384987	0.1291438602	0.4243904343	0.07344797032
facebook	0.5600918023	0.1159878002	0.5479025758	0.09357742856
nytimes	0.3591360881	0.1053282714	0.3561619568	0.09566741463
quora	0.4863544111	0.1106167707	0.500968684	0.1160446393

2. Experiments

2. Experiments where random users are selected from different datasets

- Got 10 random comments from the article
- Got 10 random comments from the rest of the comments in the all of the datasets
- Again, done 10 times in order to avoid effects of randomness
- General purpose social medias now follow the pattern.
- All of the datasets have similar means and std's with random users

	Answering		Random	
	Mean	STD	Mean	STD
chicagotribune	0.4118508929	0.1270097128	0.5214000578	0.1230789816
latimes	0.3932762817	0.09671429079	0.5268595905	0.1272388429
reddit	0.4087522402	0.1226142016	0.5177329269	0.1175239242
disqus	0.3885129398	0.1008069235	0.5070583155	0.1218285359
foodbabe	0.3779045535	0.09983987744	0.4976785046	0.1377173898
foodrevolution	0.4768770954	0.0981976633	0.5479795009	0.1316775007
cafemom	0.4010923075	0.1240303848	0.5183366842	0.1356301526
usatoday	0.4275641005	0.1112574329	0.5245200577	0.1302652644
washingtonpost	0.3878200908	0.1013928891	0.531840918	0.1394747124
facebook	0.4709319705	0.1636872796	0.550621792	0.1467391067
nytimes	0.4003628918	0.1084330895	0.5564675406	0.12994026
quora	0.3238557891	0.08514398147	0.5061730248	0.1530962292

2. Experiments

3. Analogy Experiments

- JS(A1|C1) < JS(A1|C2)
  JS(A2|C1) > JS(A2|C2) → JS(A1|A2) ~ JS(C1|C2)

- A1 and C1 should be in the same distribution
- A2 and C2 should be in the same distribution

JS(A1|A2)	JS(C1|C2)
0.4601	0.3587
0.4817	0.5645
0.3649	0.3334
0.4911	0.8305
0.5566	0.3749

3. Next Tasks

- Experiments per dataset cluster e.g:
  - biased vs unbiased
  - forums vs news sites
- Experiments with sentence embeddings
- Maybe do analogy Task on multiple datasets
- Maybe try out fasttext

Questions

1. How to do the topic labeling for sentence embeddings model?

References

- Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.
- Sridhar, Vivek Kumar Rangarajan. "Unsupervised topic modeling for short texts using distributed representations of words." *Proceedings of the 1st workshop on vector space modeling for natural language processing*. 2015.
- Fine tune GloVe embeddings using Mittens
- roamanalytics/mittens: A fast implementation of GloVe, with optional retrofitting
- https://medium.com/analytics-vidhya/basics-of-using-pre-trained-glove-vectors-in-python-d38905f356db
- https://scikit-learn.org/stable/modules/mixture.html
- https://towardsdatascience.com/gaussian-mixture-model-clusterization-how-to-select-the-number-of-components-clusters-553bef45f6e4
- https://stackoverflow.com/questions/26079881/kl-divergence-of-two-gmms
- https://medium.com/@sourcedexter/how-to-find-the-similarity-between-two-probability-distributions-using-python-a7546e90a08d
- Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Sung, Y. H. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.