Confronting Sparseness and High Dimensionality in Short Text Clustering via Feature Vector Projections

Leonidas Akritidis[1], Miltiadis Alamaniotis[2], Athanasios Fevgas[3], Panayiotis Bozanis[1]

[1]School of Science and Technology, International Hellenic University

[2]Department of Electrical and Computer Engineering, University of Texas at San Antonio

[3]Department of Electrical and Computer Engineering, University of Thessaly

The 32nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2020)

November 9-11, 2020, Virtual Conference

Short Text Clustering

- A problem that focuses on the unsupervised grouping of similar short text documents, or entitled entities.
- Specialization of the general text clustering problem which in turn is a specialization of the generic data clustering problem.
- Quite popular. Important applications include cluster creation from:
  - microblogs,
  - entitled entities (e.g. news headlines, product titles, etc.),
  - FAQs,
  - search result snippets,
  - etc.

Short-Texts: 2 difficult problems

- Short texts suffer from 2 challenging issues:
- **Data sparseness:** it concerns the absence of important features (terms) from a portion of the input records.
  - It blurs the similarities among the involved entities.
- **High dimensionality:** originates from the usage of a huge number of features (terms, n-grams, skip grams etc.) in short texts.
  - It is the source of the “Curse of Dimensionality”.

Clustering algorithms: Primary goals

- A clustering algorithm must group its input records by fulfilling (at least) two goals:
- **Homogeneity:** inclusion of *only* similar elements within a cluster, and
- **Completeness:** *all* similar elements are grouped into the same cluster.

VEPHC

- A two stage short text clustering algorithm.
- **Stage 1 (VEP part)** projects the original feature vectors onto a lower dimensional space by generating all the feature combinations of the initial text vectors.
- A feature combination corresponds to a projection vector and is treated as a candidate cluster label.
- Each projection vector is assigned a score which favors completeness and the homogeneity.
- All documents whose feature vectors are projected onto the same space are grouped into the same cluster.
- **Stage 2 (HC part)** consists of a post-processing algorithm that enhances the homogeneity and the completeness of the clusters that were generated by the previous stage.

VEPHC Part 1: Feature combinations as vector projections

- An integer hyper-parameter K is initially set.
- For each input vector $\mathbf{x}_i$ we construct a set $\mathbf{X}'_i$ containing all the possible projections of $\mathbf{x}_i$ with a dimensionality of $1, 2, \ldots K$ elements.
- In other words, $\mathbf{X}'_i$ contains all the $\{1, 2, \ldots K\}$ combinations of the components of $\mathbf{x}_i$.
- If $\mathbf{x}_i = \{x_1, x_2, x_3\}$, then $\mathbf{X}'_i = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_1 x_2\}, \{x_1 x_3\}, \{x_2 x_3\}\}$.

Scoring the vector projections

- For each input document, each projection vector $\mathbf{x}'_{ij}$ in $\mathbf{X}'_i$ is assigned a score:

$$S_{\mathbf{x}'_{ij}} = \frac{1}{l_{\mathbf{x}'_{ij}}} \log f_{\mathbf{x}'_{ij}} \sum_{\forall x_{ij} \in \mathbf{x}'_{ij}} x_{ij}^K,$$

- Where $l_{\mathbf{x}'_{ij}}$ is the length of $\mathbf{x}'_{ij}$
- $f_{\mathbf{x}'_{ij}}$ is the global frequency of $\mathbf{x}'_{ij}$.
- x_{ij}: a component of $\mathbf{x}'_{ij}$.

Scoring the vector projections (2)

- If the original feature vectors have been created with tf-idf:

$$S_{\mathbf{x}'_{ij}} = \frac{1}{l_{\mathbf{x}'_{ij}}} \log f_{\mathbf{x}'_{ij}} \sum_{\forall x_{ij} \in \mathbf{x}'_{ij}} \left(tf_{ij} \cdot \log \frac{n}{f_j} \right)^K$$

- Where n is the number of the input records.
- f_j is the freq of the j–th component of $\mathbf{x}'_{ij}$.
- The highest-scoring projection of $\mathbf{x}_i$ is declared as the dominant cluster label $\mathbf{x}_i^*$.
- The input is clustered in $\mathbf{x}_i^*$.

Example of clustering

- $\mathbf{x}_1 = \{x_1, x_2, x_3, x_4, x_5\}$
- $\mathbf{x}_2 = \{0, x_2, x_3, x_4, 0\}$
- If $\mathbf{x}_1^* = \mathbf{x}_2^* = \{x_2, x_4\}$
- Then $\mathbf{x}_1$ and $\mathbf{x}_2$ will be placed inside the same cluster.

Consequences

- The higher the dimensionality of the dominant vector, the more similar documents the corresponding cluster contains.
- The documents included in that cluster will have more words in common.
- High/Low dimensional dominant projection vectors lead to:
  - More/Lesser homogeneous clusters.
  - Lesser/More complete clusters.

VEPHC Part 2: Cluster refinement stage

- At this point, the dominant projections of the input short text documents have been computed.
- The input documents have been clustered in the corresponding cluster labels.
- The second part introduces a refinement algorithm which was designed to further improve homogeneity and completeness.

VEPHC Part 2 – Stage 1

- Split the existing clusters & create new clusters.
- In short:
- Compute the clustroid point of each cluster.
  - i.e. the point that has max sim with the rest of the elements of the cluster.

Algorithm 1: HC Part, Phase 1: Element Transfer and Creation of New Clusters

```
1  initialize an empty cluster C;
2  for each cluster c ∈ C do
3  |  u_c ← clustroid of c;
4  end
5  for each cluster c ∈ C do
6  |  for each element x ∈ c do
7  |  |  T_{x,u_c} ← similarity between x and u_c;
8  |  |  if T_{x,u_c} < T_h then
9  |  |  |  remove x from c;
10 |  |  |  T_max ← 0, c* ← NULL;
11 |  |  |  for each cluster c' ∈ C do
12 |  |  |  |  T_{x,u'_c} ← similarity between x and u'_c;
13 |  |  |  |  if T_{x,u'_c} > T_max then
14 |  |  |  |  |  T_max ← T_{x,u'_c};
15 |  |  |  |  |  c* ← c';
16 |  |  |  |  end
17 |  |  |  end
18 |  |  |  if T_max ≥ T_c then
19 |  |  |  |  c* ← c* ∪ {x};
20 |  |  |  else
21 |  |  |  |  C ← C ∪ {x};
22 |  |  |  end
23 |  |  end
24 |  end
25 end
26 C_new ← ∅;
27 for each element x ∈ C do
28 |  remove x from C;
29 |  T_max, c* ← perform steps 11–17 on C_new;
30 |  if T_max ≥ T_c then
31 |  |  c* ← c* ∪ {x};
32 |  else
33 |  |  create and initialize new cluster c_new ← ∅;
34 |  |  c_new ← c_new ∪ {x}, u_{c_new} ← x;
35 |  |  C ← C ∪ {c_new}, C_new ← C_new ∪ {c_new};
36 |  end
37 end
```

VEPHC Part 2 – Stage 1

- For each cluster:
- Evict the cluster elements that are too far away from the clustroid.
- Evicted elements may be:
  - inserted into another, more similar cluster, OR
  - placed into a new, empty cluster.

Algorithm 1: HC Part, Phase 1: Element Transfer and Creation of New Clusters

```
1  initialize an empty cluster C;
2  for each cluster c ∈ C do
3  |  u_c ← clustroid of c;
4  end
5  for each cluster c ∈ C do
6  |  for each element x ∈ c do
7  |  |  T_{x,u_c} ← similarity between x and u_c;
8  |  |  if T_{x,u_c} < T_h then
9  |  |  |  remove x from c;
10 |  |  |  T_max ← 0, c* ← NULL;
11 |  |  |  for each cluster c' ∈ C do
12 |  |  |  |  T_{x,u'_c} ← similarity between x and u'_c;
13 |  |  |  |  if T_{x,u'_c} > T_max then
14 |  |  |  |  |  T_max ← T_{x,u'_c};
15 |  |  |  |  |  c* ← c';
16 |  |  |  |  end
17 |  |  |  end
18 |  |  |  if T_max ≥ T_c then
19 |  |  |  |  c* ← c* ∪ {x};
20 |  |  |  else
21 |  |  |  |  C ← C ∪ {x};
22 |  |  |  end
23 |  |  end
24 |  end
25 end
26 C_new ← ∅;
27 for each element x ∈ C do
28 |  remove x from C;
29 |  T_max, c* ← perform steps 11–17 on C_new;
30 |  if T_max ≥ T_c then
31 |  |  c* ← c* ∪ {x};
32 |  else
33 |  |  create and initialize new cluster c_new ← ∅;
34 |  |  c_new ← c_new ∪ {x}, u_{c_new} ← x;
35 |  |  C ← C ∪ {c_new}, C_new ← C_new ∪ {c_new};
36 |  end
37 end
```

VEPHC Part 2 – Stage 2

- Cluster merging stage.
- For each cluster:
- Search for other, highly similar clusters.
- If the similarity with the most similar cluster exceeds a threshold, then we merge the two clusters.

Algorithm 2: HC Part, Phase 2: Cluster Merging

```
1  for each cluster c ∈ C do
2  |   u_c ← clustroid of c;
3  end
4  sim_array[*] ← 0, msc_array[*] ← −1;
5  for each cluster c ∈ C do
6  |   msc_array[c] ← most similar cluster (MSC);
7  |   sim_array[c] ← similarity with MSC;
8  end
9  while merge do
10 |   merge ←FALSE;
11 |   c′ ←NULL, c″ ←NULL;
12 |   for each cluster c ∈ C do
13 |   |   T_max ← 0;
14 |   |   if sim_array[c] > T_max then
15 |   |   |   T_max ← sim_array[c];
16 |   |   |   c′ ← c, c″ ← msc_array[c];
17 |   |   end
18 |   end
19 |   if T_max > T_c then
20 |   |   merge ←TRUE;
21 |   |   c′ ← c′ ∪ c″;
22 |   |   u′_c ← clustroid of c′;
23 |   |   msc_array[c′] ← MSC;
24 |   |   sim_array[c′] ← similarity with MSC;
25 |   |   C ← C − {c″};
26 |   |   update msc_array, sim_array;
27 |   end
28 end
```

Experiments

- We attested VEPHC by using two datasets:
  - A dataset of product titles from Pricerunner.
  - A dataset of news titles (headlines).
- Versus 4 generic data clustering + 3 short text clustering methods.
- In terms of both clustering quality (P, R, F1, NMI) and execution times.

EXPERIMENTAL DATASETS

	n	$\|C\|$	$l_{\max}$	l_{ave}
PriceRunner (PRUN)	35311	13233	41	8.23
UCI News Aggregator (UNA)	50138	823	15	7.05

Dependence from the hyper-parameters

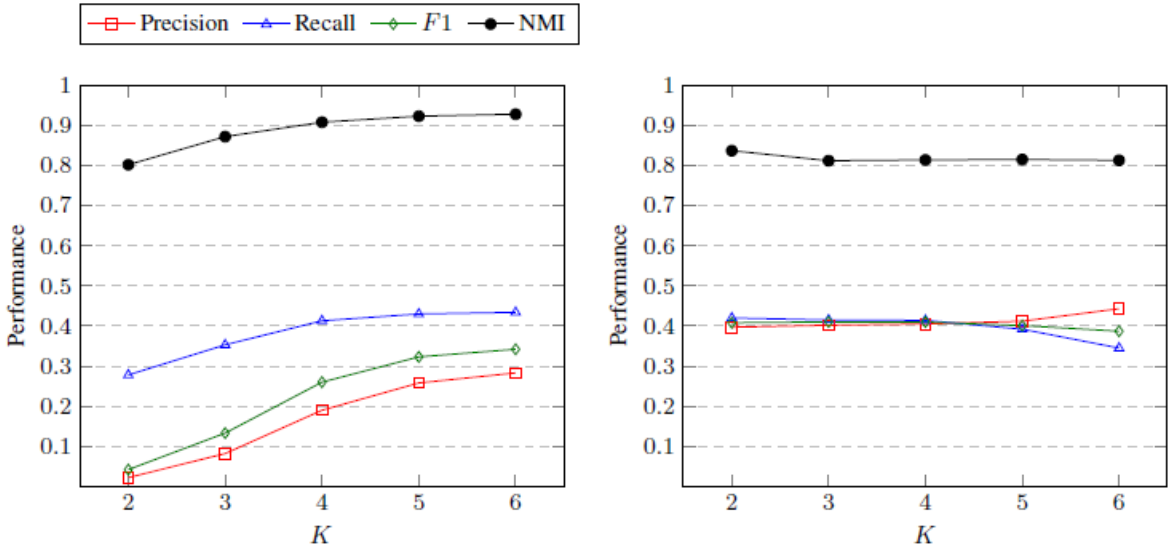

Fig. 1. Performance fluctuation of VEPHC (without the verification stage) against varying values of the hyper parameter K, on the PriceRunner (left), and UCI News Aggregator (right) datasets.

Clustering performance

PERFORMANCE EVALUATION OF VARIOUS CLUSTERING ALGORITHMS ON THE PRICERUNNER AND UCI NEWS AGGREGATOR DATASETS

Method	Setup	PriceRunner			UCI News Aggregator		
		$F1$	NMI	Time	$F1$	NMI	Time
VEPHC	$K_1 = 6, K_2 = 2$	0.385	0.946	305.1	0.471	0.851	10.2
k-Means	$k = \|C\|, I = 10$	0.048	0.399	494.2	0.028	0.697	104.3
Agglomerative	–	0.314	0.935	166.7	0.454	0.835	2492.8
Leader Clustering	–	0.306	0.934	17.2	0.284	0.778	17.1
DBSCAN	$minPoints = 2$	0.055	0.425	58.4	0.256	0.790	130.5
GSDMM-1	$\alpha = \beta = 0.1, k = 1.5 \cdot \|C\|$	0.002	0.404	3246.2	0.171	0.753	390.4
GSDMM-2	$\alpha = 0.001, \beta = 0.01, k = 1.5 \cdot \|C\|$	0.008	0.631	3317.3	0.424	0.820	453.9
Spherical k-Means	$k = \|C\|, I = 10$	0.226	0.903	426.1	0.335	0.757	119.6
vk-Means	$k = \|C\|, I = 10$	0.220	0.900	484.4	0.154	0.614	104.7
Cosine similarity	$tf - idf$ weights	0.248	–	31.3	0.388	–	75.2
Jaccard Index	$tf - idf$ weights	0.237	–	31.5	0.388	–	74.9

Conclusions

- We introduced VEPHC, a two-stage clustering algorithm for short text documents.
- Designed to limit the native sparseness and high dimensionality of short texts.
- Key elements:
  - Projection of the feature vectors onto a lower dimensional space by the construction and scoring of feature combinations.
  - A feature combination can be viewed as a projection vector.
  - Identify the dominant (i.e., the highest scoring) projection vector.
  - Two or inputs having the same projection vector are grouped into the same cluster.

Conclusions

- Key elements (continued):
  - Post processing stage: Split the initial clusters by removing the most distant items.
  - Place the evicted items into other clusters, or create new ones.
  - Merge the most similar clusters.
- Experiments conducted with two test datasets.
- The results showed significant performance improvements over the current state-of-the-art methods.
- In terms of both effectiveness and efficiency.

Thank you

Any questions?