UDC 519.2
MSC 60E05

Вестник СПбГУ. Прикладная математика. Информатика... 2018. Т. 14. Вып. 4

Setting lower bounds on Jensen—Shannon divergence and its application to nearest neighbor document search

V. Yu. Dobrynin[1], *N. Rooney*[2], *J. A. Serdyuk*[3]

[1] St. Petersburg State University, 7–9, Universitetskaya nab., St. Petersburg, 199034, Russian Federation
[2] Sophia Ltd, Northern Ireland Science Park, the Innovation Centre, Queen's Road, Queen's Island, Belfast, BT3 9DT, Northern Ireland
[3] Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

For citation: Dobrynin V. Yu., Rooney N., Serdyuk J. A. Setting lower bounds on Jensen—Shannon divergence and its application to nearest neighbor document search. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 2018, vol. 14, iss. 4, pp. 334–345. https://doi.org/10.21638/11702/spbu10.2018.406

The Jensen—Shannon divergence provides a mechanism to determine nearest neighbours in a document collection to a specific query document. This is an effective mechanism however for exhaustive search this can be a time-consuming process. In this paper, we show by setting lower bounds on the Jensen—Shannon divergence search we can reduce by up to a factor of 60% the level of calculation for exhaustive search and 98% for approximate search, based on the nearest neighbour search in a real-world document collection. In these experiments a document corpus that contains 1 854 654 articles published in New York Times from 1987-01-01 till 2007-06-19 (The New York Times Annotated Corpus) was used. As queries, 100 documents from same document corpus were selected randomly. We assess the effect on performance based on the reduction in the number of log function calculations. Approximate nearest neighbour search is based on clustering of documents using Contextual Document Clustering algorithm. We perform an approximated nearest neighbour search by finding the best matching set of cluster attractors to a query and limiting the search for documents to the attractors' corresponding clusters.

Keywords: Jensen—Shannon divergence, nearest neighbors search, dimensionality reduction.

1. Introduction. Many tasks in information retrieval require searching for the most similar (nearest neighbor (NN)) document in a document corpus given a query document. As a document corpus may contain millions of terms and each document may contain only a relatively small set of terms, this problem poses particular problems in being able to identify nearest neighbor in a manner that is faster than brute-force search. Document can be represented in a number of different forms but a standard approach is based on the bag of words model, where a document can be represented as a multinomial probability distribution over words [1]. Similarity between documents can be evaluated by standard measures for comparing probability distributions such as Kullback—Leibler or Jensen—Shannon (JS) divergence, which we utilize in this case.

A main area of research in retrieving the nearest neighbor has focused on the use hierarchical decomposition of the search space (KD-tree, metric ball tree, bregman ball tree). These approaches utilize geometric properties that enable fast search by pruning out area of the search space via a branch and bound exploration. KD-trees [2] is one example of the latter approach, where the tree defines a hierarchical space partition, where each node defines an axis-aligned rectangle [3]. Metric ball trees extend the basic mechanism behind

KD-trees to metric spaces by using metric balls in place of rectangles. Bregman ball trees (BB-trees) [4, 5] are similar to metric ball trees however such trees can be constructed for any Bregman divergence, which is a generalized dissimilarity measure and need not support the triangular inequality. Examples of Bregman divergence measures include Kullback—Leibler. Results of experiments show that significant speed up (3 times or more) for exact neighbor search can be achieved with BB-trees, if dimension of search space is not very large (256 or less). But for higher dimensionality, space partitioning technique are still a problem in exact NN search. Such algorithms are exponential in the number of dimensions d and in case of large d the best solution is to use brute-force search [6].

As a consequence, research in this area has concentrated instead on approximate nearest neighbor. In this formulation, the algorithm is allowed to return a point whose distance from the query is at most c times the distance from the query to its nearest points, where $c > 1$ is called the approximation factor [7]. One of most popular approaches in this area is based on Locality Sensitive Hashing (LSH) [8, 9]. The key idea behind this technique is to hash the points using several hash functions to ensure that for each function the probability of collision is much higher for objects that are close to each other than for those that are far apart. Then, one can determine near neighbors by hashing the query point and retrieving elements stored in buckets containing that point. In the case of d-dimensional Euclidean space LSH has polynomial preprocessing time and sub-linear query time [10].

The curse of dimensionality problem can be solved with randomized algorithms. In works [11, 12] Dynamic Continuous Indexing (DCI) and Prioritized DCI exact NN algorithms were proposed. Instead of space partitioning, data points are projected on a number of random directions and corresponding indexes are created to store ranks of data points in each direction.

Term or feature clustering is at least as popular theme of research as document clustering in information retrieval community [13–17]. In this article we also use term clustering for contraction of probability distribution of words as a document representation. However our goal is less ambitious at this point — we use very simple word statistics based clustering that is specific to our task of fast NN search.

Rather than focus on alternatives to brute force search, we consider, if there are means to significantly reduce the number of calculations in finding the nearest neighbor, when documents are represented as probability distributions and the dissimilarity measure is based on JS divergence. We consider both exact nearest neighbor using brute-force search and approximate nearest neighbor search based on clustering of documents. In [18] was presented a mechanism of clustering documents based on the concept of identifying of contextual attractors to which documents are assigned to form a simple partitioning of the document space, where a cluster of documents is created for each attractor. The contextual attractors are represented as probability distributions over a maximum 1000 terms over the term space.

As candidates for contextual attractors, considered conditional probability distributions

$$p(y|z) = \frac{\sum\limits_{x \in D_z} \mathtt{tf}(x, y)}{\sum\limits_{x \in D_z, y' \in T} \mathtt{tf}(x, y')}$$

for all context terms z ($z \in T$, where T is the set of all unique terms from the document corpus) except too rare or too common terms (terms with too small or too high document

frequency). Here D_z stands for set of documents that contain term z and $\mathtt{tf}(x, y)$ denotes term frequency (number of occurrences of term y in document x).

To select final set of contextual attractors we select a set of terms Z with relatively low entropies of distributions $p(y|z), z \in Z$. Each attractor is reduced to up to 1000 most probable terms and then each document is assigned to nearest attractor (in terms of JS divergence).

It is possible to perform an approximated nearest neighbour search by finding a best matching set of attractors to a query and limiting the search for documents to the attractors' corresponding clusters. It will be shown that by determining lower bounds on the JS divergence the level of calculation can be reduced in both approaches.

2. Lower bounds on JS. JS divergence was proposed in [19]. Many interesting inequalities related to JS divergence and many other divergences can be found in [20–23]. Let A be finite alphabet, P and Q be probability distributions over A,

$$D(P\|Q) = \sum_{x\in A} P(x) \log \frac{P(x)}{Q(x)}$$

be relative entropy. Then the JS divergence is given by

$$\mathrm{JS}(P, Q) = \frac{1}{2} D(P\|M) + \frac{1}{2} D(Q\|M), \tag{1}$$

where $M = \frac{P+Q}{2}$.

Let $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$ be probability distributions. Some probabilities in both p and q can be equal to zero but $\sum\limits_{i=1}^{n} p_i = \sum\limits_{i=1}^{n} q_i = 1$.

The formula (1) can be rewritten as

$$\mathrm{JS}(p, q) = 1 + 0.5 \sum_{i=1}^{n} \left(p_i \log_2 \frac{p_i}{p_i + q_i} + q_i \log_2 \frac{q_i}{p_i + q_i} \right).$$

Let $\bigcup_{j=1}^{m} A_j$ be a partition of $\{1, 2, ..., n\}$, then

$$\mathrm{JS}(p, q) = 1 + 0.5 \sum_{j=1}^{m} \sum_{i\in A_j} \left(p_i \log_2 \frac{p_i}{p_i + q_i} + q_i \log_2 \frac{q_i}{p_i + q_i} \right).$$

Let constants P_j, Q_j for $j \in \{1, ..., m\}$ be determined by formulas

$$P_j - \sum_{i\in A_j} p_i = 0,$$

$$Q_j - \sum_{i\in A_j} q_i = 0.$$

Claim 1:

$$\mathrm{JS}(p, q) \geqslant 1 + 0.5 \sum_{j=1}^{m} \left(P_j \log_2 \frac{P_j}{P_j + Q_j} + Q_j \log_2 \frac{Q_j}{P_j + Q_j} \right). \tag{2}$$

Proof. Consider the optimization task:

$$F_j(p', q') = \sum_{i\in A_j} \left(p'_i \log_2 \frac{p'_i}{p'_i + q'_i} + q'_i \log_2 \frac{q'_i}{p'_i + q'_i} \right) \to \min \tag{3}$$

w.r.t.

$$P_j - \sum_{i \in A_j} p'_i = 0, \tag{4}$$

$$Q_j - \sum_{i \in A_j} q'_i = 0. \tag{5}$$

To take into account these conditions on p'_i, q'_i used Lagrange multipliers:

$$H_j(p', q') = F_j(p', q') + \pi_1 \left(\sum_{i \in A_j} p'_i - P_j \right) + \pi_2 \left(\sum_{i \in A_j} q'_i - Q_j \right),$$

$$\frac{\partial H_j}{\partial p'_i} = -\log_2 \frac{p'_i + q'_i}{p'_i} + \pi_1 = 0, \tag{6}$$

$$\frac{\partial H_j}{\partial q'_i} = -\log_2 \frac{p'_i + q'_i}{q'_i} + \pi_2 = 0. \tag{7}$$

Please note that we can use Lagrange multipliers method here because the functions in (3)–(5) are C^1 functions and the gradients of the functions in (4) and (5) are linear independent, if

$$p'_i > 0 \tag{8}$$

and

$$q'_i > 0 \tag{9}$$

for all $i \in A_j$ in (6) and (7). We can assume that (8) and (9) hold because any summand in (3), where $p'_i = 0$ or $q'_i = 0$ is equal to zero so we can ignore such $i \in A_j$.

So for each $i \in A_j$ we have $\pi_1 = \log_2 \frac{p'_i + q'_i}{p'_i}$, $\pi_2 = \log_2 \frac{p'_i + q'_i}{q'_i}$, and hence $p'_i = 2^{2(\pi_2 - \pi_1)}$, $p'_i = \alpha q'_i$. So

$$\sum_{i \in A_j} p'_i = \alpha \sum_{i \in A_j} q'_i, \quad P_j = \alpha Q_j, \quad \alpha = \frac{P_j}{Q_j}.$$

This means, that

$$\min_{p', q'} F_j(p', q') = \min_{q'} \sum_{i \in A_j} \left(\alpha q'_i \log_2 \frac{\alpha q'_i}{\alpha q'_i + q'_i} + q'_i \log_2 \frac{q'_i}{\alpha q'_i + q'_i} \right) =$$

$$= P_j \log_2 \frac{P_j}{P_j + Q_j} + Q_j \log_2 \frac{Q_j}{P_j + Q_j}.$$

As a consequence

$$\mathrm{JS}(p, q) \geqslant 1 + 0.5 \sum_{j : P_j > 0, Q_j > 0} \left(P_j \log_2 \frac{P_j}{P_j + Q_j} + Q_j \log_2 \frac{Q_j}{P_j + Q_j} \right) =$$

$$= 1 + 0.5 \sum_{j=1}^{m} \left(P_j \log_2 \frac{P_j}{P_j + Q_j} + Q_j \log_2 \frac{Q_j}{P_j + Q_j} \right).$$

□

We compared our lower bound on JS divergence to the lower bound presented derived in [23]. This bound is one of many results of long standing research of different divergence measures [21, 22].

The lower bound is given by

$$\mathrm{JS}(P,Q) \geqslant D(R\|U), \tag{10}$$

where

$$R = \left(\frac{1 - D_{TV}(P,Q)}{2}, \frac{1 + D_{TV}(P,Q)}{2}\right), \quad U = \left(\frac{1}{2}, \frac{1}{2}\right)$$

and total variation distance D_{TV} is given by

$$D_{TV}(P,Q) = \frac{1}{2}\sum_{x\in A} |P(x) - Q(x)|.$$

In section 3 we show that (2) is better than (10) in about 80% of cases, when we calculate both bounds on JS divergence calculated on (query, document) pairs, where we used up to 100 documents that are most similar to query (100 queries were randomly selected from dataset).

2.1. Applying lower bounds for calculation reduction. Let q be a probability distribution of terms in a query and P be group of item distributions. P may be a set of document probability distributions of terms or a set of cluster attractor probability distributions of terms. The main goal is the faster selection of item from P that has the smallest distance from q in terms of JS distance. To facilitate this, we utilize lower bounds on JS distance (2) presented in previous section. This approach requires a term clustering — partitioning of all terms presented in q or in distributions from P into $\bigcup_{j=1}^{m} A_j$. Term clustering allows the contraction of the query and documents or attractors so that the terms probability values occuring in one cluster are summed.

The technique that we apply is based on document frequency (DF) term clustering algorithm that orders terms by their document frequency in the corpus. The first cluster containing the least frequent terms whereas the last cluster contains most frequent terms. The idea of splitting more popular and less popular terms is supported by our goal — to contract query and documents in such manner that this contraction preserve similarity and dissimilarity between them. If we insert popular and non-popular terms in one cluster then in contraction popular terms can hide non-popular term and this results in larger similarity that it should be. So if we cluster non-popular terms separately from popular ones then preserved dissimilarity between query and documents and this should result in a better low bound. Our experiments confirm this idea (compare this clustering approach with random clustering, where popular and non-popular terms may appear in same cluster).

Cluster sizes also depends on term frequency and its real values mostly depends on a decay parameter α and total number of terms under consideration. Based on Zipf's law we can say that approximately the probability to get a term from any of these clusters doesn't depend on cluster index and number of clusters depends on the decay parameter value.

2.2. DF based term clustering algorithm. The pseudocode of the clustering algorithm is shown below.

Data: W — set of all terms from q and P, W_P — set of all terms that have positive probability in at least one probability distribution from P, W_q — set of all terms that have positive probability in query q, $\alpha \in (0,1)$ is decay parameter, $\beta > 0$ is minimum size of term cluster.
Result: Term clusters $W = A_1 \cup ... \cup A_m$.
// Partitioning of terms that occur in P: $W_p = B_1 \cup ... \cup B_k$:
$k = 1, W_0 = W_p$;
while $|W_{k-1}| > \beta$ **do**
  B_k = set of $\lceil \alpha |W_{k-1}| \rceil$ terms from W_{k-1} with minimum document frequencies;
  $W_k = W_p \setminus (B_1 \cup ... \cup B_k)$;
  $k = k + 1$;
end
$B_k = W_{k-1}$;
//All term clustering $W = A_1 \cup ... \cup A_k \cup A_{k+1} \cup A_{k+2} \cup A_{k+3}$:
$W_{\texttt{shared}} = W_p \cap W_q$;
$W_{\texttt{noisy1}} = W_p \setminus W_{\texttt{shared}}$;
$W_{\texttt{noisy2}} = W_q \setminus W_{\texttt{shared}}$;
$W_{\texttt{noisy3}} = W \setminus \left(W_{\texttt{shared}} \cup W_{\texttt{noisy1}} \cup W_{\texttt{noisy2}}\right)$;
$A_i = B_i \cap W_{\texttt{shared}},\ i = \overline{1,k}$;
$A_{k+1} = W_{\texttt{noisy1}}$;
$A_{k+2} = W_{\texttt{noisy2}}$;
$A_{k+3} = W_{\texttt{noisy3}}$;

Algorithm 1: Term clustering

Depending on the nature of the query some of the generated clusters may be empty.

2.3. Nearest neighbor algorithm utilising lower bounds on JS divergence. We consider two variants of nearest neighbor search algorithm:

- Brute-force search. In this variant we calculate distance between query and all documents from a corpus and return document that is in shortest distance from the query.
- Cluster-based search. In this variant we are seeking an approximate solution. Given a set of document clusters we:
  - — calculate distances between query and attractors of each cluster and select few attractors that are in shortest distance from the query,
  - — return document from clusters correspondent to selected attractors that is in shortest distance from the query.

Given a probability distribution q and a set of probability distributions P we wish to find

$$p^{NN} = \arg\min_{p \in P} JS(q, p).$$

As a group P, we either use set of probability distributions that represent all documents from a document corpus in brute force search algorithm or set of all documents from a document cluster in cluster-based search algorithm or set of probability distributions that represent all cluster attractors in cluster-based search algorithm.

Data: Query represented by probability distribution over terms $q = (q_1, ..., q_n)$, set of probability distributions over terms $P = \{p_1, ..., p_k\}, p_i = (p_{i1}, ..., p_{in})$.
Result: The closest probability distribution $p^{NN} \in P$.
Generate term clusters $A_1 \cup ... \cup A_m$ by DF based term clustering algorithm;
Calculate contracted query $q^c = (q_1^c, ..., q_m^c)$, where $q_i^c = \sum\limits_{j \in A_i} q_j$;
Best candidate lower bound $BLB = 1.0$;
Best candidate $p_{bc} = p_1$;
foreach $p_i \in P$ **do**
    Calculate contracted probability distribution $p_i^c = (p_{i1}^c, ..., p_{im}^c)$, where $p_{ik}^c = \sum\limits_{j \in A_k} p_{ij}$;
    Calculate lower bound on $\mathrm{JS}(q, p_i)$ as $\mathrm{JS}(q^c, p_i^c)$;
    if $BLB > JS(q^c, p_i^c)$ **then**
        $BLB = \mathrm{JS}(q^c, p_i^c)$;
        $p_{bc} = p_i$;
    end
end
Select best probability distribution

$$p^{NN} = \arg \min_{p_i : \mathrm{JS}(q^c, p_i^c) \leqslant \mathrm{JS}(q, p_{bc})} \mathrm{JS}(q, p_i);$$

return p^{NN};

Algorithm 2: Search algorithm

Note that the last 3 term clusters A_{m-2}, A_{m-1}, A_m can be ignored in calculation of contracted versions of probability distributions because for each of these clusters at least one of corresponding components in q^c or in p_i^c is equal to zero and hence lower bound $\mathrm{JS}(q^c, p_i^c)$ is not dependent on these clusters.

3. Experimental investigation. In our experiments used a document corpus that contains 1 854 654 articles published in New York Times from 01.01.1987 till 19.06.2007 (The New York Times Annotated Corpus, see http://catalog.ldc.upenn.edu/LDC2008T19). Applying the Contextual Document Clustering algorithm [18] resulted in the extraction of 21 431 contexts (attractors of document clusters).

In total this corpus contains 1 142 689 unique terms (stems of non-stop words). Each context is represented by probability distribution over up to 1000 terms. In total all contexts contain 154 010 unique terms.

As queries we selected randomly 100 documents from same document corpus. Queries are run as is — there is no dependency on any pre-compute data structure so the execution of queries is the same as if the query was based on a document outside the corpus. We assess the effect on performance based on the reduction in the number of log function calculations

$$\left(1 - \frac{\text{number of log-function calculations with use of lower bound}}{\text{number of log-function calculations without use of low bound}}\right).$$

The investigation focused brute force search and utilize these results. We want to compare standard brute force search and brute force search with low bounds and select optimal parameter values for term clustering algorithm to maximize reduction of log function calculations.

Our term clustering algorithm uses two parameters α and β. In all experiments parameter β has fixed value equal to 200, but for α we tested some different values. Table 1 shows how average (over 100 queries) reduction of log function calculation in brute force search with low bounds comparing to brute force search without low bounds depends on α.

Table 1. **Average reduction of** log **function calculations in brute force search with low bounds comparing to brute force search without low bounds**

α	Number of term clusters	Average reduction in log function calculations in brute force search with low bounds comparing to brute force search without low bounds
0.1	82	0.53881
0.3	25	0.59976
0.5	14	0.60052
0.7	9	0.58914
0.9	5	0.56606

We considered cluster-based search with and without lower bounds. Maximum reductions is achieved, when $\alpha = 0.5$ in DF based term clustering algorithm. Also used random term clustering with equal sizes of clusters to compare it with DF based term clustering, to show that mechanism of DF based clustering is more effective. We want to demonstrate that if mix frequent and non-frequent terms in same cluster (random clustering), then reduction of log function calculations will be smaller in comparison to DF based clustering that split frequent and infrequent terms.

In cluster based search need to generate two sets of term clusters. First of all needed to cluster unique terms that occur in cluster attractors (contexts). In total we have 154 010 such terms, which are clustered in 11 clusters by DF based clustering algorithm with α = 0.5. These clusters are used in best clusters selection step. Then in the second step we need to find nearest document in each of the selected best clusters. Here we use clusters of all unique terms that occur in documents (stop words were removed and Porter stemming was used). In total we have 984 341 such terms that were clustered into 14 clusters by DF based term clustering algorithm with α = 0.5. All clusters generated by DF based clustering have different sizes with largest cluster of rare or low frequency terms and smallest cluster of most popular or highest frequency terms. In random clusters experiments generated clusters of equal sizes with terms randomly assigned to clusters. Numbers of clusters were same as in DF term based clustering — 11 clusters for terms from attractors and 14 clusters for documents terms.

Table 2 shows results (averaged over 100 queries) of cluster-based search (up to 10 best clusters were selected for each query) for 3 variants:

– cluster-based search without use of low bounds;

– cluster-based search with low bounds, where DF based term clustering was used;

– cluster-based search with low bounds, where random term clustering was used.

For each variant we show reduction of log function calculations compared to brute force search and the accuracy. Accuracy is the percentage of queries for which most similar document returned by brute force search is also returned by cluster-based search. This result depends on the number of selected best clusters and increases with it. Average reduction of log function calculations is calculated comparing to number of this function calculations in brute force search. Accuracy shows percentage of queries, where cluster-based search returns same result as brute force search.

Table 2. **Cluster-based search with selection of up to 10 best clusters**

Number of best clusters selected	Accuracy	Average reduction of log function calculations in cluster based search		
		without lower bounds	with lower bounds (DF based term clustering with $\alpha = 0.5$)	with lower bounds (random term clustering)
1	0.44	0.93199	0.98744	0.97061
2	0.61	0.93081	0.98632	0.96921
3	0.69	0.93002	0.98548	0.96820
4	0.74	0.92891	0.98451	0.96691
5	0.76	0.92789	0.98354	0.96569
6	0.78	0.92720	0.98283	0.96482
7	0.82	0.92666	0.98219	0.96401
8	0.82	0.92614	0.98157	0.96332
9	0.82	0.92579	0.98109	0.96275
10	0.83	0.92532	0.98059	0.96224

From the presented results we can see that in cluster based search we can achieve 0.83 accuracy, if select up to 10 best clusters. Cluster-based search with lower bounds based on DF based term clustering outperform significantly cluster based search without low bounds. In the case of 10 best clusters selection the number of log function calculations with low bounds on average is

$$\frac{1.0 - 0.92532}{1.0 - 0.98059} = 3.8475$$

times smaller comparing to the case without low bounds. Comparing to random term clusters case DF based term clustering results in

$$\frac{1.0 - 0.96224}{1.0 - 0.98059} = 1.9454$$

times smaller in terms of the number of log function calculations.

We compared the lower bound method (2) to lower bound (10). For each of 100 queries (documents randomly selected from NYT dataset) calculated JS divergence, and the respective bounds between the query and all other documents from NYT. Then select top X nearest neighbors (in terms of JS divergence) and calculated percentage of cases, when the bound (2) is closer to the actual JS divergence (larger) than bound (10). In Figure we present results averaged over 100 queries for X in between 1 and 100.

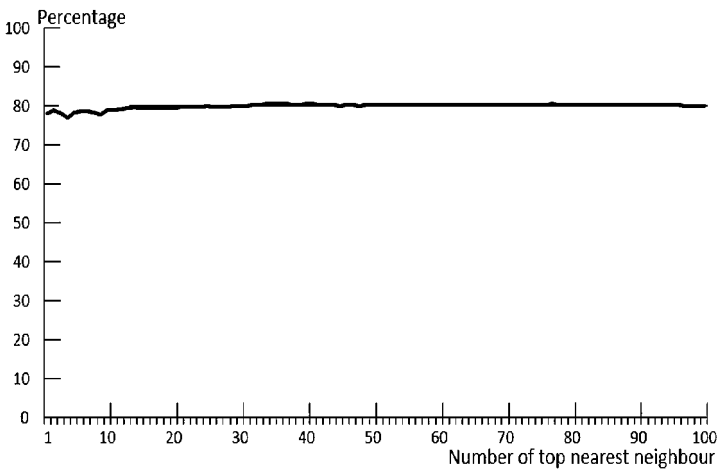

Figure. Percentage of cases when bound (2) is better than (10)

X-axis shows number of top nearest neighbors used in comparison of two bounds. Y-axis shows percentage of cases, when our bound is better.

4. Conclusions. In this paper, by determining lower bounds on JS divergence, we can significantly reduce the number of calculations to determine the nearest neighbours both in brute force and approximate search. As JS divergence is utilize in a number of different mechanisms such as distributional similarity [20] or bioinformatics [24] our findings may have application to a number of different problem areas. Our low bound on JS divergence depends on query and this means that we need to use effective algorithm for on-line generation of contracted version of large number of probability distributions. In future are planning to develop an efficient data structure for this task.

References

1. Manning C. D., Raghavan P., Schütze H. *Introduction to information retrieval*. Cambridge, Cambridge University Press, 2008, 544 p.

2. Bentley J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 1975, vol. 18, no. 9, pp. 509–517.

3. Samet H. *Foundations of multidimensional and metric data structures*. San-Francisco, Morgan Kaufmann Publ., 2006, 1024 p.

4. Cayton L. Fast nearest neighbor retrieval for bregman divergences. *Proceedings of the 25th Intern. conference on Machine Learning*. San Francisco, Morgan Kaufmann Publ., 2008, pp. 112–119.

5. Coviello E., Mumtaz A., Chan A., Lanckriet G. That was fast! Speeding up NN search of high dimensional distributions. *Proceedings of the 25th Intern. conference on Machine Learning*. San Francisco, Morgan Kaufmann Publ., 2013, pp. 468–476.

6. Li C., Chang E., Garcia-Molina H., Wiederhold G. Clustering for approximate similarity search in high-dimensional spaces. *Knowledge and Data Engineering, IEEE Transactions on*, 2002, vol. 14, no. 4, pp. 792–808.

7. Arya S., Mount D. M. Approximate nearest neighbor queries in fixed dimensions. *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms. SODA*, Society for Industrial and Applied Mathematics, 1993, pp. 271–280.

8. Andoni A., Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Foundations of Computer Science. FOCS'06. 47th Annual IEEE Symposium on*, 2006, pp. 459–468.

9. Indyk P., Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality. *Proceedings of the 30th Annual ACM Symposium on theory of computing*, 1998, pp. 604–613.

10. Andoni A., Indyk P., Nguyen H. L., Razenshteyn I. Beyond Locality-Sensitive Hashing. *SODA*, 2014, pp. 1018–1028.

11. Li K., Malik J. *Fast k-nearest neighbour search via prioritized DCI*. ArXiv preprint ArXiv:1703.00440, 2017.

12. Li K., Malik J. Fast k-nearest neighbour search via dynamic continuous indexing. *Intern. Conference on Machine Learning*, 2016, pp. 671–679.

13. Pereira F., Tishby N., Lee L. Distributional clustering of English words. *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, 1993, pp. 183–190.

14. Baker L. D., McCallum A. K. Distributional clustering of words for text classification. *Proceedings of the 21st annual Intern. ACM SIGIR conference on Research and Development in Information Retrieval*, 1998, pp. 96–103.

15. Slonim N., Tishby N. The power of word clusters for text classification. *23rd European Colloquium on Information Retrieval Research*, 2001, vol. 1, p. 200.

16. Bekkerman R., El-Yaniv R., Tishby N., Winter Y. On feature distributional clustering for text categorization. *Proceedings of the 24th annual Intern. ACM SIGIR conference on Research and Development in Information Retrieval*, 2001, pp. 146–153.

17. Dhillon I. S., Mallela S., Kumar R. A divisive information theoretic feature clustering algorithm for text classification. *The Journal of Machine Learning Research*, 2013, vol. 3, pp. 1265–1287.

18. Dobrynin V., Patterson D., Rooney N. Contextual document clustering. *Advances in Information Retrieval*. Berlin, Heidelberg, Springer Publ., 2004, pp. 167–180.

19. Lin J. Divergence measures based on the Shannon entropy. *Information Theory, IEEE Transactions on*, 1991, vol. 37, no. 1, pp. 145–151.

20. Lee L. Measures of distributional similarity. *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, 1999, pp. 25–32.

21. Topsoe F. Some inequalities for information divergence and related measures of discrimination. *Information Theory, IEEE Transactions on*, 2000, vol. 46, no. 4, pp. 1602–1609.

22. Guntuboyina A., Saha S., Schiebinger G. Sharp inequalities for f-divergences. *IEEE Transactions on Information Theory*, 2014, vol. 60, no. 1, pp. 104–121.

23. Sason I. L. Tight bounds for symmetric divergence measures and a refined bound for lossless source coding. ArXiv preprint. ArXiv:1403.7164 [cs.IT], 2014.

24. Capra J. A., Singh M. Predicting functionally important residues from sequence conservation. *Bioinformatics*, 2007, vol. 23, no. 15, pp. 1875–1882.

Received: June 5, 2018
Accepted: September 25, 2018.

Author's information:

Vladimir Yu. Dobrynin — PhD in Physics and Mathematics, Associate Professor; v.dobrynin@bk.ru

Niall Rooney — PhD in Informatics; niall@aiqudo.com

Julian A. Serdyuk — Master; julian@aiqudo.com

Вычисление нижней границы дивергенции Дженсена—Шеннона и ее применение к задаче поиска ближайшего соседа

В. Ю. Добрынин[1], *Н. Ронни*[2], *Ю. А. Сердюк*[3]

[1] Санкт-Петербургский государственный университет, Российская Федерация, 199034, Санкт-Петербург, Университетская наб., 7–9
[2] Sophia Ltd, Инновационный парк Северной Ирландии, Северная Ирландия, BT3 9DT, Белфаст, Королевское шоссе, Королевский остров
[3] Московский государственный университет имени М. В. Ломоносова, Российская Федерация, 119991, Москва, Ленинские горы, 1

Для цитирования: *Dobrynin V. Yu., Rooney N., Serdyuk J. A.* Setting lower bounds on Jensen—Shannon divergence and its application to nearest neighbor document search // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2018. Т. 14. Вып. 4. С. 334–345. https://doi.org/10.21638/11702/spbu10.2018.406

Дивергенция Дженсена—Шеннона используется для определения ближайших соседей в коллекции документов для конкретного документа запроса. Это эффективный механизм, однако исчерпывающий поиск может оказаться трудоемким процессом. В этой статье покажем, определив нижнюю оценку дивергенции Дженсена—Шеннона, что возможно сократить объем вычислений на 60% для исчерпывающего поиска и на 98% для приближенного поиска на основе выполнения поиска ближайшего соседа в одной реальной коллекции документов. В этих экспериментах был применен корпус документов, который содержит 1 854 654 статьи, опубликованные в New York Times с 01.01.1987 по 19.06.2007 (The New York Times Annotated Corpus). В качестве запросов были выбраны 100 случайных документов из данного корпуса документов. Оценивается влияние на производительность на основе сокращения количества вызовов логарифмической функции. Приближенный поиск ближайшего соседа основан на кластеризации документов с помощью алгоритма контекстной кластеризации. Выполняется приближенный поиск ближайшего соседа путем нахождения некоторого набора аттракторов кластеров,

которые наиболее близки запросу, и далее ограничивается поиск документов в кластерах, соответствующих выбранным аттракторам.

Ключевые слова: дивергенция Дженсена—Шеннона, поиск ближайшего соседа, снижение размерности.

Контактная информация:

Добрынин Владимир Юрьевич — канд. физ.-мат. наук, доц.; v.dobrynin@bk.ru

Ронни Найл — PhD in Informatics; niall@aiqudo.com

Сердюк Юлиан Анатольевич — магистр; julian@aiqudo.com