An Empirical Comparison of FAISS and FENSHSES for Nearest Neighbor Search in Hamming Space

Cun (Matthew) Mu[†]
Walmart Labs
Hoboken, NJ
matthew.mu@jet.com

Binwei Yang[†]
Walmart Labs
Sunnyvale, CA
BYang@walmartlabs.com

Zheng (John) Yan
Walmart Labs
Hoboken, NJ
john@jet.com

ABSTRACT

In this paper, we compare the performances of FAISS and FENSHSES on nearest neighbor search in Hamming space–a fundamental task with ubiquitous applications in nowadays eCommerce. Comprehensive evaluations are made in terms of indexing speed, search latency and RAM consumption. This comparison is conducted towards a better understanding on trade-offs between nearest neighbor search systems implemented in main memory and the ones implemented in secondary memory, which is largely unaddressed in literature.

CCS CONCEPTS

• **Information systems → Nearest-neighbor search**; **Image search**; • **Applied computing → Online shopping**; • **Software and its engineering → Memory management**;

KEYWORDS

Nearest neighbor search, FAISS, FENSHSES, Hamming space, Binary codes, Vector similarity search, Full-text search engines, Elasticsearch

ACM Reference Format:
Cun (Matthew) Mu[†], Binwei Yang[†], and Zheng (John) Yan. 2019. An Empirical Comparison of FAISS and FENSHSES for Nearest Neighbor Search in Hamming Space. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR 2019 eCom)*. ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Nearest neighbor search (NNS) within semantic embeddings (a.k.a., vector similarity search) has become a common practice in ubiquitous eCommerce applications including neural ranking model based text search [3, 12], content-based image retrieval [16, 22], collaborative filtering [7], large-scale product categorization [8], fraud detection [18], etc. While vector similarity search is capable of substantially boosting search relevancy by understanding customers' intents more semantically, it presents a major challenge: how to conduct nearest neighbor search among millions or even billions of high-dimensional vectors in a real-time and cost-effective manner.

[†] C. Mu and B. Yang contributed equally to this work.

SIGIR 2019 eCom, July 2019, Paris, France

The fundamental trade-off between search latency and cost-effectiveness would naturally classify nearest neighbor search solutions into two broad categories.

NNS solutions implemented in main memory. This type of NNS solutions has been extensively studied and explored in the field of information retrieval (IR). As a result, the majority of those widely used ones (e.g., Spotify's Annoy [2], Facebook's FAISS [9] and Microsoft's SPTAG [5, 21]) in nowadays software market fall into this category.

NNS solutions implemented in secondary memory. In contrast, the second type of NNS solutions are delivered only recently by active efforts from both academia and industry [1, 11, 14, 16, 19, 20] to empower full-text search engines (e.g., Elasticsearch and Solr) with the capability of finding nearest neighbors. By leveraging inverted-index-based information retrieval systems and cutting-edge engineering designs from these full-text search engines, such full-text search engine based solutions are capable of economically reduce RAM consumption [1], incoherently supporting multi-model search [16] and being extremely well-prepared for production deployment [20]. However, some of the critical performance questions have not been quantitatively answered in literature:

- how much RAM could these full-text search based solutions save?
- how much search latency would these solutions sacrifice in order to reduce RAM consumption?

In this paper, we will shed light on the above questions through a case study on the task of nearest neighbor search in Hamming space (i.e., the space of binary codes). This task is an extremely important subclass of NNS, as learning and representing textual, visual and acoustic data with compact and semantic binary vectors is a pretty mature technology and common practice in nowadays IR systems. In particular, eBay recently builds its whole visual search system [22] upon finding nearest neighbors within binary embeddings generated through deep neural network models.

We choose one representative solution of each category–FAISS (Facebook AI Similarity Search) from Facebook's AI Research Lab [9] and FENSHSES (Fast Exact Neighbor Search in Hamming Space on Elasticsearch) from the search and catalog teams at Walmart Labs [14, 15]–to evaluate their performances in finding nearest neighbors within binary codes.

2 FAISS vs. FENSHSES

We will compare performances of FAISS and FENSHSES from three key perspectives: time spent in indexing, search latency and RAM consumption.

Data generation. Our dataset $\mathcal{B}$ is generated using 2.8 million images selected from Walmart.com's home catalog through pHash [6, 10]–one of the most effective perceptual hash schemes in generating fingerprints for multimedia files (e.g. images, audios and videos)–with $m \in \{64, 256, 1024, 4096\}$ respectively. Note that vector similarity search based on pHash has been widely used in a variety of visual tasks including forensic image recognition [17], duplicate image detection [4] and copyright protection [13], etc.

Settings. For FAISS, we use its binary flat index with five threads. For a fair comparison, we accordingly deploy FENSHSES by creating its Elasticsearch index with five shards and zero replica. The rest of configurations are left as their default and suggested values. Both FAISS and FENSHSES are set up and tested on the same Microsoft Azure virtual machine.

Speed in indexing. During the indexing phase, FAISS indexes the data into main memory (i.e., RAM), while FENSHSES indexes the data into secondary memory (e.g., hard disk). As a consequence, FAISS is much faster than FENSHSES in terms of data indexing (see Table 1). But on the other hand, whenever the process is killed and needs a restart, FAISS has to go through this procedure again to re-index data into RAM, while FENSHSES could unaffectedly use its built index on hard disk without re-indexing.

# of Bits	FAISS (sec.)	FENSHSES (sec.)
64	18.5	75.5
256	37.7	140.2
1024	111.9	369.5
4096	397.3	1300.9

Table 1: Indexing time consumption. FAISS is about four times faster than FENSHSES in creating the index for nearest neighbor search.

Search latency. We randomly select 10,000 binary codes from $\mathcal{B}$ to act as query codes. For each query code $\boldsymbol{q}$, we instruct FAISS and FENSHSES to find all r-neighbors of $\boldsymbol{q}$ in $\mathcal{B}$, namely

$$B_H(\boldsymbol{q}, r) := \{\boldsymbol{b} \in \mathcal{B} \mid d_H(\boldsymbol{b}, \boldsymbol{q}) \leq r\}, \tag{2.1}$$

where $d_H(\boldsymbol{b}, \boldsymbol{q}) := \sum_{i=1}^{m} \mathbb{1}_{\{b_i \neq q_i\}}$ denotes the Hamming distance between binary code $\boldsymbol{b}$ and $\boldsymbol{q}$, and the Hamming radius $r \geq 0$. As shown in Table 2, FENSHSES is quite competitive for small radium r. This is because FENSHSES fully leverages Elasticsearch's inverted index to first conduct a sub-code filtering to only consider a subset of $\mathcal{B}$ for Hamming distance computation, which is most effective for small r. In contrast, FAISS scans every binary code in $\mathcal{B}$, so its search latency is almost invariant with respect to r. For applications (e.g., near-duplicate image detection and visual search) where we care most about nearest neighbors within a small radius, FENSHSES could be in a more favorable position than FAISS.

RAM consumption. Since FAISS is implemented in main memory, its RAM consumption undoubtedly rises along with the increase in the size of dataset $\mathcal{B}$, as shown in Table 3. In contrast, by leveraging the highly optimized disk-based index mechanics behind full-text search engines, FENSHSES consumes a much smaller amount of RAM when conducting nearest neighbor search. This property makes FENSHSES more cost-effective and thus more suitable especially to big-data applications.

# of Bits	r	FAISS (ms)	FENSHSES (ms)
64	3	34.0	**5.8**
	7	37.0	**25.7**
	11	**42.7**	117.7
256	15	42.9	**7.8**
	31	42.8	**22.5**
	47	**45.4**	77.7
1024	63	79.2	**31.6**
	127	**81.9**	89.9
	191	**90.4**	250.0
4096	255	222.7	**134.2**
	511	**223.2**	612.5
	767	**223.3**	1797.5

Table 2: Search latency. FENSHSES is quite competitive for r-neighbor search when the Hamming distance r is small, while the performance of FAISS is pretty robust with respect to r. This provides FAISS and FENSHSES different edges for the task of NNS.

# of Bits	r	FAISS (GB)	FENSHSES (GB)
64	3	2.2	1.6
	7	2.2	1.6
	11	2.2	1.6
256	15	2.3	1.6
	31	2.3	1.6
	47	2.3	1.6
1024	63	2.9	1.6
	127	2.9	1.6
	191	2.9	1.6
4096	255	4.9	1.6
	511	4.9	1.6
	767	4.9	1.6

Table 3: Main memory (RAM) consumption. The RAM consumed by FAISS substantially grows with the increase in the size of dataset $\mathcal{B}$. In contrast, FENSHESE consumes a constant amount of RAM, which is much smaller than the one consumed by FAISS.

3 CONCLUSION

In this case study, we compare FAISS and FENSHSES for the task of nearest neighbor search in Hamming space. By evaluating their performances in terms of speed in data indexing, search latency

and RAM consumption, we hope practitioners could now better understand the pros and cons of the main memory based NNS solutions and the secondary memory based ones, and thus make their best choices accordingly (at least in NNS systems within binary cods). In the future, we will compare FAISS and FENSHSES under a wider range of applications; and moreover, we will also go beyond Hamming space to evaluate vector similarity search systems for general NNS problems.

ACKNOWLEDGEMENT

We are grateful to three anonymous reviewers for their helpful suggestions and comments that substantially improve the paper. CM would like to thank Jun Zhao and Guang Yang for insightful discussions on FENSHSES. BY would like to thank Alessandro Magnani for helpful discussions on pHash and its related applications, and Zuzar Nafar for his support on this study.

REFERENCES

[1] G. Amato, P. Bolettieri, F. Carrara, F. Falchi, and C. Gennaro. 2018. Large-Scale Image Retrieval with Elasticsearch. In *SIGIR*.
[2] Erik B. 2018. *Annoy: Approximate Nearest Neighbors in C++/Python.* https://pypi.org/project/annoy/ Python package version 1.13.0.
[3] E. P. Brenner, J. Zhao, A. Kutiyanawala, and Z. Yan. 2018. End-to-End Neural Ranking for eCommerce Product Search. In *SIGIR eCommerce Workshop*.
[4] A. Chaudhuri, P. Messina, S. Kokkula, A. Subramanian, A. Krishnan, S. Gandhi, A. Magnani, and V. Kandaswamy. 2018. A Smart System for Selection of Optimal Product Images in E-Commerce. In *Big Data*.
[5] Qi Chen, Haidong Wang, Mingqin Li, Gang Ren, Scarlett Li, Jeffery Zhu, Jason Li, Chuanjie Liu, Lintao Zhang, and Jingdong Wang. 2018. *SPTAG: A library for fast approximate nearest neighbor search.* https://github.com/Microsoft/SPTAG
[6] Z. Christoph. 2010. Implementation and Benchmarking of Perceptual Image Hash Functions. In *Upper Austria University of Applied Sciences*. Hagenberg Campus.
[7] M. Deshpande and G. Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.
[8] H. Hu, R. Zhu, Y. Wang, W. Feng, X. Tan, and J. Huang. 2018. A Best Match KNN-based Approach for Large-scale Product Categorization. In *SIGIR eCommerce Data Challenge*.
[9] J. Johnson, M. Douze, and H. Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).
[10] E. Klinger and D. Starkweather. 2010. *pHash–the open source perceptual hash library.* Technical Report. accessed 2016-05-19.[Online]. Available: http://www.phash. org/apps.
[11] M. Lux and O. Marques. 2013. *Visual Information Retrieval Using Java and LIRE*. Vol. 25. Morgan & Claypool Publishers.
[12] A. Magnani, F. Liu, M. Xie, and S. Banerjee. 2019. Neural Product Retrieval at Walmart. com. In *WWW Workshop on eCommerce and NLP*. ACM.
[13] R. Mehta, N. Kapoor, S. Sourav, and R. Shorey. 2019. Decentralised Image Sharing and Copyright Protection using Blockchain and Perceptual Hashes. In *COMSNETS*.
[14] C. Mu, J. Zhao, G. Yang, B. Yang, and Z. Yan. 2019. Empowering Elasticsearch with Exact and Fast r-Neighbor Search in Hamming Space. *arXiv preprint arXiv:1902.08498* (2019).
[15] C. Mu, J. Zhao, G. Yang, B. Yang, and Z. Yan. 2019. Fast and Exact Nearest Neighbor Search in Hamming Space on Full-Text Search Engines. In *SISAP*.
[16] C. Mu, J. Zhao, G. Yang, J. Zhang, and Z. Yan. 2018. Towards Practical Visual Search Engine Within Elasticsearch. In *SIGIR eCommerce Workshop*.
[17] A. Peter, T. Hartmann, S. Müller, and S. Katzenbeisser. 2012. Privacy-preserving architecture for forensic image recognition. In *WIFS*.
[18] A. Raghava-Raju. 2017. Predicting Fraud in Electronic Commerce: Fraud Detection Techniques in E-Commerce. *International Journal of Computer Applications* 171, 2 (2017).
[19] M. Ruzicka, V. Novotny, P. Sojka, J. Pomikalek, and R. Rehurek. 2017. Flexible Similarity Search of Semantic Vectors Using Fulltext Search Engines. In *ISWC HSSUES Workshop*.
[20] J. Rygl, J. Pomikalek, R. Rehurek, M. Ruzicka, V. Novotny, and P. Sojka. 2017. Semantic Vector Encoding and Similarity Search Using Fulltext Search Engines. In *RepL4NLP Workshop*.
[21] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li. 2012. Scalable k-nn graph construction for visual descriptors. In *CVPR*.
[22] F. Yang, A. Kale, Y. Bubnov, L. Stein, Q. Wang, H. Kiapour, and R. Piramuthu. 2017. Visual search at ebay. In *KDD*.