FlashCheck: Exploration of Efficient Evidence Retrieval for Fast Fact-Checking

Kevin Nanhekhan[1(✉)], V. Venktesh[1], Erik Martin[2,3], Henrik Vatndal[2,3], Vinay Setty[2,3], and Avishek Anand[1]

[1] Delft University of Technology, Delft, The Netherlands
kevin.nanhekhan@hotmail.nl
[2] University of Stavanger, Stavanger, Norway
[3] Factiverse AI, Stavanger, Norway

Abstract. The advances in digital tools have led to the rampant spread of misinformation. While fact-checking aims to combat this, manual fact-checking is cumbersome and not scalable. It is essential for automated fact-checking to be efficient for aiding in combating misinformation in real-time and at the source. Fact-checking pipelines primarily comprise a knowledge retrieval component which extracts relevant knowledge to fact-check a claim from large knowledge sources like Wikipedia and a verification component. The existing works primarily focus on the fact-verification part rather than evidence retrieval from large data collections, which often face scalability issues for practical applications such as live fact-checking. In this study, we address this gap by exploring various methods for indexing a succinct set of factual statements from large collections like Wikipedia to enhance the retrieval phase of the fact-checking pipeline. We also explore the impact of vector quantization to further improve the efficiency of pipelines that employ dense retrieval approaches for first-stage retrieval.We study the efficiency and effectiveness of the approaches on fact-checking datasets such as HoVer and WiCE, leveraging Wikipedia as the knowledge source. We also evaluate the real-world utility of the efficient retrieval approaches by fact-checking 2024 presidential debate and also open source the collection of claims with corresponding labels identified in the debate. Through a combination of indexed facts together with Dense retrieval and Index compression, we achieve up to a **10.0x** speedup on CPUs and more than a **20.0x** speedup on GPUs compared to the classical fact-checking pipelines over large collections.

Keywords: Index Compression · fact-checking · Vector quantization

1 Introduction

The rapid dissemination of disinformation and misinformation in the digital era has catastrophic consequences, impacting public opinion. Since manual fact-checking is not scalable and time-consuming, automated fact-checking approaches have been proposed [7,11,26,30] to combat misinformation. Automated fact-checking pipelines mimic the human fact-checking process by collecting multiple pieces of evidence for different aspects of the claim, followed

C. Hauff et al. (Eds.): ECIR 2025, LNCS 15575, pp. 385–399, 2025.
https://doi.org/10.1007/978-3-031-88717-8_28

by reasoning over evidence for claim verification [1,20]. Hence, these pipelines are usually comprised of a claim detection component, an evidence/knowledge retrieval component and a fact verification component [12,14,22,28]. Mitigating misinformation spread through platforms such as political debates and election campaigns, is time-critical due to its potential to proliferate faster, sway public opinion and its perceived reliability [9,18,21]. Hence, it is essential that the verification pipeline is efficient to verify information at the source in real-time.

The Retrieval Bottleneck: One of the primary bottlenecks in existing fact-checking pipelines is the retrieval component employed for extracting information from large knowledge sources like Wikipedia. Existing approaches primarily employ sparse retrieval approaches followed by re-ranking or dense retrieval approaches [12,24,25] which perform search over a large index incurring huge computational, memory costs and high latency during inference. For instance, indexing and performing dense retrieval over the entire Wikipedia is prohibitively expensive with embedding index size being (**9.70 GiB**) and average retrieval latency per query (from WiCE) being **610 ms**. While several efficient sparse and dense Information Retrieval (IR) approaches exist [4,5,13,17], prior works in fact-checking primarily focus on fact verification components with very few works focusing on improving retrieval effectiveness [35]. However, to the best of our knowledge, our work is the first to explore principled efficient IR approaches applied for fact-checking along with a case-study of its effectiveness for real-time fact-checking.

Objectives: We envision a fact-checking pipeline with an efficient retrieval component that has *low memory footprint, computation requirements* and *low latency* compared to existing systems. This would enable to run such systems in **low-resource settings**, making the technology more **accessible** to effectively combat misinformation at scale and possibly in real-time.

Hence, in this work, we explore several approaches to extract and index succinct facts from large knowledge sources to aid in fact-checking claims. We observe that the corpus of knowledge sources such as Wikipedia could be compressed to **59%** of the original information, optimizing sparse retrieval. Further, we also optimize dense retrieval by exploring vector quantization approaches for compressing the vector index of extracted factual statements which brings down the embeddings index size to **672.95 MiB** from 9.70 GiB. We observe that this approach leads to speedups of up to **10x** on the CPU and **20x** over the classic retrieve-rerank pipelines with the original corpus. The approaches we study also obviate the need for the snippet selection stage in fact-checking pipelines as it already indexes the succinct facts resulting in efficient inference. Through extensive experiments, we answer the following research questions:

RQ1: How does indexing factual statements for large collections affect information retrieval efficiency?

RQ2: How does indexing factual statements affect overall pipeline efficiency and downstream fact-checking performance?

RQ3: How does does index compression affect the efficiency of dense retrieval and overall fact-checking pipeline?

Following are our core contributions:

- We explore approaches aimed at efficiently identifying relevant text spans from large corpora pertaining to the claims being fact-checked, enabling efficient sparse retrieval and verification processes.
- We show the methods for indexing factual knowledge coupled with index compression further optimizes dense retrieval efficiency by reducing computational overhead compared to classical retrieve-rank-select pipelines without significant loss in performance.
- We deploy the efficient retrieval system for fact-checking the 2024 presidential debate in real-time and also open source the dataset of facts identified with corresponding human annotated labels.

Reproducibility: We open-source all our data and code under: https://github.com/kevin-rn/Efficient-Fact-checking.

2 Related Work

The rapid proliferation of misinformation and disinformation has given rise to the development of automated fact-checking systems to combat them [7,22,28]. The automated fact-checking process involves three stages comprising *claim detection*, which identifies salient spans to be fact-checked, followed by *evidence retrieval* that focuses on identifying sources that support or refute claims and finally the fact-verification stage that uses the evidence collected to categorize the claims.

A significant challenge in verification includes source reliability [7], hence fact-checking primarily relies on verified knowledge sources. Sources such as encyclopedias, policy documents, and scientific journals are common knowledge sources employed for retrieving information [16,27]. Recent advancements advocate a simplified two-step evidence retrieval approach: a context retriever selects a subset of passages that might contain the answer followed by a machine reader analyzing these passages to identify the correct answer. Initially, evidence retrieval relied on inverted indexes (e.g., TF-IDF, BM25) for keyword-based searches but these lack semantic understanding [2,29]. The advances in representation learning have led to the rise of dense retrieval where queries and documents are projected to a continuous vector space [6,15,33]. While existing approaches employ brute force search in the vector space to retrieve relevant documents for the queries, this is not scalable for web-scale search [3,8,10,29,36]. Compact vector representations are crucial for efficiency, despite potential noise-induced performance drops [32]. Efficient approaches like Approximate Nearest Neighbors (ANN) search in vector databases have emerged to reduce complexity and enhance similarity search accuracy [8,29,34]. Early on hash-based and tree-based methods were used but faced limitations in large-scale databases and semantic features. Recent advancements make use of quantization techniques, such as Product Quantization [13] and Optimized PQ (OPQ) [5], to improve efficiency by vector dimension reduction with minimal performance loss.

Our contributions aim to explore efficient retrieval approaches to improve the efficiency of the fact-checking process, enhancing the practical applicability

of these systems in real-world scenarios such as live fact-checking over large knowledge bases.

3 Methodology

In this section, we describe our proposed improvements on how to enhance fact-checking efficiency by optimizing the retrieval of evidence from knowledge sources. This involves constructing a pruned index of succinct facts from large sources, like Wikipedia, coupled with vector quantization for index compression to improve retrieval latency and reduce resource consumption, as illustrated in Fig. 1.

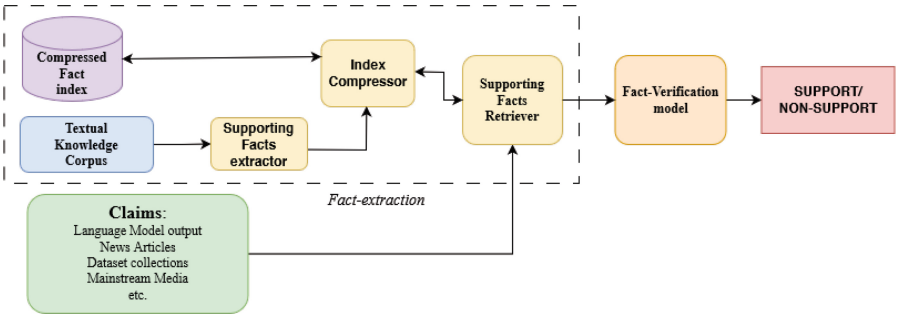


Fig. 1. Comparison of Existing and Proposed Fact-Checking Pipelines

3.1 Corpus Compression Through Extraction of Facts

Large collections like Wikipedia are usually employed as knowledge sources for fact-checking [12,14]. While each Wikipedia article has detailed information on a topic, all the information is not factual, lacking citations and less informative for purposes of verifying claims. Hence, identifying useful factual information for fact-checking from such a large corpus and pruning other parts can significantly reduce pipeline runtime and save disk space for indexing. We employ different mechanisms to identify and store factual statements from Wikipedia corpus.

Since factual statements are precise statements that are verifiable, we posit that claim detection can help identify such succinct facts. Hence, we first employ a claim-detection model to identify significant (salient) sentences within the text, aiming to extract key information effectively. For experimental purposes, a model trained on a claim detection dataset such as Claimbuster is leveraged. We term this approach **Fact Extraction (FE)**.

However, this approach could result in false positives leading to statements that do not contain factual information or aid in verifying claims. It would also result in false negatives as the statements the claim detection model fails to identify due to distribution mismatch would lead to incorrect verdicts. To circumvent this, we explore an alternative approach with a stricter criterion of retaining only verified factual statements on Wikipedia articles. We retain only

those snippets that comprise citations following Wikipedia's *verifiability*[1] and *citation*[2] guidelines that the cited statements ensure verifiability of information conveyed. This method eliminates the need for training a specific model, providing a straightforward way to identify factual statements. We term this approach **Citation Extraction (CE)** in our experiments. However, it may miss contextually related sentences, such as those in Wikipedia, where consecutive sentences can be supported by a single citation, resulting in false negatives.

To balance the tradeoffs in the above approaches, we focus on minimizing false negatives to capture all information that would aid in verifying claims. We propose to fuse the snippets identified by the claim detection-based approach and the citation extraction approach while pruning other information from the Wiki pages. While the problem of false positives from claim detection would persist, we posit that this can be minimized through retrieval and re-ranking approaches that optimize for recall. The fusion of the former approaches aims to leverage the strengths of each approach: citations for reliable support and claim detection for comprehensive coverage. We term this approach **Fusion (Fu)** in our experiments.

While pruning the Wikipedia collection helps reduce memory requirements for storing and using the plaintext collection, it also helps optimize latency for sparse retrieval approaches like BM25. However, dense retrieval approaches provide better semantic relevance estimates than sparse retrieval and hence are employed in many fact-checking pipelines. However, dense retrieval is expensive due to index size comprising document vector representations and also has high latency due to brute force search approaches for retrieving relevant documents for a query. Hence, in this work, we explore an index compression mechanism that reduces index size, while ensuring optimal performance. We term this approach **Index Compression** in our experiments.

3.2 Index Compression

Existing fact-checking systems usually adopt sparse retrieval followed by re-ranking or dense retrieval approaches to capture better semantics for first-stage retrieval. However, dense retrieval using brute force search for web-scale data is not scalable. To enhance efficiency in Dense retrieval, we propose to compress the corpus embeddings index through the JPQ index compression algorithm [32]. Unlike existing approaches that treat training of encoders and learning compressed index separately, JPQ optimizes them jointly by employing a ranking-oriented loss produced by the encoders and the index. JPQ achieves an impressive compression ratio of $4D/M$ (where D is vector dimensionality and M is the number of codebooks (embedding sets)) [32]. This approach ensures efficient retrieval with a speedup ratio of $(D + \log n)/(M + \log n)$, where n is the total number of documents, maintaining performance comparable to standard Dense Retrieval setups while maintaining a small memory footprint for the index. JPQ is based on Product Quantization (PQ) which quantizes the document embeddings $\vec{d}^{\phi}$

[1] https://en.wikipedia.org/wiki/Wikipedia:Verifiability.
[2] https://en.wikipedia.org/wiki/Wikipedia:Citing_sources.

where ϕ denotes they are quantized. Quantization occurs by generating M sets of embedding, where each set has K embeddings of dimension D/M called centroid embeddings $\vec{c}_{i,j}$. A quantized embedding for a document is constructed by selecting one embedding from each set and concatenating them. The matching between queries and quantized embeddings is performed using approximate score function $s^{\phi}(q,d) = \langle \vec{q}, \vec{d^{\phi}} \rangle$. The search in PQ happens efficiently between query embeddings and quantized document embeddings by generating query sub-vectors and matching them to centroid embeddings.

The traditional dense retrieval models [19,31] use the pairwise loss, $l(s(q,d^+), s(q,d^-))$ where q is the query and $d^+ \in D_q^+$ indicates a positive document and $d^- \in D_q^+$ indicates a negative document with respect to the query. To adopt this loss for quantization setting the score function $s(q,d)$ has to be replaced by $s^{\phi}(q,d)$. Prior quantization approaches do not adopt a ranking-oriented loss and train the retrievers/dual-encoders independently of compression. To jointly train the quantization and retrieval approach end-end, JPQ trains the query encoder and PQ index jointly for the complete optimization objective formulated as

$$\langle f^*, \{c_{i,j}\}^* \rangle = \arg \min_{f, \{c_{i,j}\}} \sum_i \sum_{d^+ \in D_i^+} \sum_{d^- \in D_i^{\phi-}} \ell(s^{\phi}(q,d^+), s^{\phi}(q,d^-)), \qquad (1)$$

where $D_i^{\phi-}$ refers to the retrieved hard-negatives. Optimizing encoders to penalize hard-negatives is crucial to retrieval performance and JPQ obtains real-time hard-negatives by retrieving hard-negatives with respect to the current query from the quantized index being learned simultaneously thus optimizing for retrieval and quantization performance.

4 Experimental Setup

4.1 Datasets

HoVer [12] consists of 26k claims requiring evidence from up to four English Wikipedia articles (2017 data dump) and was developed in multiple stages with the help of trained crowd-workers. The dataset is split into 18,171 labeled training examples (11,023 Supported and 7,148 Not-Supported), 4,000 labeled development examples (2,000 Supported and 2,000 Not-Supported), and 4,000 unlabeled test examples.

WiCE [14] focuses on claims from Wikipedia articles. It uses the same base claims as SIDE [23] and breaks down long claims into simpler sub-claims, enhancing annotation and entailment prediction. For our experiments, we used the original WiCE claims instead of sub-claims to retain context. We further adapted WiCE's three-way entailment to a binary scheme by relabelling the partially supported claims as unsupported for aligning with the HoVer dataset. The dataset is split into 1,260 labeled training examples (460 Supported and 800 Not-Supported), 349 labeled development examples (115 Supported and 234 Not-Supported, and 358 unlabeled test examples.

Wikipedia corpus: For HoVer, we used the processed 2017 English Wikipedia dump from the HotPotQA website, containing 5,486,211 articles. For WiCE, we used the latest available English Wikipedia dump from January 1, 2024, which includes 6,777,401 articles. Both dumps were processed using the HotPotQA fork[3] of the Wikiextractor tool[4] to format the data into a structured folder system containing Bzip2 files.

2024 Presidential debate: We employ the fact-checking pipeline for the task of live fact-checking of the 2024 US presidential debate. We identified 281 claims in the debate with corresponding labels. We employ the 2024 Wikipedia dump as knowledge source for fact-checking the claims.

4.2 Experiment Setup

Computational resources. The experiments were conducted on a dedicated server running Arch Linux, equipped with a 16-core 2nd Gen AMD EPYC™ 7302 processor, 256 GB RAM and two NVIDIA GeForce RTX 3090 GPUs.

Tools, Models and Hyperparameters. The Wikipedia dumps are processed using WikiExtractor, JobLib (parallelization) and SpaCy's 'en_core_web_lg' model[5] for sentence splitting instead of StanfordCoreNLP due to performance issues (processing large volumes of text in parallel). For **sparse retrieval**, we utilize BM25 from Elasticsearch[6]. For **dense retrieval**, we employ *'all-MiniLM-L6-v2'* model from Sentence Transformers[7] to encode the query and all the documents. We employ ANN search using a flat index structure over the document embeddings leveraging FAISS library[8]. For index compression JPQ, configured with *K=256* codewords and *M=96* codebooks, utilizes Roberta-based dual-encoders with OPQ for linear transformation and PQ for compression. Training involved AdamW optimizer with batch size 32, LambdaRank for pair-wise loss, and optimization settings specific to query and PQ learning rates. For claim detection, a pre-trained BERT model[9] fine-tuned on the ClaimBuster dataset is used. Other pre-trained BERT-base uncased models are employed in the pipeline, fine-tuned with a batch size of 16, a learning rate of 3e-5, and varying epochs (3 for Sentence Selection, 5 for other stages).

Performance and Efficiency Evaluation: To evaluate end-end task performance, we employ weighted F1 score. For retrieval latency, we'll measure CPU and GPU implementations of FAISS and CPU for BM25. Metrics such as index size, creation time, retrieval time, total runtime, dataset size, and disk writes will be monitored using tools like psutil[10] and nvidia-ml-py3[11]. Inference latency will be measured in milliseconds.

[3] https://github.com/qipeng/wikiextractor.
[4] https://github.com/attardi/wikiextractor.
[5] https://spacy.io/models/en.
[6] https://elasticsearch-py.readthedocs.io/en/v8.12.1/.
[7] https://www.sbert.net/docs/pretrained_models.html.
[8] https://github.com/facebookresearch/faiss.
[9] https://huggingface.co/Nithiwat/bert-base_claimbuster.
[10] https://psutil.readthedocs.io/en/latest/.
[11] https://github.com/nicolargo/nvidia-ml-py3.

5 Results

5.1 Effectiveness of Indexing Succinct Facts to Improve Information Retrieval Efficiency

To answer **RQ1**, we measure the memory and computational costs of fact-checking using full-Wikipedia compared to the pruned version proposed in this work. We first measure the index size on disk, measuring the raw JSON file size containing the article titles and texts, for each experiment setting. In Fig. 1, we observe a significant reduction in disk space usage with HoVer having a reduction ranging from **44–55%**, and WiCE from **44–57%**. Additionally, the number of sentences stored in the index also decreases, with HoVer showing a reduction from 52–61% and WiCE 52–59%, indicating that at least half of the sentences are not helpful in claim verification.

Table 1. Comparison sizes for the corpora per experiment setting, consisting of English Wikipedia articles 2017 (HoVer) and 2024 (WiCE). Reduction is measured compared to the Full-Wiki data setting. (▼%) denotes a reduction in corpus size and number of sentence compared to Full-Wiki setting.

Method	Disk Size	Size reduction	#Sentences	% decrease
HoVer				
Full-Wiki	11.28 GiB	-	94,914,378	-
Fact Extraction	6.19 GiB	(▼45%)	45,894,704	(▼52%)
Citation Extraction	**5.07 GiB**	(▼**55**%)	**36,886,889**	(▼**61**%)
Fusion	5.45 GiB	(▼52%)	39,842,574	(▼58%)
WiCE				
Full-Wiki	15.28 GiB	-	126,533,841	-
Fact Extraction	8.56 GiB	(▼44%)	61,040,380	(▼52%)
Citation Extraction	**6.56 GiB**	(▼**57**%)	**51,735,961**	(▼**59**%)
Fusion	6.85 GiB	(▼55%)	54,070,295	(▼57%)

Following the reduction in disk size, a notable improvement in retrieval latency is evident, as demonstrated in Table 2. Regarding document retrieval latency (which encompasses both column values), there's an observed speedup ranging from approximately 1.5x (334 ms) to 1.6x (316 ms) compared to the original experimental setting for HoVer (495 ms). Similarly, in WiCE experiments, we witness a comparable speedup rate ranging from 1.4x (446 ms) to 1.6x (399 ms) compared to the original experimental setting (636 ms). This observation suggests that while the reduced text size contributes to efficient retrieval, it could further be improved.

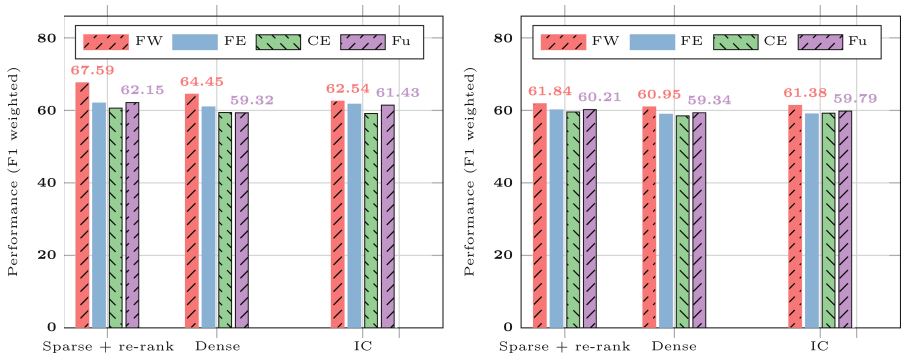
Insight 1: *Extraction of succinct facts reduces storage requirements and improves latency for Sparse retrieval while only leading to a minor loss in task performance.*

Table 2. Retrieval and total latency for Sparse retrieval with Re-ranking. Speedup is compared with respect to the total latency of the Full-Wiki setup.

Methods	Retrieval	Total Latency	Speedup
HoVer			
Full-Wiki	426 ms	659 ms	-
Fact Extraction	257 ms	338 ms	1.9x
Citation Extraction	246 ms	327 ms	(▲**2.0x**)
Fusion	265 ms	345 ms	1.9x
WiCE			
Full-Wiki	559 ms	831 ms	-
Fact Extraction	372 ms	468 ms	1.8x
Citation Extraction	330 ms	419 ms	(▲**2.0x**)
Fusion	347 ms	436 ms	1.9x

5.2 Effectiveness of Pruned Knowledge Sources on Overall Pipeline Efficiency and Downstream Fact-Checking Performance?

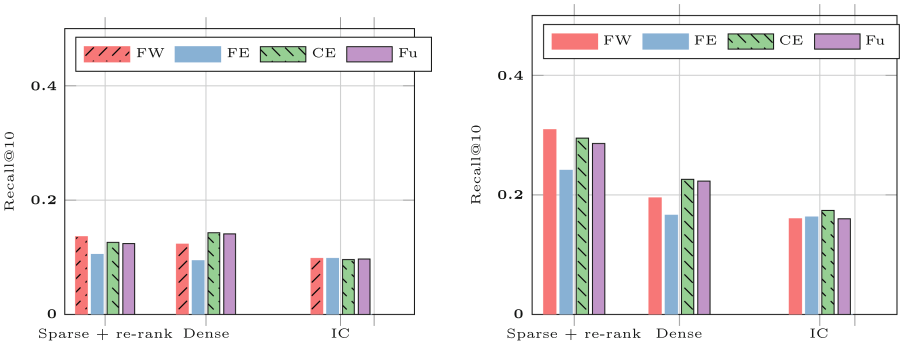
To answer **RQ2**, we now shift focus to analyzing the inference time throughout the entire pipeline instead of solely the retrieval part. Extraction of just supporting facts not only has a improvement in the retrieval stage but also on the overall inference latency across the pipeline. For HoVer this being a 1.9-2.0x speedup and 1.8-2.0x for WiCE experiments. This improvement can be attributed to not only faster retrieval times but also the elimination of the Sentence Retrieval stage, which previously imposed significant latency overhead.



(a) HoVer (b) Wice

Fig. 2. HoVer and WiCe task performance (FW- Full-Wiki, FE - Fact Extraction, IC-Index Compression, CE - Citation Extraction, Fu - Fusion)

The evaluation of Sparse and Dense Retrieval setups in HoVer and WiCE experiments reveals that Sparse Retrieval, particularly fact extraction (FE) and Fusion approaches, maintains performance closest to the Full-Wiki setup as measured by weighted F1 in Fig. 2, while citation extraction has a larger drop in



(a) WiCE (nDCG@10) (b) WiCE (Recall@10)

Fig. 3. Retrieval performance comparison

performance. Most notably, the Fusion method compared to the other methods has relatively high scores, underscoring the importance of combining supporting facts extraction methods for optimal results. We also report retrieval performance for WiCE Fig. 3 using measures nDCG@10 and Recall@10 using annotated documents provided for WiCE. We observe trends similar to overall task performance demonstrating that efficient retrieval approaches explored do not negatively impact task performance to a significant extent.

Insight 2: *We find that extracting supporting facts improves efficiency across the entire pipeline, with Sparse setups achieving up to 2.0x speedups with only a minimal performance decline.*

Table 3. Latency and speedup measurements for Index compression setup. Speedup is compared with respect to the total latency of Sparse-retrieval + Re-rank (S+R) pipeline with the Full-Wiki setup. (S+R) runs on both CPU and GPU, sparse retrieval running on CPU and rest of components running on GPU

Method	Total Latency		Speedup	
	CPU	GPU	CPU	GPU
HoVer				
Full-Wiki (S+R)	659 ms		-	-
Full-Wiki	214 ms	174 ms	3.1x	3.8x
Fact Extraction	60 ms	21 ms	11.0x	31.4x
Citation Extraction	**51 ms**	**20 ms**	(▲**12.9x**)	(▲**33.0x**)
Fusion	63 ms	24 ms	10.5x	27.5x
WiCE				
Full-Wiki (S+R)	831 ms		-	-
Full-Wiki	292 ms	238 ms	2.8x	3.5x
Fact Extraction	103 ms	48 ms	8.1x	17.3x
Citation Extraction	98 ms	46 ms	8.5x	18.1x
Fusion	**98 ms**	**46 ms**	(▲**8.5x**)	(▲**18.1x**)

5.3 Effectiveness of Index Compression on Enhancing the Efficiency of Dense Retrieval and Fact-Checking Systems?

To answer **RQ3**, we make use of index compression to further improve upon Dense Retrieval setups, not only with respect to memory requirements but also enhancing total inference latency compared to the sparse retrieve + re-rank setups in classical pipelines. The index sizes of Wikipedia collection for standard dense retrieval are 7.51 GiB for HoVer and 9.70 GiB for WiCE. Using the JPQ index compression model with M=96 subvectors, we significantly reduced the storage space for vector embeddings from 1.5 KiB to 104.12 B. This reduced the HoVer index size to 544.89 MiB and the WiCE index to **672.95 MiB**, achieving a **93% reduction (14.4:1 compression ratio)**. Further reducing subvectors could decrease the index size but may impact performance.

The utilization of JPQ index compression leads to significant reductions in retrieval latency compared to dense Retrieval and sparse retrieval, as demonstrated in Table 3. CPU retrieval gains notable speedups of approximately 10.0x for HoVer experiments and 7.0x for WiCE experiments, while GPU retrieval shows about 2.0x and 0.8x speedups, respectively. These improvements are attributed to learned compression in JPQ, enhancing computational efficiency. Significant improvements are also observed when examining the inference latency of the whole pipeline. The CPU-based approaches shows impressive speedups (upto **12.9x** for HoVer and **8.5x** for WiCE), and GPU-based approaches achieve even higher gains (**33.0x** for HoVer and **18.1x** for WiCE).

Surprisingly, in our experiments we observe that JPQ yields better results than standard Dense Retrieval as shown in Fig. 2. This is particularly due to joint training of the query encoder and index compression. In addition, JPQ employs end-end negative sampling, which further improves retrieval performance despite significant compression of embeddings.

Insight 3: *We find that index compression reduces index size by **93%** resulting in speedups for CPU-based setups up to 10x and GPU-based setups more than 20x compared to classical fact-checking pipeline.*

5.4 Discussion of Live Fact-Checking Results

We employ the pruned index (2024 Wiki collection) using our Fusion approach followed by compression of the index for live Fact-checking of 2024 presidential debate. The pipeline comprises a dense retrieval using compressed index followed by claim verification. We use the query encoder and NLI models trained on HoVer for this application. We compare this approach to also the classical sparse-retrieval+re-rank fact-checking pipeline over the Full-Wiki collection (without pruning). The task performance is shown in Fig. 4 and the corresponding pipeline efficiency is shown in Table 4. We observe that the pruned collection coupled with retrieval using index compression leads to impressive speedups (**3.4x**) over classical pipeline over the full collection without significant drop in task performance (Fig. 4). The results demonstrate that efficient retrieval is critical for live fact-checking. Our experiments demonstrate that our approach

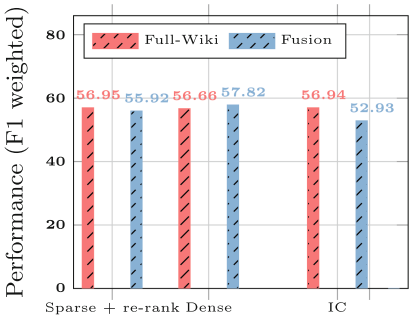


Fig. 4. Live fact-checking performance across different corpus setups

Table 4. Latency Comparisons for Live Fact-checking (in milliseconds (ms))

	Retrieval		Total Latency		Speedup	
	CPU	GPU	CPU	GPU	CPU	GPU
Sparse + Re-ranking						
Full-Wiki	463	-	695	-	-	
Fusion	274	-	479	-	1.5x	
Dense Retrieval						
Full-Wiki	433	32	553	152	1.3x	4.6x
Fusion	412	32	511	131	1.4x	5.3x
Index Compression						
Full-Wiki	100	50	228	178	3.0x	3.9x
Fusion (ours)	**89**	**43**	**203**	157	(▲**3.4x**)	4.4x

for efficient retrieval provides significant speedups on CPUs further making the technology accessible even in low-resource scenarios which has significant implications in aiding detection of misinformation and disinformation at scale.

6 Conclusion

In this work, we assess how indexing potentially relevant facts from large data collections can improve fact-checking pipelines. Our experiments show that indexing only factual content coupled with index compression significantly improves pipeline efficiency with minimal performance loss, making it feasible for lower-end, CPU-focused machines and applications like live fact-checking. Future work will expand this approach by testing various settings for indexing,

retrieval, and compression, using diverse claim datasets and larger corpora to strengthen the robustness and real-world applicability of fact-checking systems.

Acknowledgements. This work is partly funded by the Research Council of Norway project EXPLAIN (grant no: 337133). We acknowledge valuable contributions from Factiverse AI team: Tobias Tykvart for Frontend, Henrik Vatndal and others for manual analysis (Maria Amelie, Gaute Kokkvol, Sean Jacob, Christina Monets and Mari Holand).

References

1. Aly, R., Vlachos, A.: Natural logic-guided autoregressive multi-hop document retrieval for fact verification. In: Goldberg, Y., Kozareva, Z., Zhang, Y., (eds.) Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 6123–6135, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics (2022). https://doi.org/10.18653/v1/2022.emnlp-main.411
2. Baranchuk, D., Babenko, A., Malkov, Y.: Revisiting the inverted indices for billion-scale approximate nearest neighbors (2018)
3. Bondarenko, Y., Nagel, M., Blankevoort, T.: Understanding and overcoming the challenges of efficient transformer quantization (2021)
4. Bruch, S., Nardini, F.M., Rulli, C., Venturini, R.: Efficient inverted indexes for approximate retrieval over learned sparse representations. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, pp. 152–162. Association for Computing Machinery, New York, NY, USA (2024). ISBN 9798400704314. https://doi.org/10.1145/3626772.3657769
5. Ge, T., He, K., Ke, Q., Sun, J.: Optimized product quantization. IEEE Trans. Pattern Anal. Mach. Intell. **36**(4), 744–755 (2014). https://doi.org/10.1109/TPAMI.2013.240
6. Guo, J., Cai, Y., Fan, Y., Sun, F., Zhang, R., Cheng, X.: Semantic models for the first-stage retrieval: a comprehensive review. ACM Trans. Inf. Syst. **40**(4), 1–42 (2022). https://doi.org/10.1145/3486250
7. Guo, Z., Schlichtkrull, M., Vlachos, A.: A survey on automated fact-checking. Trans. Assoc. Comput. Linguist. **10**, 178–206 (2022). https://aclanthology.org/2022.tacl-1.11
8. Han, Y., Liu, C., Wang, P.: A comprehensive survey on vector database: storage and retrieval technique, challenge (2023)
9. Hassan, N., Li, C., Tremayne, M.: Detecting check-worthy factual claims in presidential debates. In: CIKM 2015. Association for Computing Machinery (2015)
10. Hofstätter, S., Lin, S.-C., Yang, J.-H., Lin, J., Hanbury, A.: Efficiently teaching an effective dense retriever with balanced topic aware sampling. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2021, pp. 113–122. Association for Computing Machinery, New York, NY, USA (2021). ISBN 9781450380379. https://doi.org/10.1145/3404835.3462891
11. Hu, X., Hong, Z., Guo, Z., Wen, L., Yu, P.: Read it twice: towards faithfully interpretable fact verification by revisiting evidence. In: Proceedings of the 46th

International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, pp. 2319–2323. Association for Computing Machinery, New York, NY, USA (2023). ISBN 9781450394086. https://doi.org/10.1145/3539618.3592049
12. Jiang, Y., Bordia, S., Zhong, Z., Dognin, C., Singh, M., Bansal, M.: Hover: a dataset for many-hop fact extraction and claim verification (2020)
13. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 117–128 (2011). https://doi.org/10.1109/TPAMI.2010.57
14. Kamoi, R., Goyal, T., Rodriguez, J., Durrett, G.: WiCE: real-world entailment for claims in Wikipedia. In: Bouamor, H., Pino, J., Bali, K., (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 7561–7583, Singapore, December 2023. Association for Computational Linguistics (2023). https://aclanthology.org/2023.emnlp-main.470
15. Karpukhin, V., et al.: Dense passage retrieval for open-domain question answering (2020)
16. Lazarski, E., Al-Khassaweneh, M., Howard, C.: Using NLP for fact checking: a survey. Designs **5**(3), 42 (2021). ISSN 2411-9660. https://doi.org/10.3390/designs5030042
17. Leonhardt, J., Müller, H., Rudra, K., Khosla, M., Anand, A., Anand, A.: Efficient neural ranking using forward indexes and lightweight encoders (2023). https://arxiv.org/abs/2311.01263
18. Li, Y., Xie, Y.: Is a picture worth a thousand words? An empirical study of image content and social media engagement. J. Market. Res. **57**, 1–9 (2020)
19. Luan, Y., Eisenstein, J., Toutanova, K., Collins, M.: Sparse, dense, and attentional representations for text retrieval. Trans. Assoc. Comput. Linguist. **9**, 329–345 (2021). https://aclanthology.org/2021.tacl-1.20
20. Nakov, P., et al.: Automated fact-checking for assisting human fact-checkers (2021). https://arxiv.org/abs/2103.07769
21. Newman, E., Garry, M., Bernstein, D., Kantner, J., Lindsay, D.: Nonprobative photographs (or words) inflate truthiness. Psychon. Bull. Rev. **19**, 969–974 (2012)
22. Pan, L., et al.: Fact-checking complex claims with program-guided reasoning. In Rogers, A., Boyd-Graber, J., Okazaki, N., (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 6981–7004. Association for Computational Linguistics, Toronto, Canada, July 2023. https://doi.org/10.18653/v1/2023.acl-long.386
23. Petroni, F., et al.: Improving Wikipedia verifiability with AI (2022)
24. Samarinas, C., Hsu, W., Lee, M.L.: Improving evidence retrieval for automated explainable fact-checking. In: Sil, A., Lin, X.V., (eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations, pp. 84–91, June 2021. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-demos.10
25. Schlichtkrull, M.S., Karpukhin, V., Oguz, B., Lewis, M., Yih, W., Riedel, S.: Joint verification and reranking for open fact checking over tables. In: Zong, C., Xia, F., Li, W., Navigli, R., (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 6787–6799. Association for Computational Linguistics, August 2021. https://doi.org/10.18653/v1/2021.acl-long.529

26. Schuster, T., Fisch, A., Barzilay, R.: Get your vitamin c! Robust fact verification with contrastive evidence (2021). https://arxiv.org/abs/2103.08541
27. Thorne, J., Vlachos, A.: Automated fact checking: Task formulations, methods and future directions (2018)
28. Thorne, J., Vlachos, A., Christodoulopoulos, C., Mittal, A.: Fever: a large-scale dataset for fact extraction and verification (2018)
29. Wei, Z., Xu, X., Wang, C., Liu, Z., Xin, P., Zhang, W.: An index construction and similarity retrieval method based on sentence-BERT. In: 2022 7th International Conference on Image, Vision and Computing (ICIVC), pp. 934–938 (2022). https://doi.org/10.1109/ICIVC55077.2022.9886134
30. Yin, W., Roth, D.: TwoWingOS: a two-wing optimization strategy for evidential claim verification. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J., (eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 105–114, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1010
31. Zhan, J., Mao, J., Liu, Y., Zhang, M., Ma, S.: RepBERT: Contextualized text embeddings for first-stage retrieval (2020). https://arxiv.org/abs/2006.15498
32. Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Jointly optimizing query encoder and product quantization to improve retrieval performance (2021)
33. Zhao, W.X., Liu, J., Ren, R., Wen, J.-R.: Dense text retrieval based on pretrained language models: a survey (2022)
34. Zhao, X., Tian, Y., Huang, K., Zheng, B., Zhou, X.: Towards efficient index construction and approximate nearest neighbor search in high-dimensional spaces. Proc. VLDB Endow. **16**(8), 1979–1991 (2023). https://doi.org/10.14778/3594512.3594527
35. Zheng, L., Li, C., Zhang, X., Shang, Y.-M., Huang, F., Jia, H.: Evidence retrieval is almost all you need for fact verification. In: Ku, L.-W., Martins, A., Srikumar, V., (eds.) Findings of the Association for Computational Linguistics: ACL 2024, pp. 9274–9281, Bangkok, Thailand, August 2024. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-acl.551
36. Zhu, X., Li, Y., Liu, J., Ma, C., Wang, W.: A survey on model compression for large language models (2023)