SMARTMATCH: Real-Time Semantic Retrieval for Translation Memory Systems

Ernesto L. Estevanell-Valladares[1,2], **Salima Lamsiyah**[3], **Alicia Picazo-Izquierdo**[1], **Tharindu Ranasinghe**[4], **Ruslan Mitkov**[1], **Rafael Muñoz**[1]
[1]University of Alicante, Spain [2]University of Havana, Cuba
[3]University of Luxembourg, Luxembourg [4]Lancaster University, UK
ernesto.estevanell@ua.es

Abstract

Translation Memory (TM) systems are core components of commercial computer-aided translation (CAT) tools. However, traditional fuzzy matching methods often fail to retrieve semantically relevant content when surface similarity is low. We introduce SMARTMATCH[1], an open-source interactive demo and evaluation toolkit for TM retrieval that connects modern sentence encoders (including LLM-derived representations) and strong lexical/fuzzy baselines with a vector database, and exposes the end-to-end retrieval pipeline through a web-based UI for qualitative inspection and preference logging. The demo allows users to (i) enter a query segment, (ii) switch retrieval backends and embedding models, (iii) inspect top-k retrieved matches with similarity scores and qualitative cues, and (iv) observe end-to-end latency in real time. We provide a reproducible benchmark on multilingual TM data, reporting retrieval quality using reference-based MT metrics (COMET, BERTScore, METEOR, chrF) together with coverage and latency/throughput trade-offs relevant to real-time CAT workflows. On DGT-TM, encoder-based retrieval achieves full coverage (100%) with millisecond-level latency (p50/p90 $\leq$ 6–20 ms) and attains the strongest semantic-quality scores on the shared query set (e.g., BERTScore up to 0.91 at k=10), while BM25 remains a strong lightweight lexical baseline with very low latency. SMARTMATCH targets CAT researchers and tool builders and bridges recent advances in sentence encoders with the real-time constraints of translation memory retrieval.

1 Introduction

Translation Memory (TM) systems are foundational components of computer-aided translation tools (CAT), widely adopted in the translation industry to improve translator productivity and consistency (Mitkov, 2022; Reinke, 2018). By storing previously translated units (TU) and retrieving them during new translation tasks, TM systems enable TU reuse, remove redundant human effort, and significantly reduce both translation time and costs while maintaining quality (Pașcalau et al., 2025). Recent surveys[2] show that 88% of full-time professional translators use at least one CAT tool, and translation memory usage more than doubled from 2023 to 2024, demonstrating that TM systems remain indispensable to the translation industry despite advances in neural machine translation.

Traditional TM retrieval relies on surface-level fuzzy matching, which struggles when lexical overlap is low (Baquero and Mitkov, 2017; Gupta et al., 2016). To mitigate this, prior work has introduced enhancements such as paraphrasing (Gupta and Orasan, 2014) and clause splitting (Timonera and Mitkov, 2015). Despite major advances in semantic retrieval driven by large language models (LLMs) (Muennighoff et al., 2023; Ranasinghe et al., 2025), TM systems have not fully adopted these methods due to concerns about latency and scalability in real-time CAT workflows. While a few studies have explored sentence encoders for TM retrieval (Zhang et al., 2020; Ranasinghe et al., 2020), these efforts have largely relied on earlier encoder architectures. Modern LLM-based sentence encoders, which demonstrate superior semantic understanding, remain unexplored in TM systems, which leaves a significant gap between state-of-the-art retrieval models and TMs.

In this paper, we describe SMARTMATCH, an open-source system for LLM-based semantic retrieval in translation memory which we developed. Unlike traditional fuzzy matching, SMARTMATCH

[1]https://github.com/EEstevanell/SmartMatch

[2]https://lokalise.com/library/data-reports/localization-trends-2025/

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 3: System Demonstrations, pages 527–536
March 24-29, 2026 ©2026 Association for Computational Linguistics

uses state-of-the-art sentence encoders within a unified pipeline that also includes strong lexical baselines (Okapi/Pensieve and BM25). We provide a reproducible benchmark on DGT-TM, showing that encoder-based retrieval achieves 100% coverage and state-of-the-art semantic quality with millisecond-level latency, and we release an interactive web UI that exposes these quality–latency trade-offs for real-time CAT scenarios.

The **main contributions** of this paper are the following:

(*i*) We **introduce** SMARTMATCH, an open-source solution for LLM-based semantic retrieval in translation memory systems incorporating state-of-the-art encoder models and vector databases.

(*ii*) We **empirically evaluate** a diverse range of embedding models for TM retrieval, comparing popular open-source encoders against industry-relevant baselines (Okapi Pensieve and BM25). On the DGT-TM benchmark, encoder-based retrieval achieves 100% coverage and attains the best COMET and BERTScore results at ranks $k \in \{5, 10\}$ on the shared query set, while BM25 remains a very strong and extremely fast lexical baseline (Table 2).

(*iii*) We **analyse** latency across different architectures, demonstrating millisecond end-to-end retrieval for encoder-based setups (p50/p90 $\leq$ 6–20 ms) and contrasting this with the heavy-tail behavior of Okapi-based retrieval (p90 $\approx$ 1.1 s).

(*iv*) We **release** an interactive web-based UI that allows users to enter text and retrieve the best match from the TM, switch models on the fly, and directly observe quality–latency trade-offs.

2 Related Work

Research on Translation Memory (TM) has evolved from early rule-based systems (Sato and Nagao, 1990; Schjoldager and Christensen, 2010) to commercial deployments (Reinke, 2018; Mitkov, 2022) supported by large multilingual resources such as DGT-TM (Steinberger et al., 2012). Prior work examined TM's practical impact on translation quality and professional workflows (Jiménez-Crespo, 2010; Baquero and Mitkov, 2017), alongside its ethical and legal dimensions (Park, 2024). To improve TM reliability, shared tasks and unsupervised cleaning methods have addressed redundancy and noise (Barbu et al., 2016; Sabet et al., 2016; Negri et al., 2017; Wolff, 2016).

With the rise of machine translation, significant efforts have focused on integrating TM into statistical and neural MT frameworks. Initial work on SMT explored the use of TM features during decoding (Wang et al., 2013; Li et al., 2014), which later evolved into neural approaches employing gating, contrastive learning, and cross-lingual retrieval (Cao and Xiong, 2018; Cheng et al., 2022; Tamura et al., 2023). Additionally, studies have shown that TM-enhanced outputs can improve translator productivity and reduce post-editing time (Sánchez-Gijón et al., 2019; Green et al., 2014). To further improve matching quality, methods such as paraphrasing (Gupta et al., 2016), clause splitting (Timonera and Mitkov, 2015), and context-aware retrieval (He et al., 2019) have been proposed. These approaches aim to increase the utility of TM suggestions in real-time translation environments.

Recent research has also explored embedding-based retrieval as an alternative to traditional fuzzy matching. Ranasinghe et al. (2020, 2021) reported early gains using sentence encoders, though their study was limited in model scope and scalability. In parallel, TM integration with large language models has gained increasing attention, which demonstrates benefits from similarity-based prompting, retrieval augmentation, and dynamic memory mechanisms (Mu et al., 2023; Qian and Kong, 2024; Xu et al., 2023). Our work extends this line by benchmarking a broader range of modern, publicly available models under realistic latency constraints.

3 SMARTMATCH System Overview

SMARTMATCH is a research-orientated, end-to-end toolkit and interactive demo for comparing TM retrieval strategies. It offers the full retrieval workflow, including experiment setup, indexing, querying, and inspection, through a web-based user interface, while remaining easy to deploy and reproduce. The system is designed to (i) support fair comparisons across retrieval paradigms (fuzzy, lexical, dense), (ii) satisfy low-latency requirements typical of CAT workflows, and (iii) enable rapid qualitative assessment via interactive exploration and feedback logging. SMARTMATCH is implemented in Python 3.11+ and integrates Weaviate for indexing/retrieval, LangChain for embedding/model orchestration, and Typer for experiment configuration and command-line execution.

3.1 Architecture at a Glance

The system is organised into two phases:

Offline indexing. Given a TM collection $\mathcal{D} = \{(s_i, t_i)\}_{i=1}^{N}$, SMARTMATCH prepares three retrieval backends: (i) a fuzzy-matching service (Okapi/Pensieve) over raw segments, (ii) a lexical index (BM25) for sparse retrieval, and (iii) a dense vector index for semantic retrieval. Users first select an experiment setting that samples a TM subset for indexing (e.g., a random training split) and specify indexing parameters such as the embedding model and batch size. For dense retrieval, SMARTMATCH iterates over the chosen TM subset, computes embeddings $E(s_i)$ for all source segments, and writes them into a dedicated Weaviate collection indexed with an HNSW ANN structure. In parallel, Weaviate maintains an inverted index to support BM25 retrieval over the same segments, ensuring that lexical and dense retrieval operate over identical content and can be compared under consistent database conditions. For fuzzy matching, SMARTMATCH triggers a Pensieve preparation step that generates the memory files required for edit-distance retrieval. Embeddings are computed by the selected encoder in the embedding service and stored in Weaviate; Weaviate is used as a common retrieval substrate (ANN + BM25) to enable controlled comparisons under consistent database conditions.

Index readiness checks. To prevent ambiguous "no-results" behaviour, SMARTMATCH enforces index readiness: if the selected Weaviate collection has not been seeded (e.g., empty collection) or Okapi/Pensieve memory files are missing, the retrieval interface 'declines' to run. This makes it explicit when indexing has not completed and avoids silent failures.

Online retrieval. At query time, a user submits a source segment q via the UI. The query can be selected from the held-out test split or entered as custom text. The request is routed to one of the enabled backends (fuzzy, BM25, or dense). For dense retrieval, the selected encoder produces $E(q)$ and Weaviate returns the top-k nearest neighbours under cosine similarity. For lexical retrieval, Weaviate returns the top-k BM25 matches. For fuzzy retrieval, the Okapi/Pensieve service returns top-k segments scored by normalised edit-distance similarity. Results are converted into a unified response schema consumed by the UI and include end-to-end latency measured in milliseconds.

3.2 Retrieval Backends

SMARTMATCH provides three interchangeable retrieval modes:

Fuzzy matching (Okapi/Pensieve). To reflect legacy CAT deployments, we integrate a Java-based Okapi/Pensieve service that computes normalised edit-distance scores over the TM. This backend serves as a traditional production-like baseline consistent with common CAT tool practice.

Lexical retrieval (BM25 in Weaviate). We use Weaviate's native BM25 implementation[3] for sparse retrieval, benefiting from the same storage layer used by dense indexing. This provides a strong keyword-based baseline and enables direct latency comparison under consistent database settings.

Dense retrieval (encoders + HNSW in Weaviate). For each encoder listed in §4.1, SMARTMATCH indexes pre-computed vectors and retrieves candidates via HNSW-based ANN search. We score candidates using cosine similarity and return the top-k matches together with similarity scores.

3.3 Unified API and Result Schema

To support interactive comparison across strategies, all backends return results in a common schema: $\{\texttt{query}, \texttt{backend}, \texttt{model}, k, [(s_i, t_i, score)]\}$. This enables the UI to render comparable top-k lists, sort by score, and optionally display metadata (e.g., segment length and language). The API also records end-to-end latency for each request (from query submission to response), enabling real-time inspection of quality–latency trade-offs and aggregate reporting (e.g., p50/p90 latency in experiments).

3.4 Interactive Demo Interface and Feedback

The web UI is designed for rapid qualitative inspection and live model comparison. Users can: (i) input a query (test sample or custom sentence), (ii) select a retrieval backend and (for dense retrieval) an embedding model, (iii) choose k (e.g., $k \in \{1, 5, 10\}$), and (iv) view retrieved source segments alongside their target translations. The interface reports per-query latency in milliseconds and

[3] https://huggingface.co/blog/xhluca/bm25s

supports view customisation so users can hide non-essential fields. Beyond exploration, a dedicated "Feedback Charts" page turns SMARTMATCH into an interactive evaluation tool: for each query, users can compare the top suggestions from multiple backends and select their preferred output, taking match quality and latency into account. Each decision is logged to a JSON feedback file, and the chart visualisation updates live as new preferences are recorded.

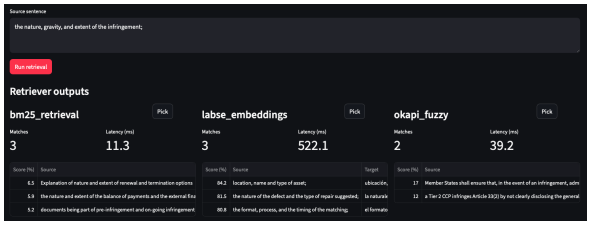

Figure 1: TM Retrieval tool

3.5 Deployment

SMARTMATCH is intended to be deployed as a lightweight set of services: (1) a Weaviate instance for BM25 and vector search, (2) an embedding service that exposes encoder inference for multiple sentence encoders, (3) an Okapi/Pensieve wrapper for fuzzy matching, and (4) a web UI for querying, comparison, and feedback collection. This modular design mirrors typical CAT pipeline integration points and allows users to enable only the components required for their workflow (e.g., lexical+dense retrieval without Okapi) while preserving a unified interface for experimentation.

4 Experimental Setup

This section summarises the experimental configuration used to benchmark the retrieval backends showcased in the SMARTMATCH demo. It is worth mentioning that all models are evaluated using *zero-shot* (no task-specific fine-tuning) and without cross-encoder re-ranking.

4.1 Retrieval Models

We evaluate 10 neural retrieval architectures, grouped into two categories.

Bi-encoder sentence encoders. We use encoder-style models that map a segment to a fixed-dimensional embedding and support ANN search: Multilingual E5 (*intfloat/multilingual-e5-base*, d=768), BGE-M3 (*BAAI/bge-m3*, d=1024), LaBSE (*sentence-transformers/LaBSE*, d=768),

Retrieval	Text	Latency
source text	Description of accounting policy that requires external consulting services	-
BM25	Description of accounting policy for business combinations [text block]	5.1
LaBSE	The description of the entity's material accounting policy information for hedging.	66.0
Okapi/Pensieve	No match found	1.2

Table 1: Example retrieval outputs from BM25, LaBSE, and Okapi/Pensieve for a low-lexical-overlap query.

LASER (*Facebook/LASER*, d=1024), M3E-Base (*moka-ai/m3e-base*, d=768), and three SBERT-family checkpoints: MPNet (*all-mpnet-base-v2*, d=768), DistilRoBERTa (*all-distilroberta-v1*, d=768), and MiniLM (*all-MiniLM-L6-v2*, d=384).

Generative LLM architectures. To assess whether hidden states from generative models can serve as retrieval representations, we evaluate: Llama-3.2 (*meta-llama/Llama-3.2-1B*, d=2048), using the final-layer hidden state of the last token as the segment representation, and T5 (*t5-base*, d=768), using the encoder output as the segment representation.

Non-neural baselines. We include two industry-relevant baselines exposed in the demo UI: (i) fuzzy matching using normalised Levenshtein similarity via Okapi/Pensieve, and (ii) lexical retrieval using BM25 implemented in Weaviate.

4.2 Datasets

We run experiments on the DGT Translation Memory released by the European Commission (Directorate-General for Translation) and the Joint Research Centre. The dataset contains professionally produced translation units across 22 official EU languages and multiple language-pair combinations, including domain-specific texts.[4] While we selected the English-Spanish pairs for this test, our system can include any language pair. Within this TM, we used the following train-test split: Volume 1-4 were used for training purposes, and Volume 5 for testing.

4.3 Indexing and Retrieval Configuration

For each experimental run, we create a dedicated TM index by sampling a *training* subset used for indexing and reserving a held-out *test* subset for

[4] https://joint-research-centre.ec.europa.eu/language-technology-resources/dgt-translation-memory_en

querying. Dense retrieval follows the standard bi-encoder pipeline: we pre-compute embeddings for all indexed source segments and store them in a Weaviate collection, which is configured with an HNSW ANN index for cosine similarity search. The same collection stores the raw text fields used by Weaviate's BM25 inverted index, enabling lexical retrieval over identical content. For the fuzzy baseline, Pensieve memory files are generated from the indexed segments prior to querying.

4.4 Hardware

All experiments were conducted on a workstation with an Intel(R) Core(TM) i7-14700 CPU (28 cores, up to 5.4 GHz), 64 GB RAM, and an NVIDIA RTX 4090 (24 GB VRAM). This configuration provides stable latency measurements and sufficient resources for embedding computation and large-scale indexing.

4.5 Evaluation Metrics

We evaluate retrieval quality with four reference-based MT metrics: COMET (Rei et al., 2020), BERTScore (Zhang et al., 2019), METEOR (Banerjee and Lavie, 2005), and chrF (Popović, 2015). We additionally report **Coverage** (percentage of queries with at least one retrieved TU) and **Latency** (end-to-end retrieval time), summarised by p50 and p90 to capture typical and tail behaviour. We evaluate ranking at $k \in \{1, 5, 10\}$ using *best-in-k* (oracle) scoring, i.e., the maximum metric score among the top-k candidates, reflecting CAT usage where translators select the most helpful suggestion from a shortlist.

4.6 Statistical Significance Testing

To ensure robustness, we performed statistical testing on the intersection of queries for which all models retrieved at least one TU. We used the non-parametric Wilcoxon signed-rank test (Rosner et al., 2006) to compare the Pensieve baseline against neural models, and applied the Benjamini–Hochberg procedure (Benjamini and Hochberg, 1995) to control the false discovery rate across multiple comparisons ($\alpha = 0.05$). We also report Cliff's delta (δ) (Cliff, 1993) as a non-parametric effect size and treat $|\delta| \geq 0.474$ as non-negligible.

5 Results

Table 2 reports the retrieval performance of all evaluated models across four metrics (COMET, BERTScore, METEOR, chrF) and three ranks (1, 5, 10). We highlight models that are both statistically significantly better than the Okapi Pensieve baseline ($p < 0.05$) and demonstrate a large effect size ($|\delta| \geq 0.474$).

Two distinct performance views for each evaluation metric are presented in Table 2. The first column reports the global performance on the full query set, calculated without penalising for lower coverage. It is important to note that these global metrics tend to favour models with lower coverage (such as Okapi Pensieve and BM25), as they effectively evaluate only on a subset of more straightforward queries where high lexical overlap exists. The second column, marked with $\bigcap$, reports performance on the intersection set of queries where all models successfully retrieved at least one result (100% coverage), which eliminates selection bias and provides a more accurate representation of comparative retrieval quality for statistical significance testing.

As shown in Table 2, modern semantic retrieval models consistently outperform traditional baselines. It is worth noting that LaBSE achieves the highest raw scores across all metrics and ranks, demonstrating its robustness as a general-purpose bitext retriever. Statistically, LaBSE, LASER, and T5 base achieve significant improvements with large effect sizes ($|\delta| \geq 0.474$) over the Pensieve baseline, specifically in COMET and BERTScore at ranks 5 and 10.

This performance divergence underscores a fundamental shift in retrieval paradigms. Vector-based models excel at capturing semantic similarity even with low lexical overlap, leading to higher COMET and BERTScore values. Conversely, traditional baselines like Pensieve and BM25 fare very well – for surface-form matches, maintaining strong performance on n-gram based metrics like METEOR and chrF but failing to retrieve semantically equivalent yet lexically distinct segments. Table 1 illustrates a typical retrieval case, showing how BM25 and LaBSE retrieve semantically related segments while the fuzzy baseline fails due to low lexical overlap.

From a practical production perspective, these quality gains are complemented by superior operational characteristics. In our testing, semantic models achieved 100% coverage, significantly outperforming Okapi Pensieve (81%) and BM25 (85%), which fail to retrieve results when lexical overlap

Model	R.	COMET	⋂	BERTScore	⋂	METEOR	⋂	chrF	⋂	Lat. p50/p90	Cov.
	1	**0.83**	0.84	**0.86**	0.87	**0.72**	**0.74**	**0.72**	**0.74**		
Okapi Pensieve	5	0.84	0.85	0.87	0.88	0.75	0.77	**0.75**	0.77	44/1116 ms	81
	10	0.84	0.86	0.87	0.89	0.75	0.77	**0.75**	0.77		
	1	0.80	0.80	0.85	0.84	0.69	0.69	0.59	0.69		
BM25	5	**0.85**	0.85	**0.88**	0.88	**0.76**	0.76	0.66	0.76	**3**/13 ms	85
	10	**0.86**	0.86	**0.89**	0.89	**0.78**	0.77	0.68	0.77		
	1	0.79	0.85	0.81	0.88	0.59	0.73	0.61	**0.74**		
BGE-M3	5	0.82	0.87	0.84	**0.90**	0.64	0.77	0.66	0.77	16/17 ms	**100**
	10	0.82	**0.88**	0.84	0.90	0.66	0.78	0.66	0.78		
	1	0.76	0.83	0.80	0.87	0.57	0.71	0.58	0.71		
Multilingual E5	5	0.80	0.86	0.83	0.89	0.63	0.76	0.63	0.76	16/18 ms	**100**
	10	0.81	0.87	0.84	0.90	0.64	0.77	0.65	0.77		
	1	0.79	0.86	0.82	0.89	0.61	**0.74**	0.62	**0.74**		
LaBSE	5	0.82	**0.88**	0.84	**0.90**	0.65	**0.78**	0.66	**0.78**	10/12 ms	**100**
	10	0.83	**0.88**	0.85	**0.91**	0.67	**0.79**	0.67	**0.79**		
	1	0.79	0.85	0.82	0.89	0.61	0.74	0.62	**0.74**		
LASER	5	0.82	0.87	0.84	**0.90**	0.66	**0.78**	0.66	**0.78**	7/14 ms	**100**
	10	0.82	**0.88**	0.84	**0.91**	0.67	**0.79**	0.67	0.78		
	1	0.78	0.85	0.81	0.88	0.58	0.71	0.60	0.73		
Llama-3.2-1B	5	0.81	0.87	0.83	**0.90**	0.62	0.75	0.64	0.76	17/20 ms	**100**
	10	0.82	0.87	0.84	0.90	0.64	0.76	0.65	0.77		
	1	0.77	0.84	0.81	0.88	0.59	0.72	0.60	0.73		
M3E	5	0.81	0.87	0.83	**0.90**	0.63	0.77	0.65	0.77	10/12 ms	**100**
	10	0.81	0.87	0.84	0.90	0.65	0.78	0.66	0.78		
	1	0.77	0.84	0.81	0.88	0.58	0.72	0.60	0.73		
MiniLM-L6-v2	5	0.81	0.87	0.83	**0.90**	0.63	0.76	0.65	0.77	6/**8** ms	**100**
	10	0.82	**0.88**	0.84	0.90	0.65	0.78	0.66	0.78		
	1	0.78	0.84	0.81	0.87	0.58	0.72	0.60	0.73		
MPNet-base-v2	5	0.81	0.87	0.83	0.89	0.63	0.76	0.65	0.77	11/14 ms	**100**
	10	0.82	**0.88**	0.84	0.90	0.65	0.77	0.66	0.78		
	1	0.78	0.85	0.81	0.88	0.59	0.72	0.61	**0.74**		
DistilRoBERTa	5	0.81	0.87	0.83	**0.90**	0.63	0.76	0.65	0.77	7/10 ms	**100**
	10	0.82	**0.88**	0.84	0.90	0.65	0.77	0.66	0.78		
	1	0.80	0.86	0.82	0.89	0.60	0.73	0.62	**0.74**		
T5 base	5	0.82	**0.88**	0.84	**0.90**	0.65	0.77	0.66	**0.78**	9/11 ms	**100**
	10	0.83	**0.88**	0.85	**0.91**	0.66	0.78	0.67	0.78		

Table 2: Retrieval performance on the DGT-TM dataset. The column with the symbol ⋂ reports values in the intersection set of queries where all models successfully retrieved at least one translation unit. **Bold** indicates the best performance per metric and rank. Underline denotes statistically significant improvement over the Okapi Pensieve baseline ($p < 0.05$) with a large effect size ($|\delta| \geq 0.474$). Latency (p50/p90) is measured in milliseconds.

is insufficient.

Regarding efficiency, while BM25 remains the fastest (3ms p50), lightweight semantic models like MiniLM-L6-v2 (6ms) and LASER (7ms) offer competitive speeds suitable for real-time applications. Notably, Okapi Pensieve exhibits high tail latency (1116ms p90), likely due to its reliance on complex fuzzy matching heuristics. In contrast, Weaviate's HNSW index shows consistent, low-latency approximate nearest neighbour search for semantic models.

6 Conclusions

In this paper we introduce SMARTMATCH, an open-source semantic retrieval system for TM. The performance of system clearly shows that modern sentence encoders with a vector database can be included into a TM pipeline with very low latency rates for dense retrieval. The system offers an architecture that enables comparison between different retrieval strategies and allows users to inspect source-target pairs with the latency per query displayed. The results show that semantic models outperform traditional edit-distance-based methods when lexical overlap is low, and the coverage rates

reveal that traditional methods fail to return candidates in almost 1 out of 5 queries.

A key limitation of our paper is that it does not cover integration aspects that real CAT tools need, such as user management, security, plugin APIs to existing CAT environments, or multi-tenant scaling. While the retrieval core is realistic, it would be beneficial to address these workflows. Another limitation is that there is no mechanism to automatically adapt retrieval configuration based on the human feedback. However, a human-in-the-loop reinforcement learning TM retrieval system could be explored based on our approach.

7 Acknowledgements

This research has been funded by the Generalitat Valenciana (Conselleria d'Educació, Investigació, Cultura i Esport) through the project: The limits and future of data-driven approaches: A comparative study of deep learning, knowledge-based and rule-based models and methods in Natural Language Processing (CIDEXG/2023/13), and by the Ministry of Science, Innovation, and Universities through the project *Safewords* (AIA2025-163322-C63). Publication fees and conference participation were supported by the University of Luxembourg.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Andrea Silvestre Baquero and Ruslan Mitkov. 2017. Translation memory systems have a long way to go. In *Proceedings of the Workshop Human-Informed Translation and Interpreting Technology*, pages 44–51.

Eduard Barbu, Carla Parra Escartín, Luisa Bentivogli, Matteo Negri, Marco Turchi, Constantin Orasan, and Marcello Federico. 2016. The first automatic translation memory cleaning shared task. *Machine Translation*, 30(3):145–166.

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300.

Qian Cao and Deyi Xiong. 2018. Encoding gated translation memory into neural machine translation. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3042–3047.

Xin Cheng, Shen Gao, Lemao Liu, Dongyan Zhao, and Rui Yan. 2022. Neural machine translation with contrastive translation memories. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3591–3601, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Norman Cliff. 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological bulletin*, 114(3):494.

Spence Green, Jason Chuang, Jeffrey Heer, and Christopher D Manning. 2014. Predictive translation memory: A mixed-initiative system for human language translation. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 177–187.

Rohit Gupta and Constantin Orasan. 2014. Incorporating paraphrasing in translation memory matching and retrieval. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation*, pages 3–10.

Rohit Gupta, Constantin Orăsan, Marcos Zampieri, Mihaela Vela, Josef van Genabith, and Ruslan Mitkov. 2016. Improving translation memory matching and retrieval using paraphrases. *Machine Translation*, 30(1):19–40.

Qiuxiang He, Guoping Huang, Lemao Liu, and Li Li. 2019. Word position aware translation memory for neural machine translation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 367–379. Springer.

Miguel A Jiménez-Crespo. 2010. The effect of translation memory tools in translated web texts: Evidence from a comparative product-based study. *Linguistica antverpiensia*, 8:213–232.

Liangyou Li, Andy Way, and Qun Liu. 2014. A discriminative framework of integrating translation memory features into smt. In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas: MT Researchers Track*, pages 249–260.

Ruslan Mitkov. 2022. Translation memory systems. In *The Routledge Handbook of Translation and Memory*, pages 364–380. Routledge.

Yongyu Mu, Abudurexiti Reheman, Zhiquan Cao, Yuchun Fan, Bei Li, Yinqiao Li, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2023. Augmenting large language model translators via translation memories. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10287–10299, Toronto, Canada. Association for Computational Linguistics.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Matteo Negri, Duygu Ataman, Masoud Jalili Sabet, Marco Turchi, and Marcello Federico. 2017. Automatic translation memory cleaning. *Machine Translation*, 31(3):93–115.

Jiyoung Park. 2024. Ethical approach to translation memory reuse: discussions from copyright and business ethics perspectives. *Translation Studies*, 17(1):37–52.

Raul Pașcalau, Laura-Rebeca Stiegelbauer, and Dumitra Mădălina Pantea. 2025. The importance of translation workflow in global business. *Studii de Știință și Cultură*, 21(2).

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Ming Qian and Chuiqing Kong. 2024. Enabling human-centered machine translation using concept-based large language model prompting and translation memory. In *International Conference on Human-Computer Interaction*, pages 118–134. Springer.

Tharindu Ranasinghe, Hansi Hettiarachchi, Constantin Orasan, and Ruslan Mitkov. 2025. MUSTS: MUltilingual semantic textual similarity benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 331–353, Vienna, Austria. Association for Computational Linguistics.

Tharindu Ranasinghe, Ruslan Mitkov, Constantin Orăsan, and Rocío Caro Quintana. 2021. Semantic textual similarity based on deep learning. *Corpora in Translation and Contrastive Research in the Digital Age: Recent advances and explorations*, 158:101.

Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2020. Intelligent translation memory matching and retrieval with sentence encoders. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 175–184, Lisboa, Portugal. European Association for Machine Translation.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Uwe Reinke. 2018. State of the art in translation memory technology. *Language Technologies for a Multilingual Europe; Rehm, G., Stein, D., Sasaki, F., Witt, A., Eds*, pages 55–84.

Bernard Rosner, Robert J Glynn, and Mei-Ling T Lee. 2006. The wilcoxon signed rank test for paired comparisons of clustered data. *Biometrics*, 62(1):185–192.

Masoud Jalili Sabet, Matteo Negri, Marco Turchi, and Eduard Barbu. 2016. An unsupervised method for automatic translation memory cleaning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 287–292.

Pilar Sánchez-Gijón, Joss Moorkens, and Andy Way. 2019. Post-editing neural machine translation versus translation memory segments. *Machine Translation*, 33(1):31–59.

Satoshi Sato and Makoto Nagao. 1990. Toward memory-based translation. In *COLING 1990 Volume 3: Papers Presented to the 13th International Conference on Computational Linguistics*.

Anne Gram Schjoldager and Tina Paulsen Christensen. 2010. Translation-memory (tm) research: what do we know and how do we know it? *Hermes*, 44:89–101.

Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. 2012. DGT-TM: A freely available translation memory in 22 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 454–459, Istanbul, Turkey. European Language Resources Association (ELRA).

Takuya Tamura, Xiaotian Wang, Takehito Utsuro, and Masaaki Nagata. 2023. Target language monolingual translation memory based nmt by cross-lingual retrieval of similar translations and reranking. In *Proceedings of Machine Translation Summit XIX, Vol. 1: Research Track*, pages 313–323.

Katerina Raisa Timonera and Ruslan Mitkov. 2015. Improving translation memory matching through clause splitting. In *Proceedings of the Workshop Natural Language Processing for Translation Memories*, pages 17–23.

Kun Wang, Chengqing Zong, and Keh-Yih Su. 2013. Integrating translation memory into phrase-based machine translation during decoding. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–21, Sofia, Bulgaria. Association for Computational Linguistics.

Friedel Wolff. 2016. Combining off-the-shelf components to clean a translation memory. *Machine Translation*, 30(3):167–181.

Jitao Xu, Josep Crego, and François Yvon. 2023. Integrating translation memories into non-autoregressive machine translation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1326–1338, Dubrovnik, Croatia. Association for Computational Linguistics.

Tianfu Zhang, Heyan Huang, Chong Feng, and Xiaochi Wei. 2020. Similarity-aware neural machine translation: reducing human translator efforts by leveraging high-potential sentences with translation memory. *Neural Computing and Applications*, 32(23):17623–17635.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A SMARTMATCH Seeding Page

To run SMARTMATCH, the first step is to feed the retrieval system with random samples from your TM.

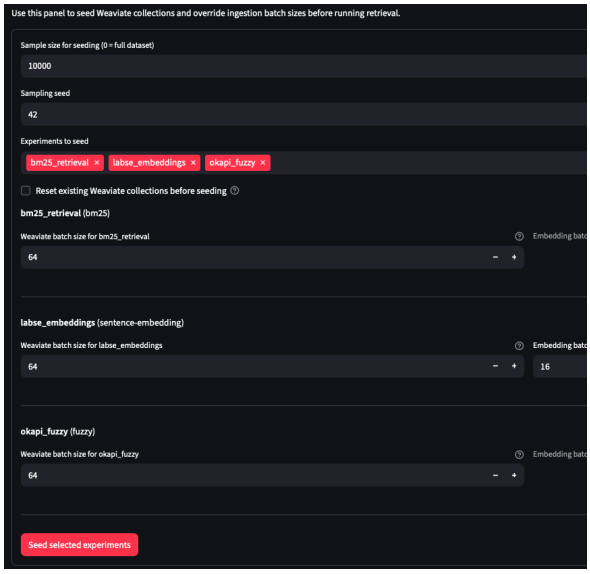

Figure 2: Seeding page

The experiment configuration samples random training data under the corresponding batch sizes. Then, the app iterates over the TM data, encodes the segments with the selected model, and stores them in a dedicated Weaviate collection. Before retrieval can run, both the semantic and BM25 models must be seeded into Weaviate, and Pensieve must generate the memory files. If a collection is empty, the retrieval page will not run.

B SMARTMATCH Retrieval Page

In the Retrieval tab, a random test sample may be selected from the original test set. The output reports how many segments are matched within the translation memory, as well as the corresponding latencies. Once segments have been retrieved, the view can be configured to display only the required information (source text, target text, score or index). In Figure 3, Okapi Pensieve serves as the traditional TM fuzzy matcher, as it compares strings and assigns higher scores when a segment is nearly identical to an entry in the memory.

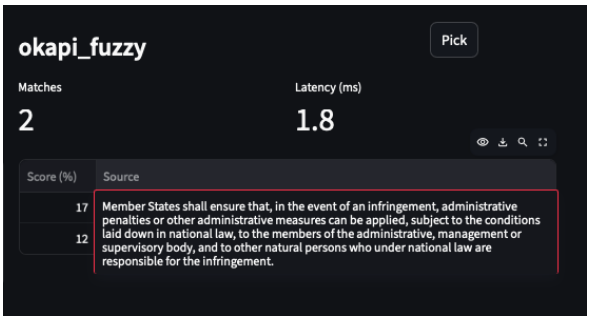

Figure 3: Okapi retrieval example

The second model, LaBSE, encodes each sentence into a vector and performs retrieval over a dense index in Weaviate, targeting segments with similar meaning even when the wording differs substantially, as shown in Figure 4.

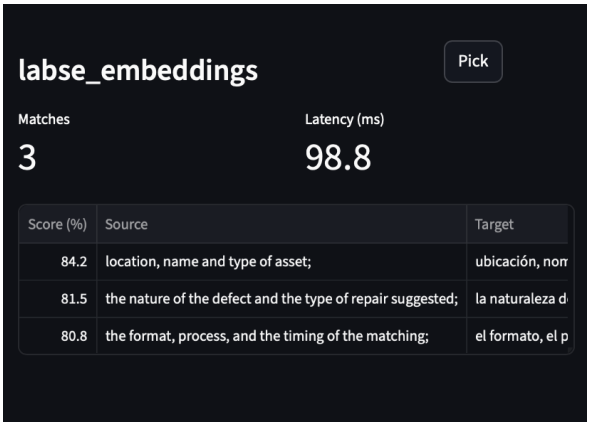

Figure 4: LaBSE retrieval example

BM25 is the last model, which operates on a lexical index in Weaviate. It is still word-based, but relies on term frequency and inverse document frequency, which enables more robust handling of word reordering and minor wording changes than simple edit distance.

After examining the results, the model that provides the most suitable concordances—taking into account both latency and match quality can be selected.

C SMARTMATCH Feedback Page

The Feedback charts page provides an interactive evaluation interface. For each query, the top suggestions from the three models are displayed, and the preferred option is selected directly in the interface.

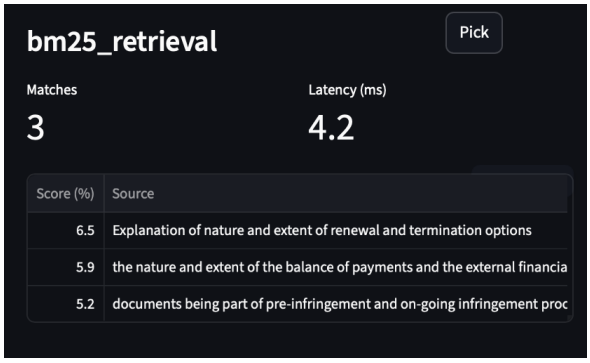

Figure 5: BM25 retrieval example

Each selection is recorded in a JSON feedback file, and the chart is updated in real time. This function-

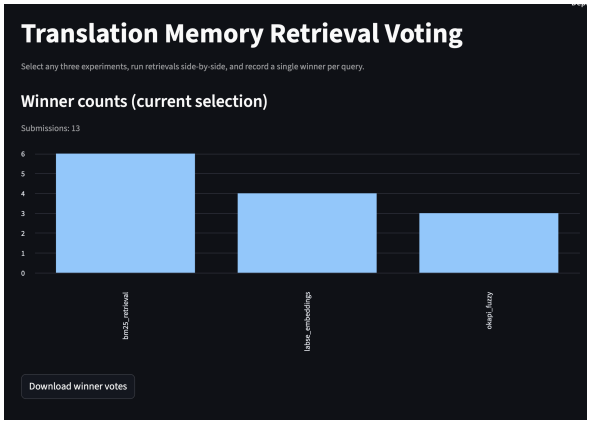

Figure 6: Feedback charts

ality can be used to compare model performance, identify which model performs best for specific segments, and collect preference data for further tuning or analysis.