OMNI TM-AE: A SCALABLE AND INTERPRETABLE EMBEDDING MODEL USING THE FULL TSETLIN MACHINE STATE SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

The increasing complexity of large-scale language models has amplified concerns regarding their interpretability and reusability. While traditional embedding models like Word2Vec and GloVe offer scalability, they lack transparency and often behave as black boxes. Conversely, interpretable models such as the Tsetlin Machine (TM) have shown promise in constructing explainable learning systems, though they previously faced limitations in scalability and reusability. In this paper, we introduce the Omni Tsetlin Machine Autoencoder (Omni TM-AE), a novel embedding model that fully exploits the information contained in the TM's state matrix, including literals previously excluded from clause formation. This method enables the construction of reusable, interpretable embeddings through a single training phase. We further introduce the Tsetlin State-Space Explorer (TSSE), an interactive interpretability tool for analyzing clause-level state distributions and tracing literal-state changes through training, linking learned representations to clauses, feedback events, and supporting training evidence. Extensive experiments across semantic similarity, sentiment classification, word analogy, document clustering, and quantitative interpretability evaluations show that Omni TM-AE performs competitively with mainstream embedding models while providing vocabulary-level traceability. These results demonstrate that it is possible to balance performance, scalability, and interpretability in Natural Language Processing (NLP) systems without resorting to opaque architectures.

1 INTRODUCTION

Modern Natural Language Processing (NLP) models commonly represent tokens using dense, multi-dimensional floating-point embeddings learned through extensive training. While effective, such representations provide limited direct insight into how individual dimensions relate to specific linguistic features, making them difficult to trace and interpret, particularly when embedded within larger architectures such as Large Language Models (LLMs).

Tsetlin Machine (TM), a machine learning model employing logical decision-making processes, introduced a novel approach for pattern recognition and decision-making that mimics artificial intelligence while remaining interpretable (Granmo, 2018). The initial embedding model Tsetlin Machine Autoencoder (TM-AE), proposed in (Bhattarai et al., 2024), performed well in similarity calculations but faced scalability challenges beyond limited datasets, particularly in applications like clustering that demand scalable solutions. The scalability limitation in Bhattarai et al. (2024) arose because training was conducted on a full input vector. More specifically, to compute similarities for a set of words, the model required simultaneous input of all words in a single vector. For instance, if the model was trained to compute similarity for a vector containing three tokens x_1, x_2, and x_3, the resulting trained model would be tightly bound to this specific set of tokens $\{x_1, x_2, x_3\}$. Consequently, if the vector contents were subsequently modified, e.g., through the addition, deletion, or substitution of any token, the output would become invalid, necessitating complete retraining.

Kadhim et al. (2025) sought to address these limitations by proposing a scalable model that trained tokens individually rather than collectively. In that approach, each token within a vector could be trained separately, and context information for each could be independently collected. However,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

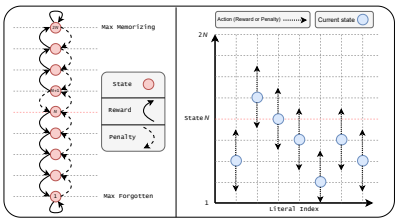

Figure 1: (Left) State-transition diagram for a single Tsetlin automaton, illustrating penalty/reward actions. (Right) An example of a clause construction space: the x-axis indices literals (vocabulary features and their negations) while the y-axis records their current automaton state. Literals above threshold N are selected to constitute the clause.

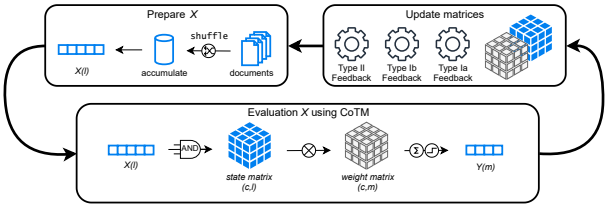

Figure 2: In the TM-AE structure the process of each example in epochs consists of three phases: (1) Preparation, where input vectors are generated from documents and encoded into binary form. (2) Evaluation, where input X is processed through a state matrix, weight matrix, and logical conditions to determine output predictions. (3) Update, where TAs in a clause adjust their states to include or exclude literals based on whether the clause matches the input data.

although that method succeeded in overcoming the initial scalability issue, it introduced new challenges. Specifically, the context information collected during the first training phase was insufficient for direct use in more complex tasks such as similarity measurement or clustering. If one wished, for example, to compute the similarity between two separately trained tokens, it was necessary to undergo a second training phase to establish relationships between them. This reliance on additional retraining significantly increased the model's complexity and training time, and limited its practicality for real-world applications. Additional related work is provided in Appendix A.1

In this work, we propose a novel method for generating reusable embeddings using only a single training phase, eliminating the need for retraining. In addition, to the best of our knowledge, this is the first approach to fully exploit all literals within the TM clause, including literals that are often excluded. By incorporating this previously excluded information, our model demonstrates substantial improvements in similarity measurement, classification, analogy, clustering, and interpretability across NLP applications. Furthermore, training time is greatly reduced by removing the dependency on additional training stages, thereby enhancing the model's reusability.

Our contributions are based on the following three key aspects:

1. Introducing the Omni Tsetlin Machine Autoencoder (Omni TM-AE), which utilizes information from both included and excluded literals to improve embedding quality.
2. Generating reusable embeddings in a single training phase that can be directly applied across multiple NLP tasks without retraining.
3. Developing the Tsetlin State-Space Explorer (TSSE), an interactive interpretability tool for visualizing literal states and tracing their training dynamics.

2 The Tsetlin Machine and Embedding Architecture

2.1 Tsetlin Automata and Clauses

The TM architecture comprises a collection of learning units referred to as Tsetlin automaton (TA) (Granmo, 2018). Each automaton undergoes state transitions based on reinforcement learning, either rewards or penalties, determined by its evaluation. Figure 1 (left) illustrates the state transition dynamics of an automaton for a literal. In this context, a $literal$ denotes either a feature from the vocabulary or its negation. Consequently, for a vocabulary of size d, the total number of literals is $2d$. The vertical axis in the figure represents the automaton's state, which ranges from complete forgetting (state 1) to maximal memorization (state $2N$). During training, the higher a literal's state, the more likely it is to be retained for clause formation. Literals that frequently contribute to accurate classification tend to ascend in the state space, whereas less relevant literals tend to descend.

To construct clauses, which are logical expressions used for classification, TA are organized into groups encompassing all literals (see an example in Figure 1, right). A clause is a conjunction (logical AND) of literals selectively chosen based on their state within the TA. The figure illustrates clause composition, where the x-axis denotes literal indices and the y-axis corresponds to TA states of literals. Literals above a defined threshold N are selected to construct the clause, reflecting their strong association with the target class. For instance, when training a class (word) labeled "happy," a resulting clause may take the form of "marriage AND love" or "family AND home", capturing contextual semantic relationships through logical expressions.

2.2 Autoencoder and Coalesced Mechanism

The TM-AE employs a specialized structure known as Coalesced Tsetlin Machine (CoTM) (Glimsdal & Granmo, 2021), which enables multi-output learning by training multiple target classes in parallel. The process, as shown in Figure 2, comprises three main stages: **Preparation Phase:** An input vector X, encoded in binary form, encapsulates document-level information. Its length l is twice the vocabulary size, given by $l = 2d$, to account for features and their negations. Training instances labeled with 1 are derived from documents containing the target class, whereas those with output 0 come from documents lacking that class. Documents are shuffled, and a selection is governed by the hyperparameter $accumulation$. Features present in the document are encoded as 1, and their negations as 0, and vice versa. **Evaluation Phase:** This stage assesses if the input sample X satisfies the logical expression of each clause. Clauses are formed by included literals, namely, the literals whose corresponding TAs have states exceeding the threshold N. Users can specify an arbitrary number of $clauses$, stored in a two-dimensional matrix $state_matrix$ with dimensions $c \times l$, where c is the number of clauses and l is the number of literals. The output is weighted using a $weight_matrix$ with dimensions $c \times m$, where m represents the number of classes. The final prediction Y is derived through majority voting. **Update Phase:** Clause updates are conducted based on the relationship between the input X, the clause (via its included literals), and the target output (the label). During training, the TA state within a clause can increase or decrease, effectively determining whether its corresponding literal is included in the clause. In essence, two types of updates, namely, Type I Feedback (including Type Ia and Ib) and Type II Feedback, are applied to the state and weight matrices. Details of the updates can be found in (Glimsdal & Granmo, 2021).

2.3 Omni Embedding Concept

To the best of our knowledge, all prior research in TM has utilized the concept of a clause to characterize the target class by focusing exclusively on literals whose TAs surpass state N. In other words, only the included literals in a clause contribute to the final classification. In contrast, the present study leverages the complete information encapsulated within the $state_matrix$, i.e., not only the included literals, but also the excluded literals, hence the term Omni. Although the excluded literals do not directly contribute to the final clause expression, they are nevertheless subject to training and may contain important information of the target class, which can be used for embedding.

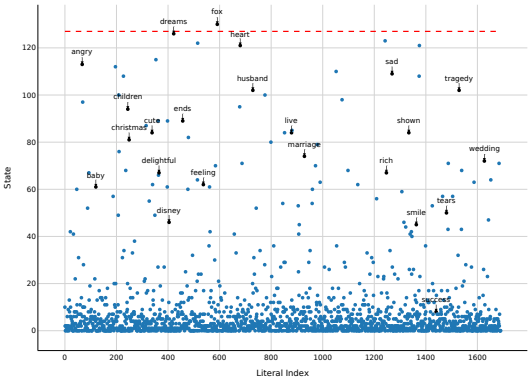

Figure 3: Visualization of a trained clause for the target class "happy," where training is performed with a smaller vocabulary (fewer than 1,800 tokens), showing that the word "fox" is the only literal exceeding the threshold state ($N = 128$, red line), while additional literals appear at lower state levels and provide latent contextual information.

To demonstrate the existence of useful information in the excluded literals, Figure 3 illustrates TAs inside a clause for a specific target class, "happy". Prior methods have focused on such clauses by considering only the included literals, i.e., whose TA states exceeding state N. Following this concept, the particular clause in Figure 3 has a single literal, i.e., "fox", that exceeded the N state (the red line where N = 128 in this ex-

periment). However, by observing Figure 3, it is evident that the excluded literals contain substantial informational content (see other annotated words in Figure 3, below state N, like "sad", "angry" and "rich"). These excluded literals also provide crucial cues for constructing an informative embedding vector for the target class, "happy".

In our proposed method, we introduce a novel technique for deriving the embedding vector grounded in the structure of the $state_matrix$. Using this method, the states of all literals, namely, both the original features and their negations, including the excluded ones, contribute to the formation of the embedding vector.

The embedding vector $\mathbf{e} \in \mathbb{Z}^d$ for a certain target word[1] is derived by training TM-AE for the target word and aggregating signed state values associated with each feature across all positive weighted clauses. Denote $\mathcal{D} = \{x_1, x_2, \ldots, x_d\}$ the set of original features (e.g., vocabulary tokens), and define a *literal* as either a feature x_i or its negation $\neg x_i$. Thus, the total number of distinct literals is $2d$. Let $\mathcal{C}$ be the set of all positive-weighted clauses for the target word. For each clause $c \in \mathcal{C}$, the TM maintains an automaton state for every literal in the full literal set $\{x_1, \neg x_1, \ldots, x_d, \neg x_d\}$. We denote the state of literal l in clause c by $n_c(l) \in \{1, 2, \ldots, 2N\}$. Accordingly,

$$S(c) = \{(l, n_c(l)) \mid l \in \{x_1, \neg x_1, \ldots, x_d, \neg x_d\}\}.$$

Thus, $S(c)$ contains the final state pairs for all $2d$ literals in clause c, not only the literals included in the final Boolean clause or updated during a specific training step. For this target word, the embedding vector has the same size as the vocabulary, namely d. For each feature x_i, where $i \in \{1, \ldots, d\}$, the signed state sum v_i is computed as:

$$v_i = \sum_{c \in \mathcal{C}} \left(n_c(x_i) - n_c(\neg x_i) \right).$$

This adds the state value of the original literal x_i and subtracts the state value of its negation $\neg x_i$ across all positive-weighted clauses, thereby capturing the net signed contribution of feature x_i. The count t_i is defined as the number of literal-state contributions used to compute v_i. Since both x_i and $\neg x_i$ are considered for each positive-weighted clause, t_i counts two contributions per clause $t_i = 2|\mathcal{C}|$. Based on this, the embedding component for each x_i is computed as:

$$e_i = \begin{cases} \left\lfloor \dfrac{v_i}{t_i} \right\rfloor & \text{if } t_i > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the complete embedding vector for the target word is given by $\mathbf{e} = [e_1, e_2, \ldots, e_d]$.

In summary, for each feature, the method consistently considers both the original literal and its negation by adding the state values of the feature and subtracting the state values of its negation. These contributions are normalized by the number of literal-state contributions used for each feature, ensuring that the embedding component captures the net influence of the feature in both positive and negative contexts. More details on literal state distribution, algorithm, and learning dynamics can be found in Appendix A.2 and A.3

2.4 Interpretability and State-Space Tool

A central property of Omni TM-AE is its vocabulary-aligned representation: each embedding dimension corresponds to a named vocabulary feature and is derived from the Tsetlin automaton states of its original and negated literals across clauses. Thus, individual embedding coordinates can be inspected through the literal states and clauses from which they are constructed rather than through anonymous latent dimensions.

To make this information practically accessible, we developed the TSSE, illustrated in Figure 4. Its state-space interface exposes original and negated literal states, their inclusion or exclusion from individual clauses, and clause polarity. For example, for the target word "driving," related words such as "vehicle," "license," "road," and "safe" can attain high states, allowing their contribution to the learned representation to be directly inspected.

[1]The notation displays only the embedding of a single target word. This approach is applicable to compute the embedding of any target word in the same way.

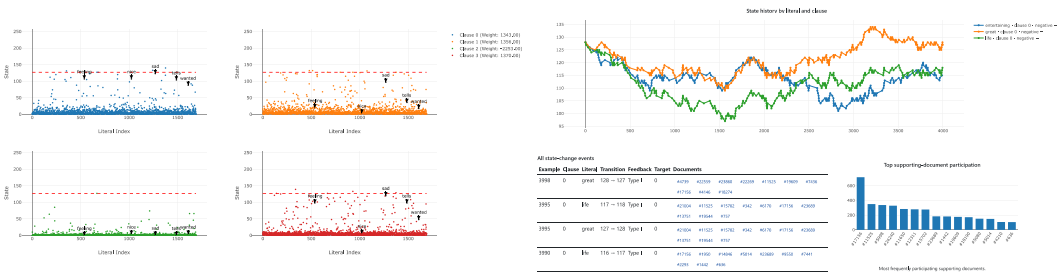

Figure 4: Example TSSE interfaces showing clause-level literal states (left) and training-history tracing with state changes and supporting documents (right).

TSSE additionally supports training-dynamics tracing. During diagnostic runs, a selected literal can be followed across training examples and clauses, with each recorded state transition associated with its previous and updated state, feedback type, target value, and the documents contributing to the synthesized input X. This provides a direct trace from a final vocabulary-aligned embedding coordinate back to the learning events that shaped it.

The tool also summarizes supporting-document participation for selected literals. Frequently involved documents can be reconstructed and inspected together with their associated state changes, helping identify recurring contexts, sampling imbalance, or potentially misleading training evidence. Training-history recording is optional and is used only for interpretability and diagnostic analysis. The TSSE tool will be made publicly available online after the double-blind review period concludes.

3 Experiment Results

We evaluate Omni TM-AE on semantic similarity, sentiment classification under semantic perturbation, word analogy, and visual, quantitative clustering, quantitative interpretability and ablation analyses. All experiments were executed in containerized Ubuntu environments with CUDA on an NVIDIA DGX H100 server with 2.0 TiB of RAM. Three Python environments were used: one for ELMo with its required TensorFlow version, one with RAPIDS cuML for GPU-accelerated clustering, and a default environment for the remaining experiments. Unless otherwise stated, stochastic procedures used seed 42, and results reported with mean and standard deviation were repeated over five runs.

Training data were tokenized using `CountVectorizer` with binary encoding. Positive and negative examples for each target word were balanced at 50%. Tokens outside the fixed vocabulary were ignored, and multi-word expressions were not modeled separately. Detailed experimental settings and reproducibility information are provided in Appendix A.4.

3.1 Semantic Similarity Evaluation

We evaluate semantic similarity using five standard benchmarks: WS-353-Sim, MTURK287, MTURK771, RG65, and MEN. Model predictions are compared with human similarity scores using Spearman and Kendall rank correlations. We compare Word2Vec, FastText, GloVe, TM-AE, and Omni TM-AE, all trained on the One Billion Word Benchmark corpus (Chelba et al., 2014) with a fixed vocabulary of 40,000 words. Word2Vec, FastText, and GloVe used 100-dimensional embeddings, window size 5, and 25 epochs, while Omni TM-AE used 32 clauses, $T = 20000$, $s = 1.0$, 4 epochs, accumulation 24, and 8-bit Tsetlin automata. Full settings are provided in Appendix A.4.

As shown in Table 1, Omni TM-AE achieves the highest Spearman scores on RG65 and MEN, with 0.622 and 0.596, respectively. Its average Kendall correlation is 0.386, matching Word2Vec, while its average Spearman score of 0.536 exceeds GloVe (0.519) and remains close to Word2Vec (0.555).

Prior TM-based embeddings were not evaluated across all datasets because of their scalability limitations. TM-AE (Bhattarai et al., 2024) was evaluated on RG65, where it achieved lower performance and required about 8 hours of training, while Omni TM-AE computed the corresponding

Table 1: Comparison of Spearman (S) and Kendall (K) similarity measures across different datasets and models. Note: TM-AE results for datasets other than RG65 were not conducted due to time constraints.

Dataset	Word2Vec		Fast-Text		GloVe		TM-AE		Omni TM-AE	
	S	K	S	K	S	K	S	K	S	K
WS-353-Sim	0.602 ±4.2%	0.424 ±4.7%	0.603 ±0.5%	0.422 ±0.6%	0.518 ±1.6%	0.358 ±1.4%	N/A	N/A	0.461 ±2.3%	0.331 ±2.6%
MTURK287	0.567 ±2.1%	0.397 ±2.6%	0.576 ±0.5%	0.401 ±0.5%	0.521 ±0.6%	0.364 ±0.8%	N/A	N/A	0.516 ±1.7%	0.365 ±1.4%
MTURK771	0.487 ±5.7%	0.336 ±6.1%	0.487 ±0.4%	0.335 ±0.5%	0.457 ±0.4%	0.316 ±0.4%	N/A	N/A	0.486 ±0.9%	0.347 ±0.8%
RG65	0.544 ±7.0%	0.372 ±9.2%	0.507 ±0.9%	0.343 ±1.2%	0.536 ±3.3%	0.369 ±3.7%	0.506 ±0.0%	0.361 ±0.0%	0.622 ±2.6%	0.445 ±2.4%
MEN	0.577 ±4.3%	0.404 ±4.9%	0.581 ±0.3%	0.406 ±0.3%	0.563 ±0.4%	0.393 ±0.5%	N/A	N/A	0.596 ±0.4%	0.439 ±0.5%
Avg.	0.555	0.386	0.551	0.381	0.519	0.360	N/A	N/A	0.536	0.386

embeddings in 47 seconds from the pre-generated representation. The multi-phase method in (Kadhim et al., 2025) requires additional retraining, limiting its practicality for large-scale downstream use.

For additional context, Appendix A.7 reports results from several off-the-shelf pretrained transformer embedding models on the same similarity benchmarks; these are included as performance-range references rather than controlled baselines.

3.2 Sentiment Classification under Semantic Perturbation

Table 2: Sentiment classification accuracy across embedding sources and classifiers.

Embedding Source	**RF**	**LR**	**NB**	**SVM**	**MLP**	**TM**	**Average**
BERT	0.8387 ±0.2%	0.8505 ±0.3%	0.8313 ±0.1%	0.8152 ±0.2%	0.8053 ±0.5%	0.8344 ±0.7%	0.8292
ELMo	0.8444 ±0.1%	0.8539 ±0.2%	0.8305 ±0.1%	0.8348 ±0.3%	0.8561 ±0.2%	0.8348 ±0.4%	0.8424
FastText	0.7231 ±0.1%	0.7395 ±0.4%	0.8031 ±0.0%	0.7370 ±0.5%	0.7718 ±0.7%	0.6711 ±8.2%	0.7409
GloVe	0.8103 ±0.2%	0.8237 ±0.5%	0.8261 ±0.0%	0.8037 ±0.2%	0.8236 ±0.2%	0.7897 ±1.7%	0.8128
Omni TM-AE	0.8389 ±0.2%	0.8534 ±0.3%	0.8165 ±0.1%	0.8274 ±0.2%	0.8494 ±0.2%	0.8407 ±0.5%	0.8377
Word2Vec	0.8409 ±0.1%	0.8559 ±0.3%	0.8178 ±0.0%	0.8337 ±0.2%	0.8531 ±0.1%	0.8420 ±0.7%	0.8406

To evaluate the embeddings in a downstream task, we conduct sentiment classification on IMDb (Maas et al., 2011) using six classifiers: RF, LR, NB, SVM, MLP, and TM. We compare GloVe, Word2Vec, FastText, Omni TM-AE, BERT, and ELMo using a vocabulary of 20,000 tokens. Before classification, 5% of the words in each document are replaced using a label-conditioned perturbation: for positive documents, a replacement is randomly selected from the five most similar words, while for negative documents, it is selected from the five least similar words among the 400 nearest candidates. Full model and classifier configurations are provided in Appendix A.4.

As shown in Table 2, Omni TM-AE achieves an average accuracy of 0.8377, the third highest result after ELMo (0.8424) and Word2Vec (0.8406), and within 0.47 percentage points of the highest average. It also exceeds BERT (0.8292), GloVe (0.8128), and FastText (0.7409) under this evaluation.

Prior TM-based embeddings were not included because TM-AE (Bhattarai et al., 2024) does not scale to this setting, while the multi-phase method in (Kadhim et al., 2025) requires additional retraining that makes this experiment impractical.

3.3 WORD ANALOGY

Table 3: Average accuracy on semantic and syntactic analogy categories.

	GloVe	FastText	Word2Vec	Omni TM-AE
Semantic	0.3894	0.3572	0.4353	**0.4652**
Syntactic	0.3392	**0.4375**	0.4259	0.3174

We evaluate relational structure using the Google Analogy Benchmark, which contains semantic and syntactic relations of the form $a : b :: c : d$. Analogies are solved using the standard 3CosAdd formulation, $\mathbf{b} - \mathbf{a} + \mathbf{c}$, with candidates ranked by cosine similarity; out-of-vocabulary cases are excluded. Full per-category results and evaluation details are provided in Appendix A.5.

Omni TM-AE achieves the highest semantic analogy average of 0.4652, compared with Word2Vec (0.4353), GloVe (0.3894), and FastText (0.3572). In contrast, its syntactic average is 0.3174, below FastText (0.4375), Word2Vec (0.4259), and GloVe (0.3392). These results indicate stronger capture of semantic relations than morphology-driven syntactic transformations.

3.4 CLUSTERING ANALYSIS

We evaluate the clustering properties of Omni TM-AE at both word and document levels. First, embeddings were generated for 130 words manually grouped into 13 semantic categories. Figure 5 shows their two-dimensional t-SNE projection, where several semantic groups form visually coherent regions while related categories remain close in the embedding space. For example, words from the **Professions** and **Technology** groups appear near one another, reflecting their semantic overlap. The complete word groups, t-SNE configuration, and an analysis of selected cross-cluster relationships are provided in Appendix A.6.

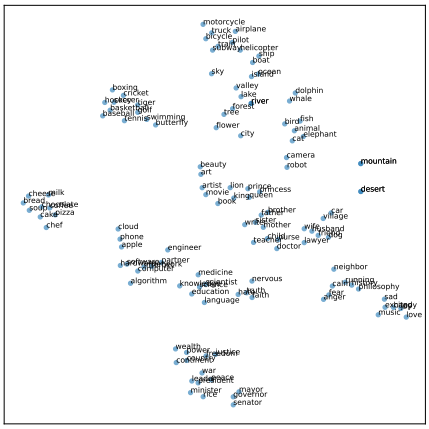

Figure 5: t-SNE visualization of Omni TM-AE embeddings for 130 words grouped into 13 semantic categories.

For quantitative evaluation, document embeddings were constructed by aggregating their constituent word embeddings and evaluated on five labeled datasets: 20 News, Reuters, Yelp, Amazon, and AG News. Clustering quality was measured using Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). All embedding models followed the same training conditions as the similarity experiment. Because Omni TM-AE produces vocabulary-sized representations, its document embeddings were reduced to 100 dimensions using PCA and subsequently L_2-normalized before clustering.

As shown in Table 4, Omni TM-AE achieves the highest NMI on Yelp (0.5715). Its average NMI and ARI are 0.3354 and 0.1102, respectively, compared with 0.3883 and 0.1535 for Word2Vec and 0.3857 and 0.1533 for FastText. Thus, while Omni TM-AE does not match the strongest dense baselines on average, it supports both word- and document-level clustering while retaining vocabulary-level interpretability.

3.5 INTERPRETABILITY EVALUATION

We quantitatively evaluate interpretability using four external semantic-feature norm datasets: McRae, Lancaster, THINGS, and SWOW. We measure feature precision at $K = \{3, 6, 10\}$, word-intrusion accuracy at the same values of K, and faithfulness based on the effect of removing top-ranked explanatory features. For Omni TM-AE, explanations are obtained directly from vocabulary-aligned

Table 4: Clustering performance comparison of the Omni TM-AE embedding model against baseline models (Word2Vec, FastText, GloVe) using five labeled datasets.

Model	Metric	20 News	Reuters	Yelp	Amazon	AG News	Average
Fasttext	NMI	0.2435 ±0.0%	0.4718 ±0.0%	0.5498 ±0.0%	0.1530 ±1.1%	0.5106 ±0.0%	0.3857
	ARI	0.1111 ±0.0%	0.1196 ±0.0%	0.0060 ±0.0%	0.0475 ±2.3%	0.4821 ±0.0%	0.1533
GloVe	NMI	0.2342 ±0.0%	0.4553 ±0.0%	0.5498 ±0.0%	0.1275 ±0.2%	0.4760 ±0.0%	0.3686
	ARI	0.1154 ±0.0%	0.0928 ±0.0%	0.0050 ±0.0%	0.0397 ±0.2%	0.4500 ±0.0%	0.1406
Omni TM-AE	NMI	0.2338 ±1.8%	0.4080 ±1.0%	0.5715 ±0.1%	0.0798 ±4.5%	0.3837 ±0.0%	0.3354
	ARI	0.1027 ±2.9%	0.0778 ±6.0%	0.0024 ±4.0%	0.0184 ±5.7%	0.3498 ±0.0%	0.1102
Word2Vec	NMI	0.2558 ±0.0%	0.4681 ±0.0%	0.5494 ±0.0%	0.1535 ±0.1%	0.5145 ±0.0%	0.3883
	ARI	0.1179 ±0.0%	0.1052 ±0.2%	0.0056 ±0.0%	0.0496 ±0.2%	0.4893 ±0.0%	0.1535

literal states, while dense baselines use nearest-neighbor evidence in their embedding spaces. Full evaluation details are provided in Appendix A.4.

As shown in Table 5, Omni TM-AE achieves the highest P@3 on all four datasets and the highest P@6 on McRae, Lancaster, and SWOW. The largest improvement occurs on Lancaster, where it reaches 76.1% P@3 and 67.6% P@6, compared with the strongest dense-baseline scores of 48.4% and 52.9%. It also achieves the best intrusion results across all three K values on Lancaster. Dense baselines obtain near-ceiling intrusion scores on several other datasets, although these reflect semantic coherence of nearest-neighbor explanations rather than intrinsic vocabulary-aligned evidence.

Faithfulness provides a clearer distinction between the two representation types. Omni TM-AE achieves 46.8%, 57.3%, and 75.0% on McRae, THINGS, and SWOW, respectively, substantially exceeding the dense baselines on these datasets. Lancaster is the main exception, where Omni TM-AE achieves strong semantic precision and intrusion scores but lower faithfulness. Overall, the results show that its vocabulary-aligned evidence is both semantically meaningful and, in most evaluated settings, more directly tied to the learned representation.

3.6 Ablation of Embedding Extraction

To isolate the contribution of the main Omni TM-AE embedding components, we compare the full formulation with three variants: excluding states below the literal-inclusion threshold, removing negated literals, and removing the signed difference between original and negated literal states. Table 6 summarizes their performance across semantic similarity, analogy, and interpretability.

The full formulation provides the strongest overall balance across the evaluated tasks. Removing excluded literals substantially reduces similarity and analogy performance, demonstrating that states below the inclusion threshold contain useful embedding information. Removing negated literals particularly affects analogy and intrusion performance, supporting the use of both original and negated literal states. Removing signed subtraction causes the largest degradation in similarity and faithfulness. Together, these results support constructing Omni TM-AE embeddings from the full TM state space rather than only from literals included in the learned clauses.

4 Conclusion

This work presents Omni TM-AE, a novel and interpretable embedding model based on the Tsetlin Machine Autoencoder. By integrating the information from all literals, including those traditionally excluded from clause formation, the proposed method constructs reusable and scalable embeddings using only a single training phase. This eliminates the retraining requirements found in earlier

Table 5: Interpretability evaluation on four semantic-feature norm datasets. Results are mean ± standard deviation over five runs; best mean values are shown in bold.

Data	Method	P@3	P@6	P@10	Intr.@3	Intr.@6	Intr.@10	Faith.
McRae	GloVe	15.3 ± 0.3	12.5 ± 0.2	10.5 ± 0.2	**100.0 ± 0.0**	**99.8 ± 0.3**	99.3 ± 0.3	10.5 ± 0.2
	FastText	15.0 ± 0.2	12.7 ± 0.1	**11.1 ± 0.2**	**100.0 ± 0.0**	**99.8 ± 0.3**	**99.7 ± 0.3**	6.9 ± 0.1
	Word2Vec	14.7 ± 0.2	12.5 ± 0.2	11.0 ± 0.1	**100.0 ± 0.0**	99.8 ± 0.3	**99.7 ± 0.3**	7.0 ± 0.1
	Omni TM-AE	**20.2 ± 0.4**	**14.4 ± 0.3**	10.3 ± 0.2	97.6 ± 0.4	93.0 ± 1.3	92.3 ± 1.4	**46.8 ± 1.1**
Lanc.	GloVe	44.4 ± 0.4	51.1 ± 0.5	55.4 ± 0.2	88.8 ± 0.2	55.8 ± 0.4	25.2 ± 0.4	**37.0 ± 0.3**
	FastText	46.4 ± 0.8	50.4 ± 0.7	55.3 ± 0.1	89.6 ± 0.2	56.9 ± 0.6	23.6 ± 1.0	21.7 ± 1.2
	Word2Vec	48.4 ± 0.2	52.9 ± 0.2	55.6 ± 0.1	89.7 ± 0.3	58.5 ± 0.5	26.1 ± 0.4	24.4 ± 0.2
	Omni TM-AE	**76.1 ± 0.0**	**67.6 ± 0.0**	**56.6 ± 0.0**	**95.1 ± 0.2**	**75.3 ± 0.3**	**29.8 ± 0.4**	1.0 ± 0.1
THINGS	GloVe	23.2 ± 0.6	**19.4 ± 0.3**	**16.5 ± 0.2**	**100.0 ± 0.0**	99.9 ± 0.2	99.8 ± 0.2	7.2 ± 0.2
	FastText	22.6 ± 0.7	18.5 ± 0.3	15.9 ± 0.2	**100.0 ± 0.0**	**100.0 ± 0.0**	**99.9 ± 0.2**	3.9 ± 0.1
	Word2Vec	21.9 ± 0.4	18.4 ± 0.4	15.9 ± 0.3	**100.0 ± 0.0**	99.9 ± 0.2	99.8 ± 0.2	3.9 ± 0.1
	Omni TM-AE	**27.3 ± 0.6**	18.2 ± 0.3	12.8 ± 0.2	98.6 ± 0.6	95.1 ± 0.7	94.0 ± 1.2	**57.3 ± 1.6**
SWOW	GloVe	34.2 ± 0.4	27.8 ± 0.2	23.1 ± 0.1	**100.0 ± 0.0**	100.0 ± 0.1	**99.9 ± 0.0**	8.5 ± 0.1
	FastText	34.5 ± 0.1	28.5 ± 0.2	**24.0 ± 0.1**	**100.0 ± 0.0**	**100.0 ± 0.0**	**99.9 ± 0.1**	5.9 ± 0.1
	Word2Vec	35.2 ± 0.1	28.7 ± 0.1	**24.0 ± 0.0**	**100.0 ± 0.0**	100.0 ± 0.1	**99.9 ± 0.1**	6.0 ± 0.1
	Omni TM-AE	**38.5 ± 0.2**	**30.1 ± 0.4**	21.3 ± 0.2	99.9 ± 0.1	98.4 ± 0.3	96.1 ± 0.6	**75.0 ± 0.7**

Table 6: Macro-level ablation summary for Omni TM-AE embedding extraction variants.

Evaluation	Metric	Full	w/o Excluded	w/o Negated	w/o Signed Sub.
Similarity	Spearman	0.466 ± 0.186	0.188 ± 0.126	0.210 ± 0.137	-0.412 ± 0.164
Similarity	Kendall	0.336 ± 0.135	0.153 ± 0.102	0.169 ± 0.109	-0.292 ± 0.117
Analogy	Macro Avg.	0.297 ± 0.198	0.076 ± 0.033	0.158 ± 0.140	0.291 ± 0.195
Analogy	Semantic Avg.	0.368 ± 0.206	0.085 ± 0.050	0.292 ± 0.164	0.361 ± 0.202
Analogy	Syntactic Avg.	0.255 ± 0.143	0.071 ± 0.040	0.077 ± 0.044	0.249 ± 0.139
Interpretability	P@3	35.265 ± 25.509	31.391 ± 19.635	32.290 ± 18.347	14.530 ± 18.979
Interpretability	P@6	29.611 ± 26.023	27.570 ± 28.189	30.759 ± 25.902	14.149 ± 22.806
Interpretability	Intr.@6	72.771 ± 7.307	9.589 ± 5.100	41.296 ± 26.057	65.297 ± 17.098
Interpretability	Faith.	36.003 ± 25.197	33.771 ± 24.507	37.340 ± 23.229	3.859 ± 4.646

TM-based approaches and significantly reduces model complexity. Our experiments across semantic similarity, classification, word analogy, clustering, and quantitative interpretability tasks reveal that Omni TM-AE achieves competitive performance across several tasks while retaining vocabulary-level interpretability. In summary, Omni TM-AE marks a substantial advancement in bridging the gap between interpretable learning and high-performing NLP embeddings, laying the groundwork for more responsible and comprehensible AI systems.

AI USE STATEMENT

Generative AI tools were used only to improve the writing of this manuscript, including grammar, clarity, conciseness, and language polishing. They were not used to generate research ideas, methodology, experimental results, data analysis, code, or scientific claims. All AI-assisted edits were reviewed and verified by the authors, who take full responsibility for the final content of the manuscript.

REPRODUCIBILITY STATEMENT

Experimental settings and implementation details are provided in Appendix A.4. An anonymized source-code package is provided as supplementary material for reproducibility, and the code will be made publicly available online after the double-blind review period concludes.

REFERENCES

K. Darshana Abeyrathna, Ahmed A. O. Abouzeid, Bimal Bhattarai, Charul Giri, Sondre Glimsdal, Ole-Christoffer Granmo, Lei Jiao, Rupsa Saha, Jivitesh Sharma, Svein A. Tunheim, and Xuan Zhang. Building concise logical patterns by constraining Tsetlin machine clause size. In Edith Elkind (ed.), *IJCAI*, pp. 3395–3403, 8 2023.

Kuruge Darshana Abeyrathna, Bimal Bhattarai, Morten Goodwin, Saeed Rahimi Gorji, Ole-Christoffer Granmo, Lei Jiao, Rupsa Saha, and Rohan K. Yadav. Massively parallel and asynchronous Tsetlin machine architecture supporting almost constant-time scaling. In Marina Meila and Tong Zhang (eds.), *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10–20. PMLR, 18–24 Jul 2021.

Bimal Bhattarai, Ole-Christoffer Granmo, Lei Jiao, Rohan Yadav, and Jivitesh Sharma. Tsetlin machine embedding: Representing words using logical expressions. *Findings of EACL*, pp. 1512–1522, 2024.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2014. URL https://arxiv.org/abs/1312.3005.

Sondre Glimsdal and Ole-Christoffer Granmo. Coalesced multi-output Tsetlin machines with clause sharing. *CoRR*, 2021. URL https://arxiv.org/abs/2108.07594.

Ole-Christoffer Granmo. The Tsetlin machine - a game theoretic bandit driven approach to optimal pattern recognition with propositional logic, 2018. URL https://arxiv.org/abs/1804.01508.

Ylva Grønningsæter, Halvor S. Smørvik, and Ole-Christoffer Granmo. An optimized toolbox for advanced image processing with Tsetlin machine composites. In *2024 International Symposium on the Tsetlin Machine (ISTM)*, pp. 1–8, 2024.

Shannon How, Jagmohan Chauhan, Geoff Merrett, and Jonathon Hare. FedTMOS: Efficient one-shot federated learning with tsetlin machine. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Representation Learning*, volume 2025, pp. 52263–52282, 2025.

Sindhusha Jeeru, Lei Jiao, Per-Arne Andersen, and Ole-Christoffer Granmo. Interpretable rule-based architecture for GNSS jamming signal classification. *IEEE Sensors Journal*, 2025.

Lei Jiao, Xuan Zhang, Ole-Christoffer Granmo, and K. Darshana Abeyrathna. On the convergence of Tsetlin machines for the XOR operator. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(5):6072–6085, Jan. 2023.

Ahmed K. Kadhim, Ole-Christoffer Granmo, Lei Jiao, and Rishad Shafik. Exploring state space and reasoning by elimination in Tsetlin machines. In *2024 International Symposium on the Tsetlin Machine (ISTM)*, pp. 1–8. IEEE Computer Society, 2024.

Ahmed K. Kadhim, Lei Jiao, Rishad Shafik, Ole-Christoffer Granmo, and Bimal Bhattarai. Scalable multi-phase word embedding using conjunctive propositional clauses. In *2025 International Symposium on the Tsetlin Machine (ISTM)*, pp. 107–115, 2025.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Sidharth Maheshwari, Tousif Rahman, Rishad Shafik, Alex Yakovlev, Ashur Rafiev, Lei Jiao, and Ole-Christoffer Granmo. REDRESS: Generating compressed models for edge inference using Tsetlin machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):11152–11168, 2023. doi: 10.1109/TPAMI.2023.3268415.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, pp. 1532–1543, 2014.

Raihan Seraj, Jivitesh Sharma, and Ole-Christoffer Granmo. Tsetlin machine for solving contextual bandit problems. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30194–30205. Curran Associates, Inc., 2022.

Jivitesh Sharma, Rohan Yadav, Ole-Christoffer Granmo, and Lei Jiao. Drop clause: Enhancing performance, robustness and pattern recognition capabilities of the Tsetlin machine. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13547–13555, Jun. 2023.

Svein Anders Tunheim, Lei Jiao, Rishad Shafik, Alex Yakovlev, and Ole-Christoffer Granmo. Tsetlin machine-based image classification FPGA accelerator with on-device training. *IEEE Trans. Circuits Syst. I, Reg. Papers*, 72(2):830–843, Feb. 2025a.

Svein Anders Tunheim, Yujin Zheng, Lei Jiao, Rishad Shafik, Alex Yakovlev, and Ole Christoffer Granmo. An all-digital 8.6-nj/frame 65-nm Tsetlin machine image classification accelerator. *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2025b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6000 – 6010. Curran Associates, Inc., 2017.

Gideon Vos, Maryam Ebrahimpour, Liza van Eijk, Zoltan Sarnyai, and Mostafa Rahimi Azghadi. Decoding neural emotion patterns through large language model embeddings. *Neurocomputing*, 669:132513, 2026. ISSN 0925-2312.

Rohan K Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin. Human-level interpretable learning for aspect-based sentiment analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14203–14212, May 2021a.

Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin. Enhancing interpretable clauses semantically using pretrained word representation. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 265–274. Association for Computational Linguistics, November 2021b.

Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin. Robust interpretable text classification against spurious correlations using AND-rules with negation. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 4439–4446. International Joint Conferences on Artificial Intelligence Organization, 7 2022.

Xuan Zhang, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin. On the convergence of Tsetlin machines for the IDENTITY- and NOT Operators. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44 (10):6345–6359, Jul. 2022.

Xuan Zhang, Lei Jiao, and Ole-Christoffer Granmo. Convergence analysis of Tsetlin machines under noise-free and noisy training conditions: From 2 bits to k bits. In *ICLR*, volume 2026, pp. 66002–66058, 2026.

Zixiao Zhu and Kezhi Mao. Knowledge-based bert word embedding fine-tuning for emotion recognition. *Neurocomputing*, 552:126488, 2023. ISSN 0925-2312.

A Appendix

A.1 Related Work

The embedding phase constitutes one of the foundational steps in most LLMs, including architectures such as Transformers (Vaswani et al., 2017). Early embedding models, such as GloVe (Pennington et al., 2014), introduced simple statistical methods for constructing word embeddings by counting the frequency of word occurrences within the training dataset. Word2Vec (Mikolov et al., 2013) and its improved version FastText (Bojanowski et al., 2017) advanced this field by providing a unique representational space for symbols through the collection of contextual information from training data. These models successfully replaced earlier statistical NLP methods for embedding construction and paved the way for the development of LLMs by integrating embeddings at multiple stages of their architecture Vos et al. (2026). However, these models inherently lack interpretability and explainability, and they treat tokens as mathematical entities within a black-box framework trained solely to extract contextual features from datasets Zhu & Mao (2023).

TM represents a distinct paradigm in machine learning, characterized by its transparent and interpretable structure (Granmo, 2018). It employs logical expressions to construct patterns that reveal the underlying classes they describe (Glimsdal & Granmo, 2021; Kadhim et al., 2024; Yadav et al., 2022; Abeyrathna et al., 2023; Sharma et al., 2023). The TM has shown strong performance across a wide range of applications (Abeyrathna et al., 2021), including aspect-based sentiment analysis (Yadav et al., 2021a), text classification (Yadav et al., 2021b; 2022), contextual bandit problems (Seraj et al., 2022), image classification (Grønningsæter et al., 2024; Jeeru et al., 2025), and federated learning (How et al., 2025). In addition, TMs are inherently hardware-friendly and exhibit high energy efficiency when implemented on hardware (Tunheim et al., 2025a; Maheshwari et al., 2023; Tunheim et al., 2025b), making them especially well-suited for IoT devices and edge computing scenarios where low power usage and computational efficiency are critical. While real-world deployment can introduce some variability, convergence analyses (Jiao et al., 2023; Zhang et al., 2022) demonstrate that TMs almost surely converge to fundamental Boolean operations, underscoring the robustness.

In the field of NLP, the implementation of TM as described by (Bhattarai et al., 2024) marked its first use in generating word embeddings, known as TM-AE that facilitate understanding the reasoning behind specific output results. However, a major limitation was the difficulty in reusing the training outputs or extending them to accommodate larger input vectors. In (Kadhim et al., 2025), an attempt was made to solve the problem of reusing context information. However, because that method relied heavily on the final clauses and the layers contained within it, the extracted information is usually insufficient to build embeddings through a single phase. Instead, the process required two distinct stages, thereby impeding the model's ability to achieve rapid embeddings for complex applications such as clustering. In the present work, we propose a novel TM-based autoencoder to construct a reusable and scalable single-stage embedding.

A.2 Literal State Distribution

To examine how large vocabularies affect literal state distributions and motivate the use of both original and negated literals, we analyze the clause-level state patterns shown in Figure 6. In this regime, training for the target word "happy" using a large vocabulary of 40,000 tokens reveals that clause-level state values for original literals concentrate within a narrow range (0 to 25). This limited range of the original literals' state values becomes more pronounced when the vocabulary size is expanded, from 1,800 tokens, as shown in Figure 3, to 40,000 tokens in Figure 6. Despite this limitation, increasing the vocabulary size offers the advantage of enabling more rapid convergence of the TM during the training process.

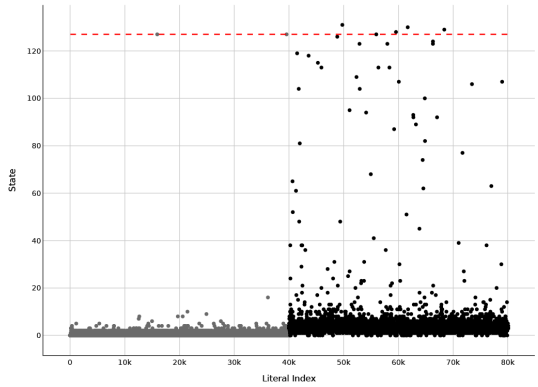

Figure 6: Visualization of a trained clause for the target class "happy," where training is performed with a large vocabulary of 40,000 tokens (80,000 literals including negations), illustrating the distribution of literal states and showing that the model reduces the state values of original literals to improve clause discrimination within a few epochs.

Algorithm 1 Embedding Vector Computation for a Target Word Using Positive Clauses

Require: Set of positive-weighted clauses $\mathcal{C}$ for the target word, where each clause $c \in \mathcal{C}$ contains $S(c)$ set of literal-state pairs (l, n), vocabulary $\mathcal{D} = \{x_1, x_2, \ldots, x_d\}$
Ensure: Embedding vector $\mathbf{e} \in \mathbb{Z}^d$

Function COMPUTEEMBEDDING($\mathcal{C}, \mathcal{D}$)
  Initialize $\mathbf{v} \leftarrow \mathbf{0}_d$ ▷ Signed state sums
  $t \leftarrow 2|\mathcal{C}|$ ▷ Counts x_i and $\neg x_i$ for each positive-weighted clause
  for $i \leftarrow 1$ **to** d **do**
    for all clause $c \in \mathcal{C}$ **do**
      for all $(l, n) \in S(c)$ **do**
        if l is of the form x_i **then**
          $v_i \leftarrow v_i + n$
        else if l is of the form $\neg x_i$ **then**
          $v_i \leftarrow v_i - n$
        end if
      end for
    end for
    $e_i \leftarrow \left\lfloor \frac{v_i}{t} \right\rfloor$
  end for
  return $\mathbf{e} = [e_1, e_2, \ldots, e_d]$

To overcome the constraint associated with the restricted state range of original literals (0 to 25), the proposed method computes the embedding by incorporating the state values of both the original literals and their negations with negated literals indexed from 40,001 to 80,000 in Figure 6. This inclusive strategy allows the model to leverage the full extent of the clause structure.

Compared to previous approaches, the proposed method introduces a novel framework for generating scalable and reusable embeddings that was not achieved in (Bhattarai et al., 2024). Moreover, it delivers a highly accurate and efficient embedding mechanism, while eliminating the complexity associated with multi-phase training procedures, as required in the work (Kadhim et al., 2025).

To provide a clear description of the proposed embedding computation process used in the Omni TM-AE model, Algorithm 1 presents the detailed steps required to construct the embedding vector for a target word from the clause-level literal states. The algorithm illustrates how the signed state values of both the original literals and their negations are aggregated, normalized, and combined into the final embedding vector. This formalization highlights the simplicity and efficiency of the proposed method, which eliminates the need for multi-phase training or complex post-processing steps.

A.3 Remarks on Logic-Driven Learning Dynamics of Omni Embeddings

The convergence behavior of Omni embeddings is rooted in the logic-based learning dynamics of the TM, which tightly couple representation learning to clause-level feedback and stabilization (Zhang et al., 2026). In TM learning, automaton state transitions are governed by two feedback types, namely, Type I feedback and Type II feedback (Granmo, 2018; Glimsdal & Granmo, 2021), whose activation depends on the clause output, the target polarity, and the current clause configuration. Type I feedback promotes literals that contribute to correct clause activations, either by reinforcing inclusion when a clause fires (Type Ia) or by discouraging unnecessary complexity when it does not (Type Ib). Type II feedback suppresses literals that cause false positive clause activations by promoting their exclusion. Through this mechanism, automaton states encode whether a literal is logically compatible with the propositional structure required to represent a target concept.

Building on this foundation, a defining characteristic of the Omni embedding is the explicit inclusion of negated literals. This enables the model to construct knowledge not only by describing what a target concept represents, but also by encoding what it does not represent. Such reasoning by elimination is a core property of the TM (Kadhim et al., 2024), where clauses are progressively refined by excluding literals that introduce contradictions. By aggregating both original and negated literal states, each embedding dimension captures the net logical role of a feature: positive values indicate reinforced inclusion, while negative values arise when the absence of a feature is systematically required to prevent false activations. This dual encoding allows concepts to be represented through patterns of inclusion and exclusion rather than descriptive presence alone.

As a consequence, the Omni embedding mechanism remains fundamentally logic-driven rather than statistically driven. During document preparation, input documents are binarized, and word frequencies or co-occurrence statistics within documents are ignored. Consequently, the learned representations reflect how literals participate in clause-level logical decisions formed through structured feedback mechanisms, rather than direct statistical associations as in models such as GloVe (Pennington et al., 2014).

This logic-centric formulation directly supports early stabilization of clause configurations and, in turn, efficient convergence of the embedding vectors. The stabilization mechanism is governed by the probabilistic clause update rule (Glimsdal & Granmo, 2021). Following the coalesced formulation, the probability of applying feedback to a clause is given by

$$P_{\text{update}} = \begin{cases} \frac{T - \sum_i w_i o_i}{2T}, & \text{for positive targets,} \\ \frac{T + \sum_i w_i o_i}{2T}, & \text{for negative targets,} \end{cases}$$

where $o_i \in \{0, 1\}$ denotes the output of clause i, w_i its associated class weight, and T a user-defined threshold controlling feedback frequency. As the clause ensemble increasingly agrees with the target concept, the weighted class sum $\sum_i w_i o_i$ approaches saturation, causing the update probability to decrease and thereby reducing the frequency of clause and automaton updates.

In large-vocabulary settings, each training example activates only a small subset of literals when building the binarized input vector X. However, the availability of many documents (e.g., approximately 30M documents in the One Billion Word dataset (Chelba et al., 2014)) and diverse logical contexts accelerates clause refinement by exposing clauses to a broad range of eliminative and confirmatory evidence. The specificity parameter s further regulates this process by controlling the sparsity of literal-level feedback, limiting the number of automata affected per update. Together, these mechanisms promote early stabilization of clause configurations and automaton states, after which subsequent training primarily reinforces already-established logical structures.

From an embedding perspective, this dynamic implies that Omni embeddings converge once the logical definition of the target concept has been established. Larger vocabularies and datasets therefore facilitate faster stabilization by providing sufficient logical evidence, yielding expressive and interpretable embeddings with few training passes, without reliance on statistical frequency information.

A.4 Experimental Setup

The proposed model was evaluated through six experiments: semantic similarity, classification, word analogy, visual and quantitative clustering, quantitative interpretability, and ablation analysis. Publicly available datasets and open-source embedding models were used throughout.

Semantic similarity. All embedding models were trained on the One Billion Word Benchmark (Chelba et al., 2014) using a fixed vocabulary of 40,000 words. Word2Vec, FastText, and GloVe used a vector dimension of 100, window size 5, and 25 training epochs, with other hyperparameters left at their default values. Omni TM-AE used $number_of_examples = 2000$, $accumulation = 24$, $clauses = 32$, $T = 20000$, $s = 1.0$, $epochs = 4$, and $number_of_state_bits_ta = 8$. Here, $accumulation$ denotes the number of documents used to construct an input vector X, T controls voting saturation, s controls literal inclusion and clause sparsity, and $number_of_state_bits_ta$ determines the number of bits used for each Tsetlin automaton state.

Sentiment classification. All embedding models were evaluated on IMDb (Maas et al., 2011) with a vocabulary limited to 20,000 tokens. Word2Vec, FastText, and GloVe used 100-dimensional embeddings, window size 5, and 25 training epochs, with other hyperparameters left at their defaults. Omni TM-AE used $number_of_examples = 2000$, $accumulation = 24$, $clauses = 32$, $T = 20000$, $s = 1.0$, $epochs = 4$, and $number_of_state_bits_ta = 8$.

BERT used batch size 8, learning rate 5×10^{-5}, three epochs, and gradient clipping with maximum norm 1.0. ELMo embeddings were obtained from the ELMo v3 TensorFlow Hub model using its default signature on pre-tokenized sentences.

For the classification stage, RF used $n_estimators = 100$, while LR, NB, SVM, and MLP used their default configurations with $random_state = 42$ where applicable. The TM classifier used $clauses = 1000$, $T = 8000$, $s = 2.0$, $epochs = 10$, and $clause_drop_p = 0.75$ (Sharma et al., 2023).

The perturbation replaced 5% of document tokens. For positive documents, the five nearest words according to the corresponding embedding were retrieved and one was selected randomly. For negative documents, the 400 nearest words were retrieved, from which the five least similar candidates were selected before randomly choosing one replacement.

Word analogy. The Google Analogy Benchmark was evaluated using the standard 3CosAdd formulation. For an analogy $a : b :: c : d$, the target vector was computed as $\mathbf{b} - \mathbf{a} + \mathbf{c}$, and candidate words were ranked by cosine similarity. Analogies containing out-of-vocabulary words were excluded. The same trained GloVe, FastText, Word2Vec, and Omni TM-AE embeddings used in the semantic-similarity evaluation were used for this experiment. Results were averaged over five runs.

A.5 Detailed Word Analogy Results

Table 7 reports the complete per-category results on the Google Analogy Benchmark. The benchmark includes semantic categories such as capital–country, city–state, currency, and family relations, as well as syntactic categories covering adjective transformations, tense, participles, nationality adjectives, and plurals. All results are reported as mean accuracy with standard deviation over five runs.

A.6 Additional Clustering Analysis

Word-level visualization. The qualitative clustering experiment used 130 words manually organized into 13 semantic groups. The t-SNE projection used $n_components = 2$, $perplexity = 30$, $learning_rate = 200$, $n_iter = 2000$, and $random_state = 42$.

Although most words appeared near their expected semantic groups, some cross-group relationships were observed. For example, "partner," assigned to the Relationships group, appeared near technology-related words. To examine this behavior, we analyzed its document-level co-occurrence patterns in the One Billion Word dataset. "Partner" co-occurred with "network" in 552 documents and with

Table 7: Per-category analogy accuracy on the Google Analogy Benchmark. Values show mean accuracy with standard deviation over five runs. Best mean results per row are shown in bold.

Category	GloVe	FastText	Word2Vec	Omni TM-AE
capital-common-countries	0.5842 ± 0.0095	0.6225 ± 0.0087	0.6261 ± 0.0055	**0.7984 ± 0.0503**
capital-world	0.6546 ± 0.0090	0.4745 ± 0.0032	0.7094 ± 0.0044	**0.8088 ± 0.0184**
city-in-state	0.1877 ± 0.0040	0.2074 ± 0.0044	0.2221 ± 0.0067	**0.2742 ± 0.0113**
currency	0.1225 ± 0.0108	0.0635 ± 0.0039	**0.1910 ± 0.0089**	0.0933 ± 0.0301
family	0.3979 ± 0.0111	0.4182 ± 0.0045	**0.4279 ± 0.0132**	0.3511 ± 0.0465
gram1-adjective-to-adverb	0.2340 ± 0.0061	**0.3609 ± 0.0083**	0.3497 ± 0.0084	0.1366 ± 0.0159
gram2-opposite	0.1921 ± 0.0078	**0.3522 ± 0.0073**	0.3011 ± 0.0077	0.0455 ± 0.0160
gram3-comparative	0.2805 ± 0.0070	0.3674 ± 0.0054	**0.3688 ± 0.0049**	0.3464 ± 0.0293
gram4-superlative	0.1942 ± 0.0067	0.3005 ± 0.0058	**0.3187 ± 0.0060**	0.1118 ± 0.0238
gram5-present-participle	0.3133 ± 0.0115	0.3831 ± 0.0098	0.3764 ± 0.0077	**0.3859 ± 0.0228**
gram6-nationality-adjective	0.5911 ± 0.0088	0.6483 ± 0.0024	0.6448 ± 0.0022	**0.7631 ± 0.0283**
gram7-past-tense	0.4187 ± 0.0069	0.5542 ± 0.0085	**0.5635 ± 0.0022**	0.3990 ± 0.0292
gram8-plural	0.5980 ± 0.0077	**0.6550 ± 0.0048**	0.6244 ± 0.0051	0.4517 ± 0.0250
gram9-plural-verbs	0.2310 ± 0.0100	**0.3159 ± 0.0059**	0.2857 ± 0.0045	0.2169 ± 0.0158
Semantic Avg	0.3894 ± 0.0038	0.3572 ± 0.0038	0.4353 ± 0.0050	**0.4652 ± 0.0148**
Syntactic Avg	0.3392 ± 0.0021	**0.4375 ± 0.0016**	0.4259 ± 0.0032	0.3174 ± 0.0084

"software" in 355 documents, compared with only 29 documents containing both "partner" and "engineer."

Because the TM-AE input X uses binary word-presence indicators, repeated occurrences of a word within the same document do not increase its contribution. However, repeated presence and absence across documents influence Tsetlin automaton updates and can therefore shape the learned literal states. This provides a traceable explanation for some of the cross-cluster relationships observed in Figure 5.

Document-level clustering. The five evaluation datasets contained different numbers of classes and documents. 20 News, Reuters, Yelp, Amazon, and AG News contained 20, 79, 11,537, 30, and 4 labels, respectively, with dataset sizes of 18,846, 10,788, 229,907, 21,000, and 120,000 documents.

A.7 Contextual Reference with Pretrained Transformer Embeddings

To further contextualize the semantic similarity results, we include a performance-range reference with several off-the-shelf pretrained transformer embedding models in Table 8. These models are not directly comparable to the embeddings trained in this study, since they are large pretrained language models trained on substantially broader corpora and are not trained from scratch on the One Billion Word Benchmark under the same experimental constraints. Therefore, the table is not intended as a controlled baseline comparison, but rather as an upper-reference indication of the level of performance achievable by modern pretrained sentence and text embedding models on the same word-similarity benchmarks.

Table 8: Word-similarity benchmark results for off-the-shelf pretrained transformer embedding models; ρ denotes Spearman and τ denotes Kendall.

Model	Metric	RG-65	WS-353-Sim	MTurk-287	MTurk-771	MEN	Avg.
SBERT MPNet	ρ	0.772	0.821	0.709	0.685	0.781	0.754
	τ	0.602	0.619	0.509	0.490	0.579	0.560
SBERT MiniLM	ρ	0.737	0.786	0.703	0.686	0.765	0.735
	τ	0.554	0.590	0.505	0.493	0.569	0.542
E5 Base	ρ	0.764	0.799	0.664	0.711	0.759	0.739
	τ	0.594	0.600	0.471	0.520	0.565	0.550
E5 Large	ρ	0.742	0.764	0.652	0.666	0.695	0.704
	τ	0.548	0.564	0.461	0.481	0.508	0.512
E5-Mistral	ρ	0.901	0.810	0.714	0.817	0.799	0.808
	τ	0.728	0.618	0.519	0.623	0.603	0.618
NV-Embed-v2	ρ	0.738	0.572	0.503	0.517	0.517	0.569
	τ	0.554	0.412	0.358	0.363	0.365	0.411