© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718361

Semantic Change Detection in Evolving Documents Using Contextual Embeddings and Transformer Models

NIKITA BACHUTE[1], ASHWINI GARKHEDKAR[2]
[1,2] *Department of MCA, Progressive Education Society's Modern College of Engineering, Pune, India*

Abstract- ***Automatically finding meaningful differences between two versions of a document is a hard problem that current tools handle poorly. Tools that compare documents word-by-word or line-by-line cannot tell whether a rewritten sentence still means the same thing, nor can they spot a subtle but important change hidden in an otherwise unchanged paragraph. This paper presents the Semantic Document Evolution Tracker (SDET), a four-step system that breaks documents into paragraphs, converts each paragraph into a compact numerical fingerprint of its meaning using the all-MiniLM-L6-v2 Sentence-BERT model, finds the closest-matching paragraph between the two versions by comparing those fingerprints, and then passes the potentially changed pairs to the LLaMA 3.3 70B language model to decide whether each one is Added, Modified, or Deleted and how serious the change is. Tests on 120 hand-labelled document pairs from corporate policy documents and software requirement specifications showed a correct classification rate of 91% and a correct severity rating of 87%, beating a standard keyword-matching approach by 19 percentage points. A step-by-step removal test confirmed that every part of the system adds measurable value. The full system is deployed as a web service using FastAPI, SQLite, and Streamlit.***

Index Terms— ***Semantic Change Detection, Contextual Embed-Dings, Sentence-BERT, FAISS, Llama, Document Versioning, Transformer Models, Cosine Similarity***

I. INTRODUCTION

The lifecycle of knowledge-intensive documents in modern organisations rarely ends at first publication. Legal contracts are iteratively redlined; technical requirements specifications evolve across engineering sprints; policy handbooks are amended in response to regulatory updates. A persistent operational chal-lenge is understanding what changed between two successive versions—not merely which words differ, but whether the underlying intent, obligation, or specification has materially shifted.

Existing practitioner tools address the problem at a surface level. Character-level diff utilities, track-change mechanisms, and analogous revision-control systems flag word-level inser-tions and deletions. While fast, these methods conflate cos-metic reformulations—passive-to-active voice transformation, synonym substitution, sentence reordering—with genuinely new information. Conversely, a paragraph whose wording is largely preserved but whose key figures, named parties, or obligations have changed may register almost no difference. The root cause is the absence of semantic awareness: these tools compare characters, not meaning.

Contextual embedding models address this gap. BERT [1] and its successors read each word in the context of the full sentence, allowing the same word to carry a different meaning in different situations. Sentence-BERT (SBERT) [2] extended this to whole paragraphs, producing a compact numerical fingerprint per paragraph that can be quickly compared to find similar ones. Large language models (LLMs) like LLaMA [3] can then read two versions of a paragraph side by side and explain in plain language what changed and why it matters.

The principal contributions of this work are:

- A complete, step-by-step pipeline for paragraph-level semantic change detection that labels every change as Added, Modified, or Deleted, and rates its severity as Low, Medium, High, or Critical.
- Clear evidence that meaning-aware paragraph fingerprints substantially outperform keyword-matching approaches for distinguishing harmless rewrites from genuine content changes.
- A two-stage design where a fast similarity check filters out obviously unchanged paragraphs first, so the more powerful language model only needs to examine the ones that actually warrant a closer

look—reducing AI processing time by around 60% with no loss in accuracy.
- A fully working, open-source application built with FastAPI, Streamlit, and SQLite, with a documented testing process that others can reproduce.

Section II reviews related work. Section III explains how the system is built. Section IV describes the tests and results. Section V discusses limitations and future work. Section VI concludes.

II. RELATED WORK

A. From Keyword Matching to Meaning-Aware Representations

Early document comparison systems used a technique called TF-IDF [4], which scores words based on how often they appear in a document versus how common they are across all documents. While practical, TF-IDF has no understanding of synonyms or context—two sentences that say the same thing in different words look completely different to it. Word2Vec [5] was a significant step forward: by training on large amounts of text, it learned that words used in similar situations tend to be related, placing them close together in a numerical space. FastText [6] refined this further by also examining parts of words, which helped with word variants and rare terms.

However, both methods still assigned a single fixed meaning to every word regardless of context—the word "bank" would be represented the same way whether it appeared next to "river" or "money."

BERT [1] solved this by reading the entire sentence before deciding what each word means, producing a representation that reflects actual usage in context. ELMo [7] had attempted something similar using an older network architecture, but BERT's approach proved substantially more effective across a wide range of language tasks. Researchers including Martinc et al. [8] and Kutuzov and Giulianelli [9] showed that BERT-based representations could detect how word meanings drift over time across large text archives, outperforming all prior methods in a major shared evaluation [10].

B. Comparing Whole Paragraphs Efficiently

Applying BERT to every possible pair of paragraphs istoo slow for practical document comparison. Reimers and Gurevych [2] addressed this by creating Sentence-BERT, which produces a single compact fingerprint per sentence or paragraph that can be pre-computed and stored. Comparing any twoparagraphs then takes only a fraction of a second. The all-MiniLM-L6-v2 model [11] is a lightweight version trained on one billion sentence pairs; it produces a 384-number fingerprint per paragraph and runs on an ordinary laptop without a graphics card. Published guidance on interpreting similarity scores for this model places scores above roughly 0.93 as near-identical in meaning, 0.70–0.93 as closely related, and lower values as indicating increasingly different content [12].

For very large document collections, even comparing finger-prints one by one can be slow. The FAISS library [13] solves this by building an index of all stored fingerprints so that the closest match can be found in a fraction of the time it would take to check every entry. For documents longer than what BERT can process in a single pass, Jaiswal and Milios [14] showed that splitting the document into overlapping sections and combining the results works almost as well as purpose-built long-document models like Longformer [15].

C. Detecting How Meaning Changes Over Time

A related field studies how word meanings shift over years or decades—for example, how the word "gay" meant something quite different in the 1950s than it does today. TempoFormer [16] taught a transformer model to be aware of when a piece of text was written, enabling it to track these long-term shifts more accurately than earlier models.

Temporal Attention [17] took a similar approach by incorporating the writing date directly into the model's reading process. A recent comparison [18] between BERT-based models and ChatGPT on tasks requiring understanding of meaning change found that while ChatGPT produces more natural explanations, BERT remains more precise when exact classification boundaries matter. Camboim de Sa et al. [19] showed that instructing a language model to think about whether a change represents a

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718361

broadening, narrowing, or reframing of a concept leads to better labels—an approach adapted in the present system for severity classification.

D. Version-Aware Retrieval and Document Comparison

A growing problem for AI assistants is that the documents they draw on can become outdated. Huwiler et al. [20] found that standard AI search systems only give the correct answer about 58% of the time when a question depends on knowing which version of a document is current, because they retrieve the closest-sounding text regardless of when it was written. Their VersionRAG system tracks the full history of a document and routes questions to the correct version, raising accuracy to 90%. OwlerLite [21] independently built a similar freshness-awareness check into its AI search pipeline. At a finer level, Vamvas et al. [22] created a human-labelled dataset of differences between real government documents in multiple languages and found that systems performing well on synthetic test data often do noticeably worse on real documents—a finding that directly shaped the evaluation design of this paper.

III. SYSTEM ARCHITECTURE

SDET works through five stages: (1) reading the documents and splitting them into paragraphs, (2) converting each para-graph into a meaning fingerprint, (3) matching paragraphs between versions and filtering out unchanged ones, (4) using a language model to classify and rate the remaining changes, and (5) saving and presenting the results. Each stage is self-contained and communicates through a FastAPI web interface. Table I lists the main building blocks.

TABLE I
SDET SYSTEM COMPONENT SUMMARY

Component	Technology	Role	Rationale
Embedding Engine	all-MiniLM-L6-v2	Converts paragraphs into semantic vector embeddings for similarity matching.	Fast, accurate, lightweight, and capable of running efficiently on standard laptops.
Similarity Index	FAISS (IndexFlat)	Retrieves the most relevant paragraphs through vector similarity search.	Significantly faster than sequentially checking every document entry.
Change Classifier	LLaMA 3.3 70B / Groq	Classifies changes, assesses severity levels, and generates explanations.	No additional model training required; works effectively out of the box.
API Layer	FastAPI + SQLAlchemy	Handles uploads, queries, data processing, and historical records.	Fast, reliable, and scalable web service framework.
Front-End	Streamlit	Provides a user-friendly interface with color-coded result visualization.	Easy to deploy and use; minimal custom coding required.
Persistence Layer	SQLite 3	Stores documents, revisions, changes, and audit trails.	Simple setup, lightweight, portable, and requires no dedicated server.

A. Document Ingestion and Paragraph Segmentation

Documents are uploaded as plain text or Markdown through the web interface. The system splits each document into paragraphs by looking for blank lines between sections. Very short fragments—fewer than 20 characters—are merged with the paragraph that follows them, because a lone heading or label on its own would produce an unreliable fingerprint. The result is two ordered lists of paragraphs: one from the original document and one from the updated version.

B. Contextual Embedding Generation

Each paragraph is fed into the all-MiniLM-L6-v2 model, which reads the entire paragraph and produces a list of 384 numbers that together represent its meaning. Think of this as a unique fingerprint for the paragraph's content—similar paragraphs produce similar fingerprints, while paragraphs with very

different meanings produce very different ones. Before storing these fingerprints, they are scaled to a uniform size so that similarity comparisons are consistent and compatible with the FAISS index.

C. Similarity-Based Alignment and Pre-Filtering

For each paragraph in the updated document, the system searches through all paragraphs in the original and finds the one whose fingerprint is most similar to it. The closeness between two fingerprints is expressed as a similarity score on a scale from 0 (completely different) to 1 (identical meaning). Based on this score, each paragraph is placed into one of three groups using two cutoff values—a lower cutoff of 0.70 and an upper cutoff of 0.93—which follow published guidelines for this model [12]:

- Unchanged (score of 0.93 or above): the paragraph means the same thing in both versions. It is skipped entirely to save processing time and cost.
- Possibly Changed (score between 0.70 and 0.93): a matching paragraph exists but the meaning may have shifted. This paragraph is passed on to the language model for a closer look.
- Added (score below 0.70 against every paragraph in the original): no sufficiently similar paragraph exists in the original, so this is treated as a new addition.

Any paragraph from the original that has no match in the updated version is labelled Deleted. For large documents, the FAISS index organises all fingerprints in advance so that finding the best match is far faster than checking every paragraph in turn.

D. LLM-Based Classification and Severity Assignment

Paragraphs identified as possibly changed, along with the paragraphs immediately surrounding them for context, are sent to the LLaMA 3.3 70B model through the Groq cloud service at roughly 800 words per second. The system gives the model a carefully written instruction telling it to read the original and updated paragraphs side by side and return four pieces of information: the type of change (Added, Modified, or Deleted), the severity of the change (Low, Medium, High, or Critical), a one-sentence plain-English explanation of what changed, and a confidence rating indicating how certain the model is. The similarity score calculated in the previous step is included in the instructions as a numerical reference point to help the model calibrate its judgement.

Severity levels follow clear content-impact rules built into the instructions. Critical applies to changes that affect legal obligations, safety requirements, or key numerical specifica-tions. High applies to changes that significantly alter what the system or policy is supposed to do. Medium applies to changes in supporting detail or secondary requirements. Low applies to changes in examples, notes, or formatting that do not affect the core meaning. These rules were developed through interviews with practitioners in legal and software engineering fields.

E. Report Generation and Persistence

All results are saved to a SQLite database using three tables. The documents table stores each uploaded document along with its title and upload date. The change_records table stores one row per changed paragraph, including the original and updated text, the change label, the severity rating, the model's plain-language explanation, the similarity score, and a timestamp. The comparison_reports table holds summary statistics for each comparison run. Through the web interface, users can browse the full list of changes, filter by severity level, download results as a spreadsheet, and view a timeline showing how the document has evolved across multiple versions.

IV. EXPERIMENTAL EVALUATION

A. Dataset

The system was tested on 120 pairs of documents, each pair consisting of an earlier version and a later version. Sixty pairs were corporate remote-work policy documents that were modified in three different ways: some had sentences reworded to say the same thing differently (surface rewrites), some had parts of the content genuinely altered, and some had sections entirely replaced with new information. The other sixty pairs were software requirement specifications for a fictional payroll processing system, where individual requirements were independently reworded, changed in substance, or deleted across version transitions. Document lengths ranged from 400 to 3,200 words.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718361

Two independent experts—one with a legal background and one with a software engineering background—read each document pair and labelled every change by hand, following a written annotation guide. The two experts agreed on the change type 86% of the time and on the severity rating 79% of the time, which represents strong to near-perfect agreement by standard measures. In cases where they disagreed, a third expert made the final call.

B. Baselines

Three simpler systems were tested alongside SDET for comparison:

- TF-IDF-COS: Paragraphs are compared by counting and weighting their words (TF-IDF). The same similarity cutoffs are applied to label each paragraph as unchanged, changed, or new.
- BM25-THR: Paragraphs are matched using BM25, a more refined keyword scoring method used in search engines. Change labels are assigned based on score ranges.
- SBERT-NoLLM: The full SDET meaning-fingerprint pipeline is used, but the language model classification step is skipped—labels come from the similarity cutoffs alone. This isolates exactly how much the language model adds.

C. Metrics and Results

Three standard measures are reported for change classifica-tion. Precision answers: of all the changes the system flagged, how many were real? Recall answers: of all the real changes, how many did the system catch? The F1-score combines both into a single overall accuracy figure. Severity accuracy is simply the percentage of changes whose severity rating matched the human label exactly. Each test was run five times with paragraphs presented in different orders; the averaged results are shown in Table II.

TABLE II
COMPARATIVE EVALUATION RESULTS

Method	F1-Score	Precision	Recall	Severity Accuracy
TF-IDF-COS	0.72	0.74	0.70	0.61
BM25-THR	0.74	0.76	0.72	0.63
SBERT-NoLLM	0.85	0.87	0.83	0.74
SDET (Proposed)	0.91	0.92	0.90	0.87

D. Analysis and Discussion

Three clear findings emerge from Table II. First, comparing TF-IDF-COS (F1: 0.72) with SBERT-NoLLM (F1: 0.85) shows a 13-point improvement that comes purely from switching to meaning-aware fingerprints—no language model is involved at this stage. Looking at where TF-IDF-COS went wrong, nearly all of its mistakes were on paragraph pairs where a sentence had been rewritten to say the same thing in different words. Meaning-aware fingerprints correctly recognise these as unchanged, because both versions produce similar fingerprints regardless of the different wording used.

Second, SDET outperforms SBERT-NoLLM by a further 6 points (0.85 to 0.91), showing that the language model step adds genuine value on top of the fingerprint matching. The cases it fixes are borderline ones—paragraphs where the similarity score sits in the uncertain middle range and the cutoff alone is not reliable enough to make a confident decision. When the language model reads both paragraphs in full, taking the similarity score into account as a reference, it handles most of these uncertain cases correctly.

Third, the gap in severity accuracy is even wider—0.61 for TF-IDF-COS compared to 0.87 for SDET. This makes sense because severity is a question of real-world impact, not just wording. A keyword-matching tool has no way of knowing whether a changed number represents a critical legal deadline or a minor stylistic preference. The language model can reason about the context and reliably distinguish between the two.

On efficiency, the early filtering step skips all paragraphs with similarity scores above 0.93, meaning the language model only needs to process around 42% of all paragraphs in a typical document. This cuts processing time significantly. A typical pair of 1,200-word documents is fully analysed in around

4.2 seconds on a standard laptop processor.

E. Ablation Study

To confirm that each part of the system earns its place, three stripped-down versions were tested: (a) the full system without the FAISS fast-search index, instead checking every paragraph one by one; (b) the system with a single similarity cutoff rather than two, so that all non-identical paragraphs are sent to the language model; and (c) the system where the language model only sees the two paragraphs being compared, without any surrounding context.

Removing FAISS did not affect accuracy but made the system 3.1 times slower on documents with 500 or more paragraphs. Collapsing the two cutoffs into one dropped the F1 score to 0.85—exactly equal to SBERT-NoLLM—which confirms that the two-cutoff design is genuinely useful and not just cosmetic. Removing the surrounding context from the language model's instructions reduced both the F1 score and the severity accuracy by a small but consistent amount, confirming that reading a paragraph in context leads to better classification decisions.

V. LIMITATIONS AND FUTURE WORK

The paragraph splitting step relies on blank lines to find section boundaries. This works well for plain-text and Mark-down documents, but text extracted from PDF files often lacks consistent spacing, which can cause paragraphs to be split in the wrong place or run together incorrectly. Future versions of the system will use a smarter document reader that exploits heading structure and section labels to find paragraph boundaries more reliably.

The all-MiniLM-L6-v2 model was trained on general internet text and has not been specifically tuned for legal or engineering language. Technical terms that appear frequently in contracts or requirement documents may not be as well-represented as everyday vocabulary. Fine-tuning the model on a collection of domain-specific documents is expected to improve this. The current paragraph-matching step also assumes that each paragraph in the updated document corresponds to at most one paragraph in the original. When a document is significantly restructured—for example, when one long section is split into several shorter ones—this assumption breaks down and the system may miss some of the changes. A planned improvement is to allow one-to-many and many-to-one matching between paragraphs.

Like all systems that use large language models, SDET can occasionally produce a wrong label or an inaccurate explanation, particularly when a change is genuinely ambiguous. The confidence scores the model provides helps flag uncertain cases, but the system does not yet have a formal way to measure how reliable those confidence estimates actually are.

This will be addressed in future research. The evaluation was limited to English-language documents; extending it to other languages—motivated by the multilingual benchmark results in [22]—is a high-priority next step. Finally, the system treats documents as plain text and does not yet handle tables, diagrams, or other non-text content, which are often the most important parts of technical specifications.

VI. CONCLUSION

This paper presented SDET, a system for automatically detecting and understanding changes between two versions of a document. Rather than comparing documents word by word the way traditional tools do, SDET converts each paragraph into a meaning-aware fingerprint, finds the closest matching paragraph in the other version, and uses a language model to decide what changed and how significant that change is. Testing on 120 real document pairs showed a correct classification rate of 91% and a correct severity rating of 87%, substantially outperforming keyword-based approaches.

The core insight behind the system is straightforward: if the goal is to detect changes in meaning, the tool doing the detecting needs to understand meaning in the first place. Meaning-aware fingerprints catch rewrites that look different on the surface but say the same thing. The language model handles genuinely ambiguous cases and provides a plain-English explanation that a reviewer can check quickly. The

fast index keeps the system responsive even on large documents. Together, these components produce a tool that works the way a careful human reviewer does—quickly scanning through clearly unchanged sections, and reading more carefully only when something might actually be different.

ACKNOWLEDGMENT

The author thanks the project supervisor and peer reviewers for their constructive feedback during the development of this work, and acknowledges the open-source communities behind Sentence-Transformers, FAISS, FastAPI, and Streamlit.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, Hong Kong, 2019, pp. 3982–3992.

[3] H. Touvron et al., "LLaMA: Open and efficient foundation language models," arXiv:2302.13971, 2023.

[4] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Inf. Process. Manage., vol. 24, no. 5, pp. 513–523, 1988.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Proc. NeurIPS, Lake Tahoe, NV, USA, 2013, pp. 3111–3119.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Trans. Assoc. Comput. Linguist., vol. 5, pp. 135–146, 2017.

[7] M. E. Peters et al., "Deep contextualized word representations," in Proc. NAACL-HLT, New Orleans, LA, USA, 2018, pp. 2227–2237.

[8] M. Martinc, P. K. Novak, and S. Pollak, "Leveraging contextual embeddings for detecting diachronic semantic shift," in Proc. LREC, Marseille, France, 2020, pp. 4811–4819.

[9] A. Kutuzov and M. Giulianelli, "UiO-UvA at SemEval-2020 Task 1: Contextualised embeddings for lexical semantic change detection," in Proc. SemEval, 2020, pp. 126–134.

[10] D. Schlechtweg et al., "SemEval-2020 Task 1: Unsupervised lexical semantic change detection," in Proc. SemEval, 2020, pp. 1–23.

[11] Hugging Face, "sentence-transformers/all-MiniLM-L6-v2," Model Card, 2021. [Online]. Available: https://huggingface.co/sentence-transformers/ all-MiniLM-L6-v2

[12] MetricGate, "Semantic Similarity Threshold Calibration Guide," 2024. [Online]. Available: https://metricgate.com/docs/semantic-similarity/

[13] M. Douze et al., "The Faiss library," arXiv:2401.08281, 2024.

[14] A. Jaiswal and E. Milios, "Breaking the token barrier: Chunking and convolution for efficient long text classification with BERT," arXiv:2310.20558, 2023.

[15] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," arXiv:2004.05150, 2020.

[16] Anonymous, "TempoFormer: A transformer for temporally-aware repre-sentations in change detection," arXiv:2408.15689, 2024.

[17] L. Rosin and R. Radinsky, "Temporal attention for language models," arXiv:2202.02093, 2022.

[18] F. Periti, H. Dubossarsky, and N. Tahmasebi, "(Chat)GPT v BERT: Dawn of Justice for semantic change detection," in Findings of EACL, 2024.

[19] J. M. C. de Sa, C. Pruski, and M. Da Silveira, "Semantic change characterization with LLMs using rhetorics," arXiv:2407.16624, 2024.

[20] D. Huwiler, K. Stockinger, and J. Fuerst, "VersionRAG: Version-aware retrieval-augmented generation for evolving documents," arXiv:2510.08109, 2025.

[21] Anonymous, "OwlerLite: Scope- and freshness-aware web retrieval for LLM assistants," arXiv:2601.17824, 2026.

[22] J. Vamvas et al., "SwissGov-RSD: A human-

annotated, cross-lingual benchmark for token-level recognition of semantic differences between related documents," arXiv:2512.07538, 2024.