# CIS5300 Milestone 2: Evaluation Metrics and Baselines
# Project: Context-Aware Legal Information Retrieval

John Harshith Kavuturu (67055516) Navya Saxena (52078368)
Aniket Ghorpade (52552619) Pavithra Manikandan (38819531)

November 18, 2025

## 1 Introduction

This report describes the evaluation metrics and baseline systems implemented for the legal passage retrieval task. The goal is to retrieve relevant legal passages from a corpus given a natural language query, as part of the LegalBench-RAG benchmark.

## 2 Evaluation Metrics

We evaluate our retrieval system using four metrics:
Exact Match, Span F1, Recall@10, and nDCG@10.

### 2.1 Exact Match

Exact Match (EM) measures whether the top retrieved passage exactly matches any gold standard answer (case-insensitive):

$$\text{EM} = \begin{cases} 1 & \text{if predicted\_text} \equiv \text{gold\_text} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The final score is the average EM over all test queries.

### 2.2 Span F1

Span F1 measures token-level overlap between predicted and gold passages. We compute:

$$\text{Precision} = \frac{|\text{pred\_tokens} \cap \text{gold\_tokens}|}{|\text{pred\_tokens}|} \tag{2}$$

$$\text{Recall} = \frac{|\text{pred\_tokens} \cap \text{gold\_tokens}|}{|\text{gold\_tokens}|} \tag{3}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

For queries with multiple gold answers, we use the maximum F1 score.

### 2.3 Recall@K

Recall@K measures the fraction of gold answer passages found in the top-K retrieved passages:

$$\text{Recall@K} = \frac{|\{\text{gold\_passages}\} \cap \{\text{top\_k\_retrieved}\}|}{|\{\text{gold\_passages}\}|} \tag{5}$$

We use K=10 for our evaluation.

### 2.4 Normalized Discounted Cumulative Gain (nDCG@K)

nDCG@K measures ranking quality by considering the position of relevant items:

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{\text{rel}_i}{\log_2(i+1)} \tag{6}$$

$$\text{IDCG@K} = \sum_{i=1}^{\min(\text{num\_relevant},K)} \frac{1}{\log_2(i+1)} \tag{7}$$

$$\text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \tag{8}$$

where $\text{rel}_i = 1$ if passage at position $i$ is relevant (matches a gold answer), 0 otherwise.

## 3 Simple Baseline: TF-IDF Retrieval

### 3.1 Approach

The simple baseline uses TF-IDF (Term Frequency-Inverse Document Frequency) with cosine similarity. TF-IDF weights terms by their frequency in a document and inverse frequency across the corpus:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{9}$$

where:

$$\text{TF}(t, d) = \frac{\text{count of } t \text{ in } d}{\text{total terms in } d} \tag{10}$$

$$\text{IDF}(t, D) = \log \frac{|\text{documents in } D|}{|\text{documents containing } t|} \tag{11}$$

We compute cosine similarity between query and passage TF-IDF vectors to rank passages.

### 3.2 Implementation

- Max features: 5,000
- N-grams: unigrams and bigrams (1-2 word sequences)
- Stopwords: English stopwords removed
- Text preprocessing: lowercasing, whitespace normalization

# 4 Strong Baseline: BM25 Retrieval

## 4.1 Approach

BM25 (Best Matching 25) is a probabilistic ranking function that improves upon TF-IDF:

$$\text{BM25}(Q, D) = \sum_{q_i \in Q} \text{IDF}(q_i) \times \frac{f(q_i, D) \times (k_1 + 1)}{f(q_i, D) + k_1 \times (1 - b + b \times \frac{|D|}{\text{avgdl}})} \tag{12}$$

where:

- $f(q_i, D)$: frequency of term $q_i$ in document $D$
- $|D|$: document length (in words)
- avgdl: average document length in corpus
- $k_1 = 1.5$: term frequency saturation parameter
- $b = 0.75$: length normalization parameter

BM25 addresses limitations of TF-IDF through term frequency saturation and better length normalization.

## 4.2 Implementation

- Parameters: $k_1 = 1.5$, $b = 0.75$ (standard values)
- Tokenization: NLTK word tokenizer with stopword removal
- Preprocessing: lowercasing, alphanumeric token filtering

# 5 Experimental Results

## 5.1 Dataset

We evaluate on the LegalBench-RAG benchmark:

- Test set: ContractNLI subset (977 queries)
- Corpus: Legal contract documents
- Evaluation: First 100 test queries (for initial results)

## 5.2 Baseline Performance

## 5.3 Observations

- BM25 outperforms TF-IDF across all metrics: Span F1 (0.2217 vs 0.2009), Recall@10 (0.5267 vs 0.3717), and nDCG@10 (0.4544 vs 0.2757). This confirms that BM25's term frequency saturation and length normalization provide better ranking for passage retrieval.
- Both baselines achieve 0.0000 Exact Match, indicating that exact string matching is rare in this task. This is expected given that retrieved passages are text chunks that may not exactly match the gold answer spans.

| Metric | TF-IDF | BM25 |
|---|---|---|
| Exact Match | 0.0000 | 0.0000 |
| Span F1 | 0.2009 | 0.2217 |
| Recall@10 | 0.3717 | 0.5267 |
| nDCG@10 | 0.2757 | 0.4544 |

Table 1: Baseline performance on LegalBench-RAG test set (first 100 queries from ContractNLI subset). The corpus consists of 563 passages extracted from legal contract documents.

- BM25 shows a 41.7% relative improvement in Recall@10 over TF-IDF (0.5267 vs 0.3717), demonstrating its effectiveness at retrieving relevant passages in the top-10 results.
- The nDCG@10 scores (0.4544 for BM25, 0.2757 for TF-IDF) indicate that BM25 not only retrieves more relevant passages but also ranks them better, with relevant items appearing earlier in the ranked list.
- Both baselines rely on lexical matching and do not capture semantic relationships, which presents an opportunity for improvement through semantic embeddings or neural retrieval models.

## 6 Conclusion

We have implemented evaluation metrics (Exact Match, Span F1, Recall@10, nDCG@10) and two baseline retrieval systems (TF-IDF and BM25) for the legal passage retrieval task. On the LegalBench-RAG ContractNLI subset (first 100 queries), BM25 outperforms TF-IDF with a Recall@10 of 0.5267 (vs 0.3717) and nDCG@10 of 0.4544 (vs 0.2757), demonstrating its effectiveness for lexical retrieval. However, both baselines achieve 0.0000 Exact Match, indicating the difficulty of exact string matching in this task. The BM25 baseline serves as a strong starting point for future improvements, which may include semantic embeddings, neural retrieval models, or hybrid approaches combining lexical and semantic matching to further improve retrieval performance.

## References


1. Rajpurkar, P., et al. (2016). SQuAD: 100,000+ Questions for Machine Reading Comprehension. *arXiv:1606.05250*.
2. Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333-389.
3. Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM TOIS*, 20(4), 422-446.


## Appendix: Sentence-BERT Baseline

**Sentence-BERT overview:** Sentence-BERT (SBERT) is a bi-encoder architecture that maps sentences or passages into a shared embedding space, where semantic similarity can be computed

efficiently via vector similarity. Formally, for a query $q$ and a passage $p$, it produces dense vectors

$$\mathbf{s}_q = f_{\text{SBERT}}(q) \in \mathbb{R}^d, \qquad \mathbf{s}_p = f_{\text{SBERT}}(p) \in \mathbb{R}^d,$$

and uses a similarity function such as cosine similarity

$$\text{score}(q, p) = \cos(\mathbf{s}_q, \mathbf{s}_p) = \frac{\mathbf{s}_q^\top \mathbf{s}_p}{\|\mathbf{s}_q\| \, \|\mathbf{s}_p\|}$$

At indexing time, we precompute embeddings $\{\mathbf{s}_p\}$ for all passages; at query time, we encode the query $q$ once and retrieve the top-$k$ passages by their similarity scores. In our project, we use a pretrained SBERT-style checkpoint based on the `sentence-transformers/all-mpnet-base-v2` encoder to obtain passage embeddings, and retrieve passages using dot-product/cosine similarity.

**Sentence-BERT baseline results.** We ran Sentence-BERT as a dense retrieval baseline and obtained the following development-set results: This SBERT baseline performs better than our TF-IDF baseline but does not yet match the effectiveness of BM25 on this task. The implementation of this dense retrieval setup is in the notebook `NLP_Milestone_2.1.ipynb`.

```
Sentence-BERT Baseline Results:
  exact_match: 0.0000 and span_f1: 0.2147
  recall@10: 0.4317 and ndcg@10: 0.3232
  num_examples: 100.0000
```

**Why SBERT underperforms BM25 here.** Firstly, we used the generic `sentence-transformers/all-mpnet-base-v2` encoder without any legal-domain fine-tuning or contrastive training on ContractNLI passages. This checkpoint is very good at everyday paraphrase detection, but it often misses the subtle legal terminology and rare tokens that BM25 can exploit via exact lexical matches. Secondly, in the notebook we only index the first 10000 chunks, and after filtering we end up with only about 563 actual chunks; dense retrieval typically shines when trained and evaluated over much larger collections with many hard negatives, so in this relatively small and truncated passage set the resulting dot-product scores are noisy and simple lexical overlap from BM25 still wins. Thirdly, our exact-span evaluation protocol inherently favors lexical matches: the metrics reward retrieving the precise gold span, so BM25 benefits from directly matching rare legal tokens that appear in both the query and the gold clause, whereas SBERT often retrieves semantically similar but not lexically identical clauses, which depresses Exact Match, Recall@10, and nDCG@10 even when the retrieved passage is conceptually on-topic. To make SBERT consistently competitive with or better than BM25, we would need to (i) use a legal-domain SBERT checkpoint or fine-tune the encoder on our training CSV (for example, with contrastive learning on ContractNLI question-span pairs), (ii) increase chunk size and coverage so that dense vectors see complete legal clauses instead of heavily truncated snippets, (iii) pair the SBERT bi-encoder with a cross-encoder re-ranker that can re-score the top-$k$ candidates and push the exact gold spans toward the top of the ranking, and (iv) scale to an approximate nearest neighbor (ANN) index such as FAISS so that we can index the entire corpus without the 10 k chunk limit and fully exploit dense retrieval at larger scale.

**Next steps and code.** All further work on improving dense and hybrid retrieval (and aiming for much better results) is planned for Milestone 3. You can check all of the work done so far, including all the baseline implementations and related notebooks/scripts, in our project repository:

https://github.com/JohnHarshith/CIS5300-F2025-Project