Contents

Systematic Review Screening: Summary & Methodology Justification

1. Methodological Justification

Why TF-IDF with (1,3) n-grams?

We follow the design choices found in several foundational papers:

> *"We construct a ranker by extracting bag-of-n-grams (n 3) over words in the titles and abstracts. We use both tf-idf scores and binary features"* — Norman et al., *Automating Document Discovery*

- **LREC 2020** shows a ~**10 percentage point F boost** from using trigrams vs. unigrams.
- **Norman et al. (L18-1582)** use TF-IDF with up to trigrams (n 3) as standard in their baseline system, alongside metadata and binary indicators.
- **Cohen et al. (2006)** extract lexical features with bag-of-n-grams up to n=3 and use them in SVM and logistic regression ranking tasks.

Therefore, we use `ngram_range=(1,3)` as our **default baseline**, but also test `(1,2)` to validate whether this improvement generalizes to smaller datasets.

---

2. Experimental Setup

We conduct a **grid search** over both TF-IDF and classifier hyperparameters. For each classifier (LogReg, SVM, CNB, Cosine), we extract:

- A **balanced model** (maximizing F1)

- A **high-recall model** (recall 0.95)

We use:

```
scoring={'f1': 'f1', 'recall': 'recall'}
refit='f1'
```

This allows us to compare both performance curves and threshold trade-offs.

---

3. N-gram Range: Experimental Results Summary

Classifier	Avg F1 (1,2)	Avg F1 (1,3)	Winner	Delta
Logistic Reg.	0.5232	0.5207	(1,2)	+0.2 pp
SVM	0.5309	0.5306	(1,2)	+0.0 pp
Comp. NB	0.2267	0.2383	(1,3)	+1.2 pp
Cosine Sim.	0.2765	0.2605	(1,2)	+1.6 pp

Takeaway: Despite trigrams being advocated in literature, our **smaller dataset (~2.1k)** shows **minimal or no improvement** from using (1,3). In fact, (1,2) yields **slightly better results overall**, especially for cosine and logreg models.

---

4. Classifier-Specific Findings

Classifier Performance Summary

Classifier	Mode	Precision	Recall	F1	F2	AUC	WSS@95
LogReg	Balanced	0.5667	0.7083	0.6296	0.6746	0.9476	0.8124
	High-Recall	0.3067	0.9583	0.4646	0.6725	0.9296	0.6060
SVM	Balanced	0.5714	0.8333	0.6780	0.7634	0.9565	0.7894
	High-Recall	0.3067	0.9583	0.4646	0.6725	0.9285	0.6060
CNB	Balanced	0.3333	0.9167	0.4889	0.6790	0.9223	0.6472
	High-Recall	0.3151	0.9583	0.4742	0.6805	0.9300	0.6151
Cosine	Balanced	0.5161	0.6667	0.5818	0.6299	0.9225	0.8078
	High-Recall	0.3194	0.9583	0.4792	0.6845	0.9246	0.6197

Logistic Regression

Full Report: Logistic Regression

- **Balanced F1**: 0.6296 | AUC: 0.9476 | WSS@95: 0.8124
- **High-Recall (recall=95.83%)**: F1 = 0.4646 | Precision = 0.3067

ROC Curve – Logistic Regression (Balanced)

ROC Curve – Logistic Regression (High Recall)

SVM

Full Report: SVM

- **Balanced F1**: **0.6780** | AUC: **0.9565** | Precision = **0.5714**
- **High-Recall**: Slightly worse AUC than LogReg but competitive F1.

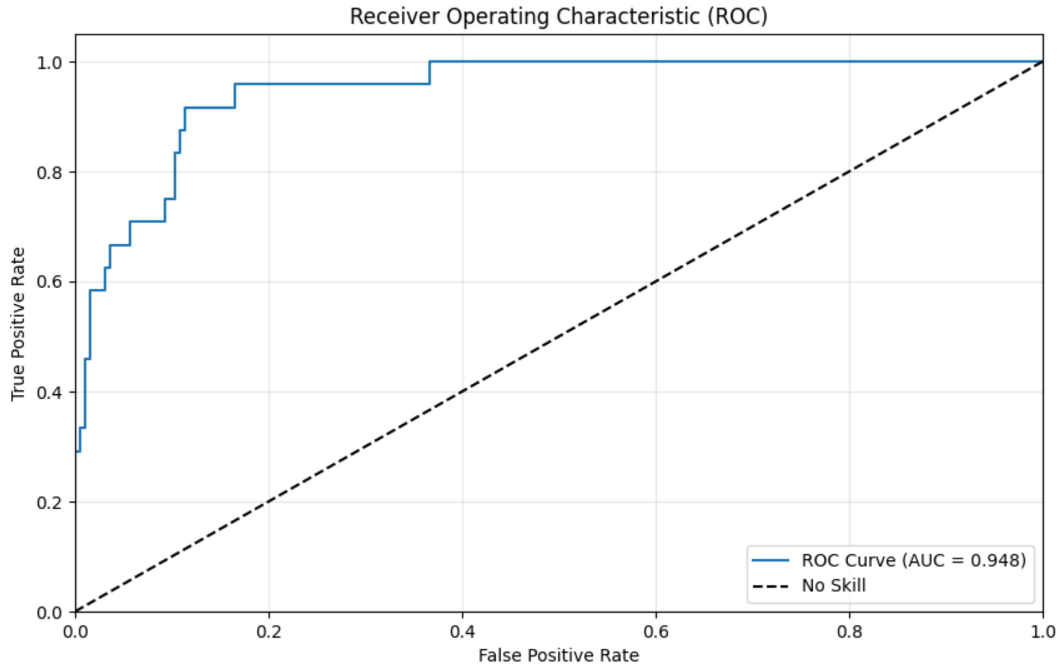

Figure 1: LogReg ROC Balanced

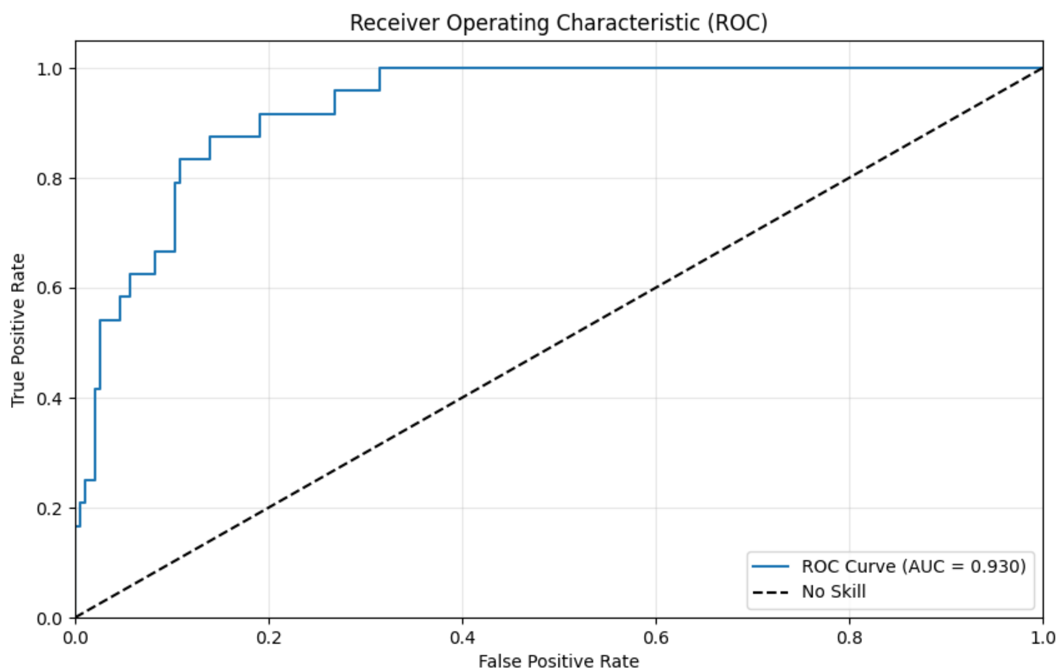

Figure 2: LogReg ROC High Recall

ROC Curve – SVM (Balanced)

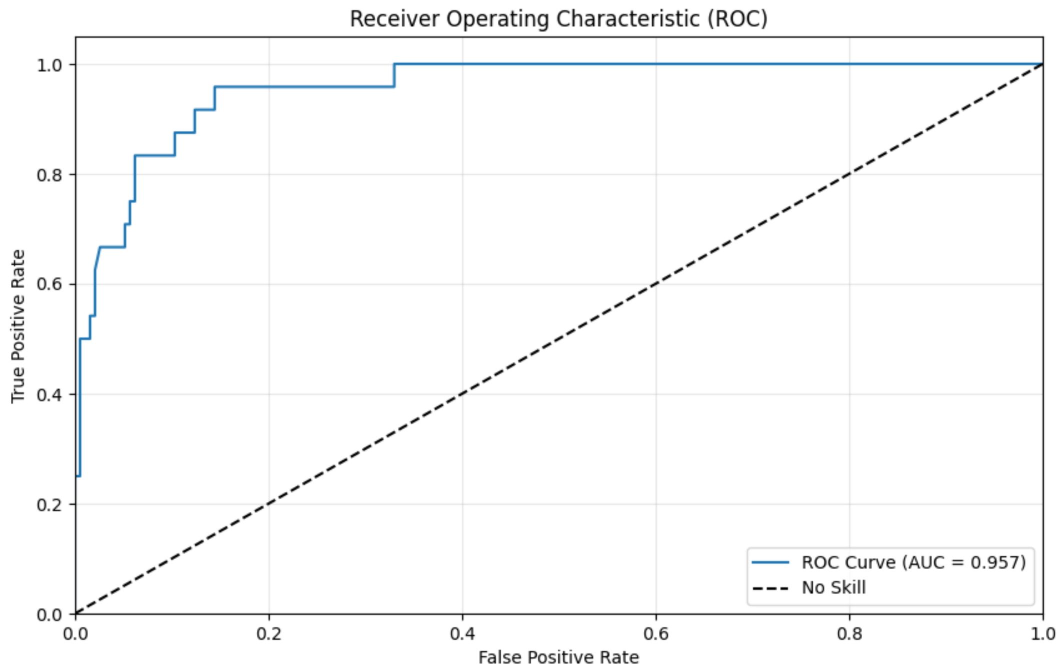

Figure 3: SVM ROC Balanced

ROC Curve – SVM (High Recall)

Complement Naive Bayes

Full Report: CNB

- Performs weaker overall. High-recall F1 = 0.4742. Slight improvement from trigrams

ROC Curve – CNB (Balanced)

ROC Curve - CNB (High Recall)

Cosine Similarity

Full Report: Cosine Similarity

- **Surprisingly effective in high-recall**: best F1 @ recall=95% (0.4792) despite weak absolute metrics
- Most relevant documents are similar to each other in language, but cosine also pulls in too many false positives.
  - Cohen et al. (2006) shows that unsupervised heuristics like this can work surprisingly well, especially when recall is the priority.
  - Frunza also used cosine in pre-filtering before training.

ROC Curve – Cosine Similarity (Balanced)

ROC Curve – Cosine Similarity (High Recall)

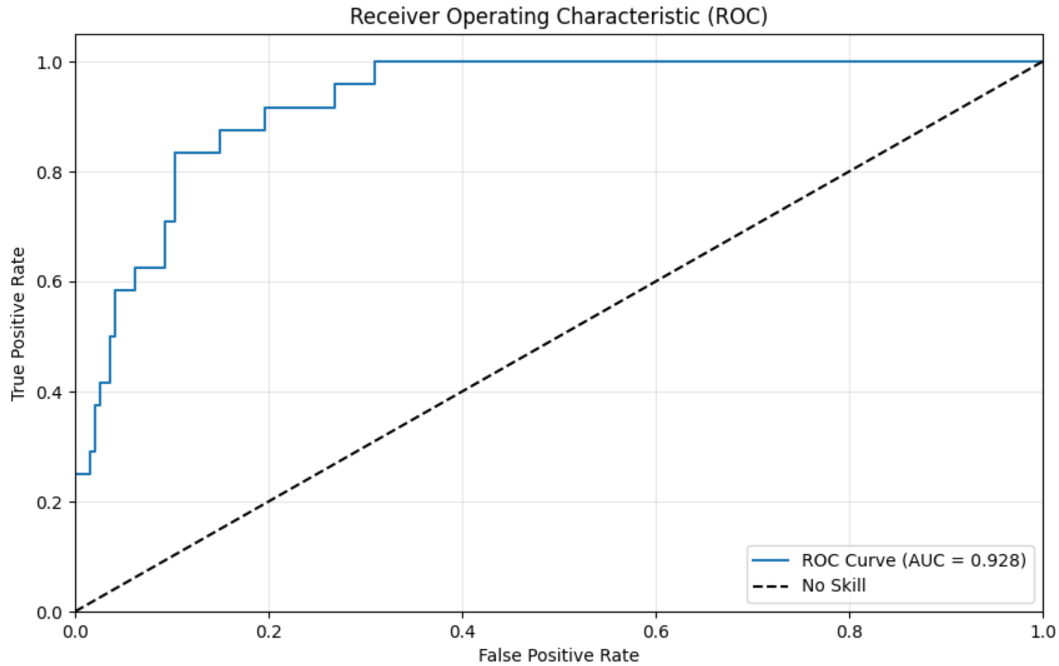

Figure 4: SVM ROC High Recall

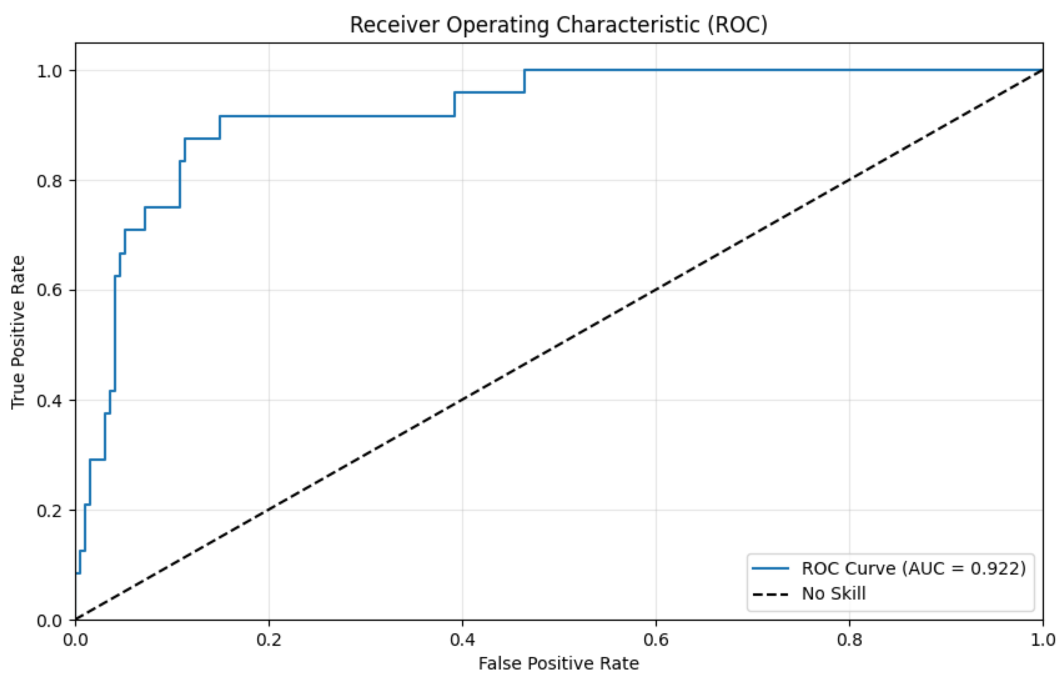

Figure 5: CNB ROC Balanced

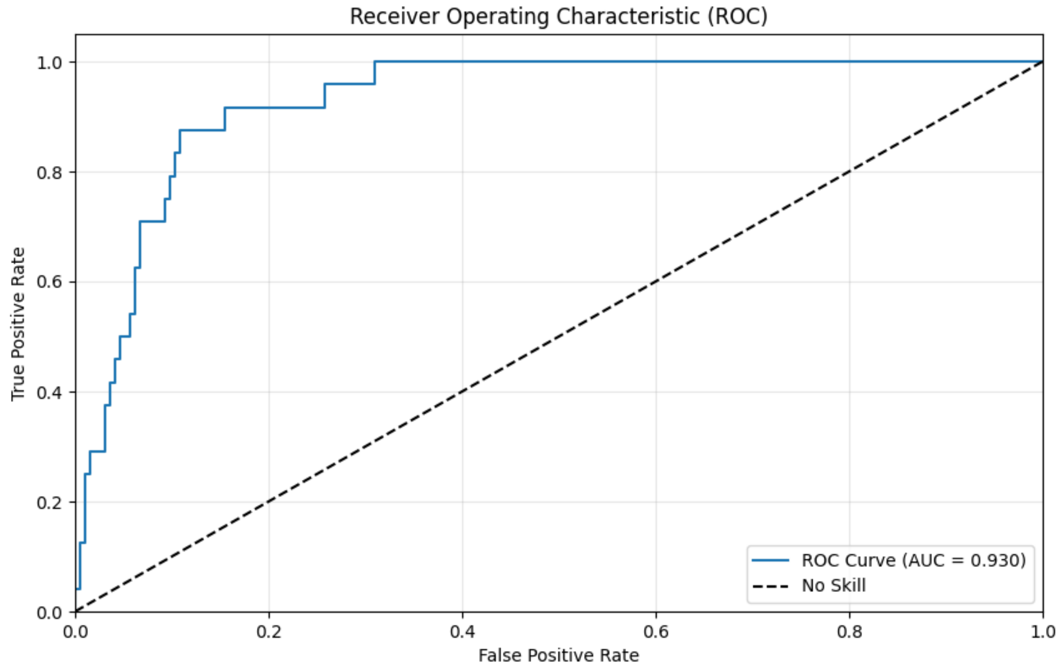

Figure 6: CNB ROC High Recall

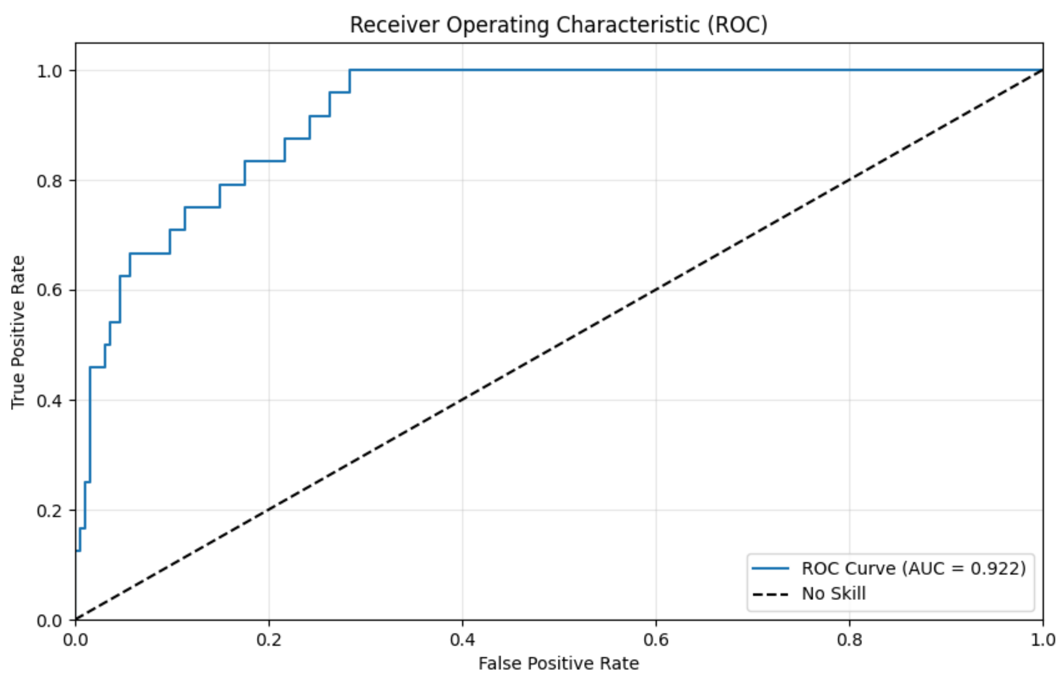

Figure 7: Cosine ROC Balanced

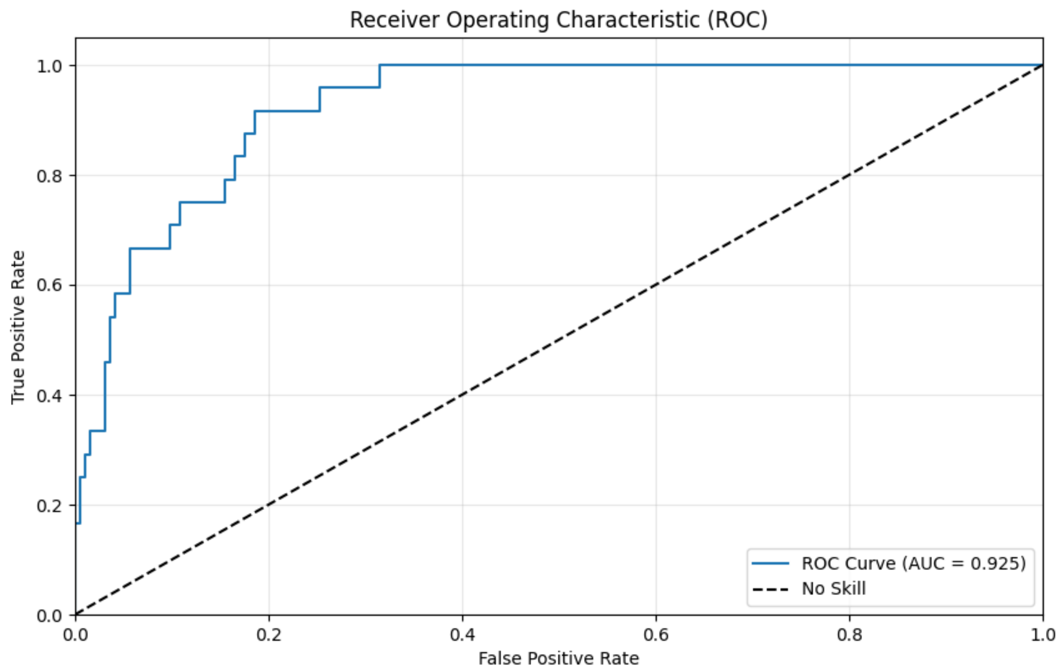

Figure 8: Cosine ROC High Recall

5. Justification for Metric Choice

- While F was initially used, it caused unstable thresholding and deviated from **field norms**.
- **EMNLP 2020** uses **micro-F** for tuning.
- **Norman et al. (L18-1582)** and **Cohen et al. (2006, 2009)** report metrics like F, AUC, and WSS@95, but **do not optimize for F**.

We therefore choose F as the primary tuning metric, and report WSS@95 to align with domain expectations.

---

6. Final Configurations (as of Current Grid Search)

Model	N-gram	Recall	F1	AUC	Notes
LogReg (Balanced)	(1,3)	0.7083	0.6296	0.9476	Baseline
LogReg (Recall@95)	(1,2)	0.9583	0.4646	0.9296	Best high-recall
SVM (Balanced)	(1,2)	0.8333	**0.6780**	**0.9565**	Best overall F1
CNB (Recall@95)	(1,3)	0.9583	0.4742	0.9300	Trigram marginal gain
Cosine (Recall@95)	(1,3)	0.9583	**0.4792**	0.9246	Best recall-optimized

---

Classifier Usage in Reference Papers

Paper	SVM Used	Logistic Regression Used	Notes
Cohen et al. (2006)	Yes	Yes	Compared both for ranking performance
Cohen et al. (2009)	Yes	No explicit mention	Used SVM-Light for cross-topic ranker
Frunza et al. (2010)	Yes	No	Reported SVM as best-performing classifier
Norman et al. (L18-1582)	Yes	Yes	Used standard and active-learning variants of logistic regression
LREC 2020 (Rezapour)	Not specified	Not specified	Focused more on annotation design; no classifier was specified
EMNLP 2020	Yes	Yes	Both used as baselines; SVM performed slightly better than LR (F1: 83.4 vs 81.4)

Justification Summary for Classifier Use

Across our reference corpus, **SVMs and logistic regression are the two most commonly used traditional classifiers**.

- **SVM** was used in **5 out of 6 papers**, typically cited for its robustness in sparse, high-dimensional settings (Cohen 2006, 2009; Norman 2018; Frunza 2010; EMNLP 2020).
- **Logistic regression** also appears in **3 of those 6**, often used with or without active learning (Norman et al., EMNLP 2020, Cohen 2006).

Norman et al. explicitly tested logistic regression variants and found it competitive depending on dataset characteristics . Similarly, EMNLP 2020 observed that logistic regression achieved an F1 of 81.4, only slightly behind SVM's 83.4 .

7. Next Steps

- Use **SVM (1,2)** for high-F1 filtering
- Use **Cosine Similarity (1,3)** for recall-sensitive screening
- Use **LogReg** for explainability and ranking flexibility
- Explore **custom token filters**, **regex patterns**, and **balancing** in next phase
- Consider model ensembling or re-ranking (e.g., LogReg followed by Cosine) as suggested in Cohen et al.
- Per question classifier suggested in Frunza