Screening Prioritisation

Aaron Fletcher

November 25, 2024

Topics

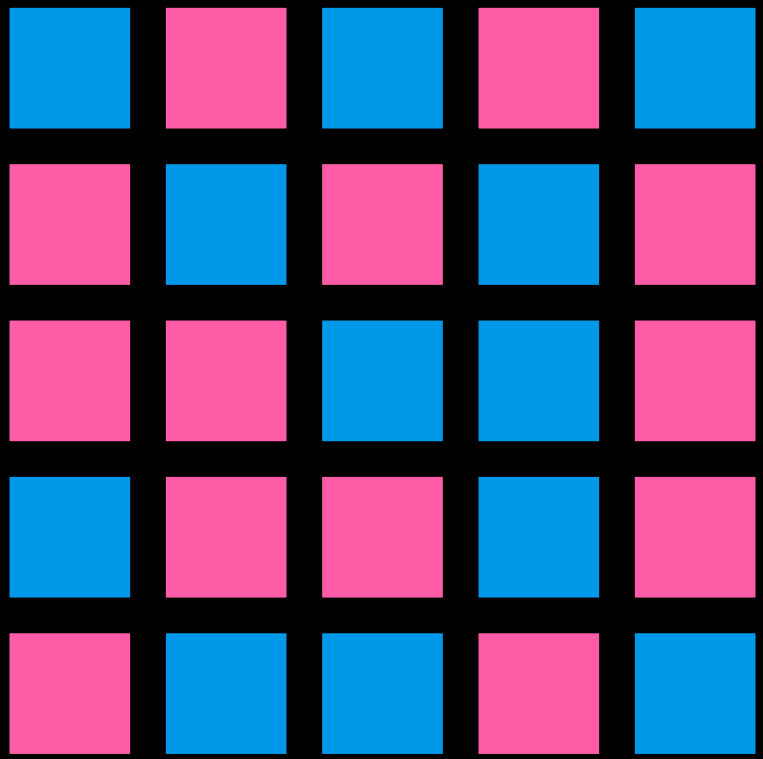
- Overview of screening prioritisation
- Active learning
- Implementations of active learning for screening

1. Screening

Looking at a collection of items
and determining each item's relevance to a question

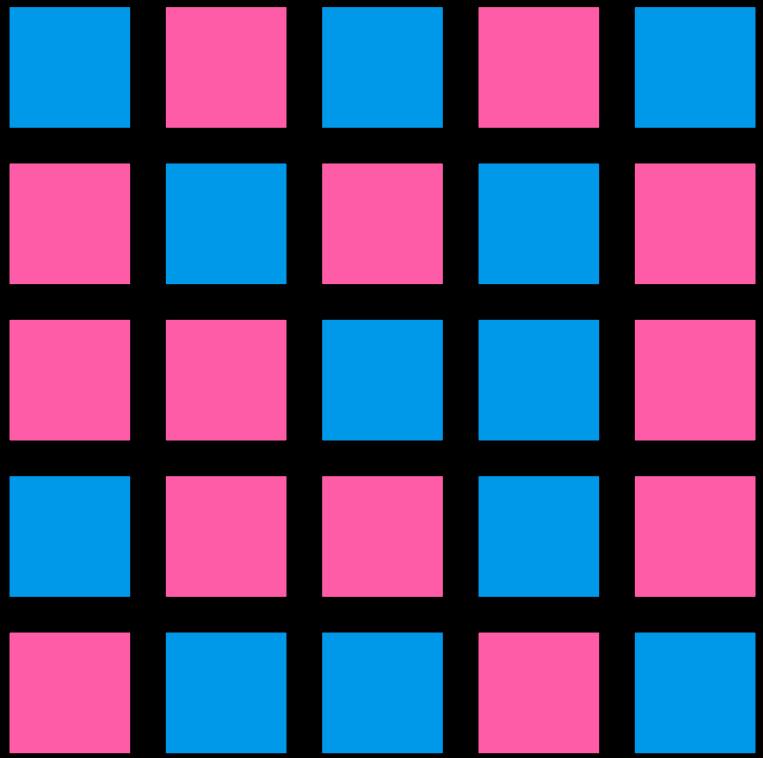
Screening

- How many squares are red?
- How did you work that out?



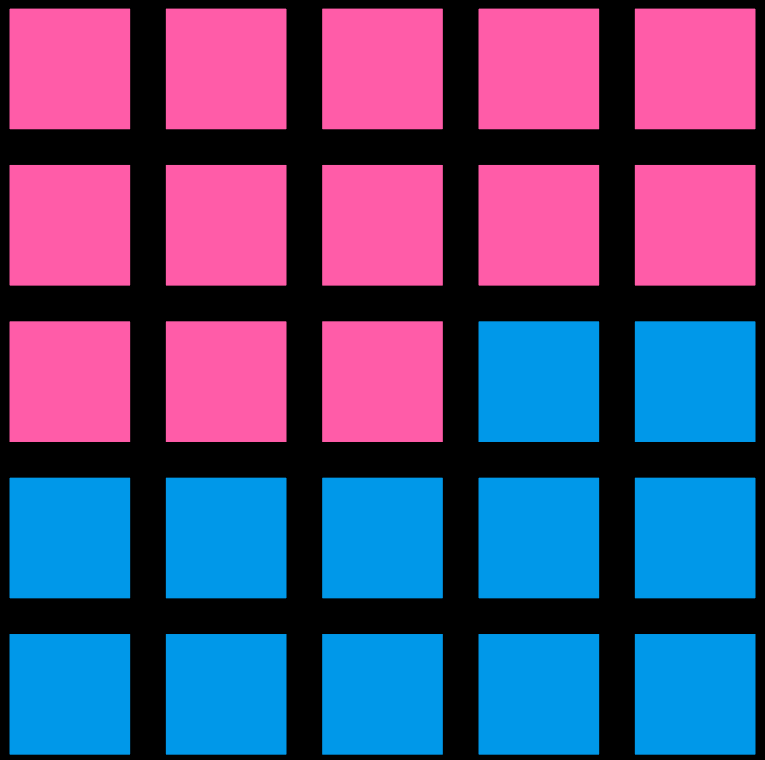
Screening

- Without order/ranking, all squares need to be looked at.



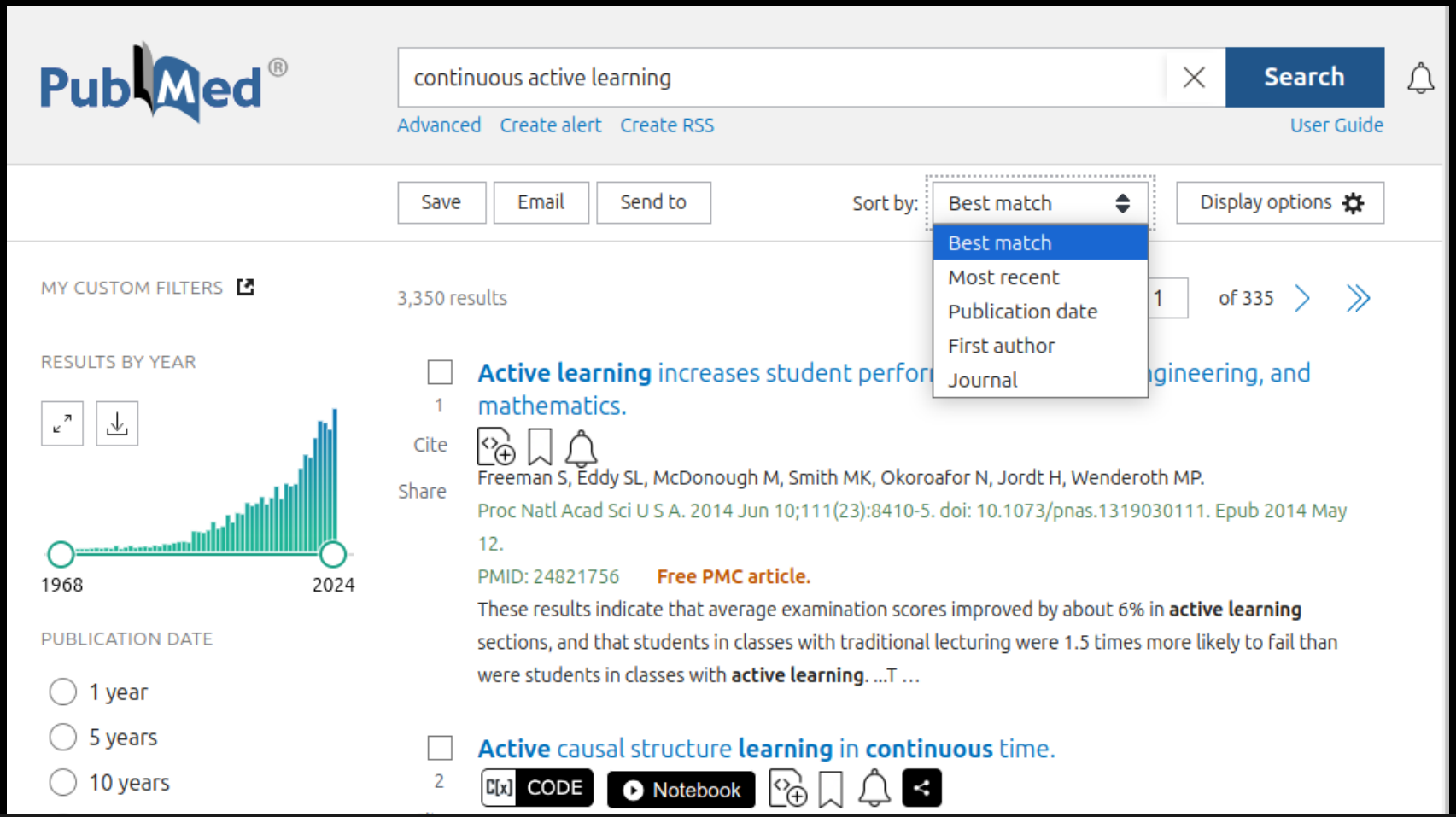
Screening Prioritisation: Order

- With order, we can look at fewer.

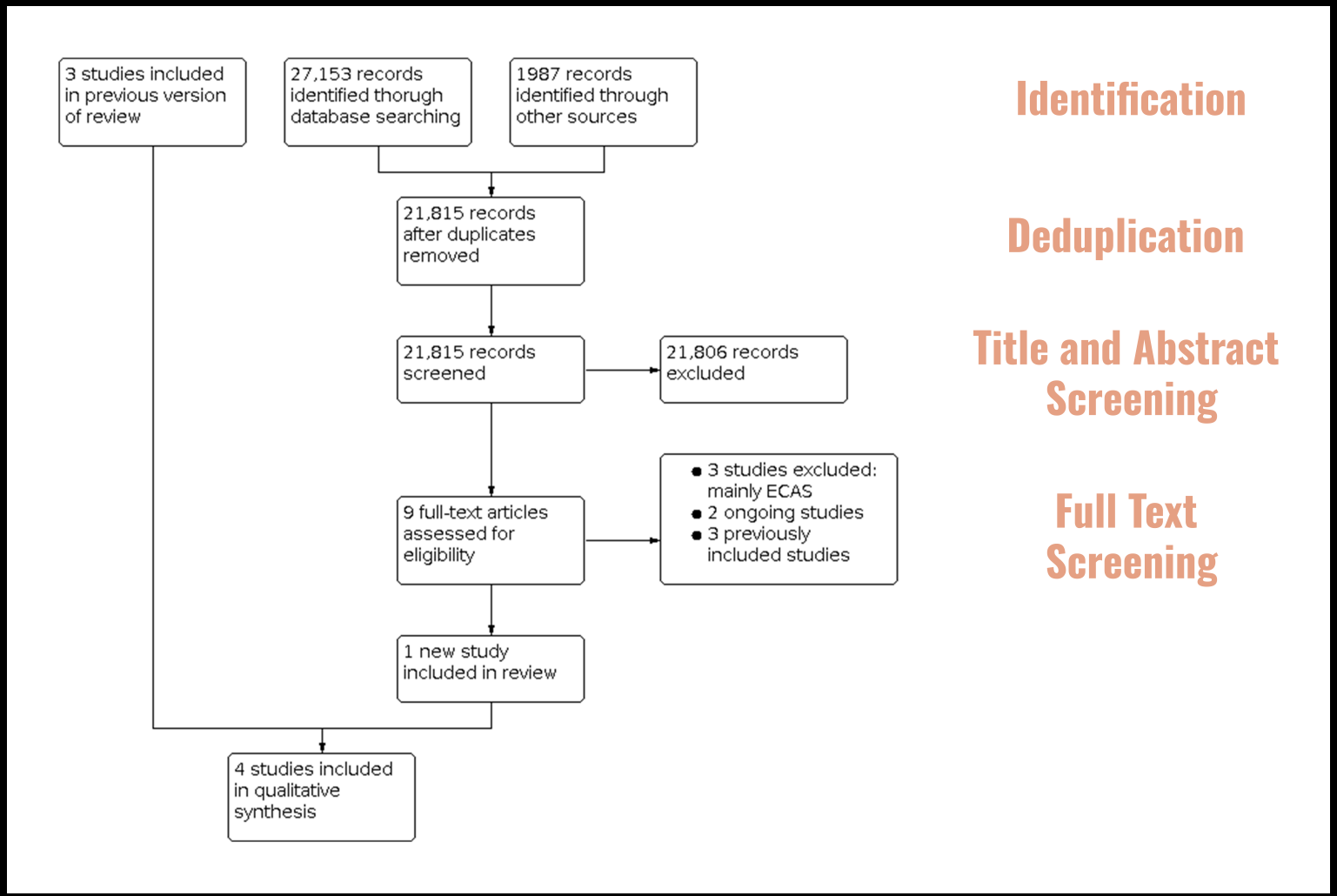


Screening Prioritisation: Ranking

- Ranking documents by relevance to a query is a common approach (There are others)
- Ranking of items is used in everyday life, e.g. Google, Amazon, Literature searches



Specific Example: Systematic Reviews[1]



[1] Luo et al. 2023.

Specific Example: Systematic Reviews

- SR have multiple phases (Identification, Deduplication, Screening, Eligibility)
- Screening has two stages (Title and Abstract Screening and Full Text Screening)
- 3 studies included from previous version of the review
- **Identification and Deduplication**: 29140 to 21,815 docs
- **Screening**: 21,815 to 9 docs
- **Eligibility**: to 1 doc
- ... 21815 docs were screened to find 1 eligible doc

Specific Example: Systematic Reviews

- Abstracts screening can average 0.13 to 2.88 abstracts a minute[2]
- Title and abstracts screening:
  - (at worst) 21815 / 0.13 = 167807 minutes (∼2796 hours)
  - (at best) 21815 / 2.88 = 7563 minutes (∼126 hours)

Key idea

Without prioritisation, unnecessary work is done

[2] Nussbaumer-Streit et al. 2021.

Specific Example: Systematic Reviews

- Wouldn't it be great if...

  - Only relevant abstracts are assessed first.

  or

  - Only difficult-to-categorise abstracts are assessed.

2. Active Learning

Iteratively querying a user to label data

Active Learning

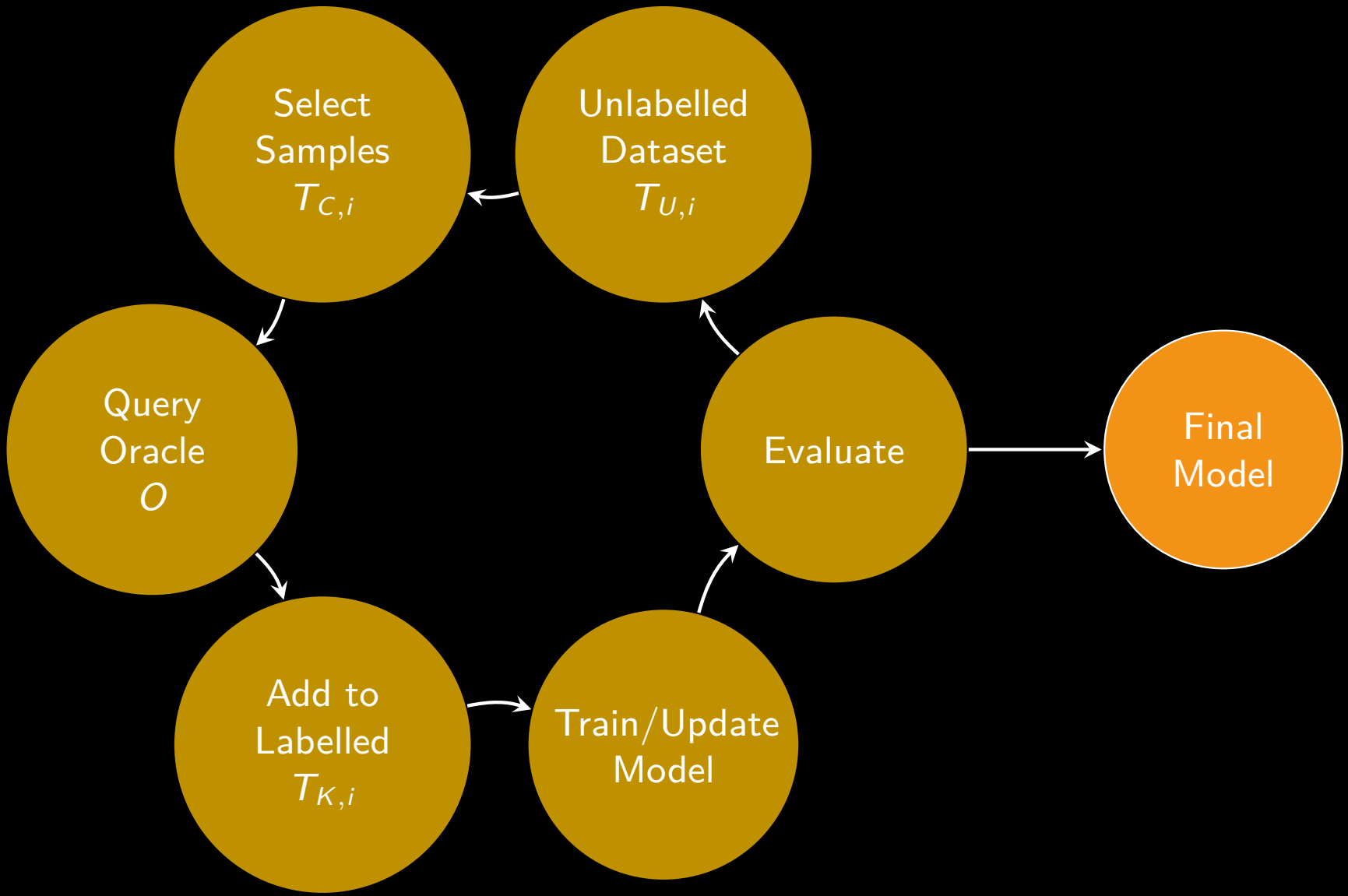
- Particularly useful when data labelling is expensive
- Can be used to prioritise screening
- A form of supervised learning

Active Learning: Process

1. **Represent documents** in a way that machines can understand
2. Form an initial training set
3. Train a **model** on that training set to rank documents
4. Use a **policy** to select a subset of unlabelled ranked documents
5. Add the selected documents to the training set
6. Retrain the model
7. Repeat

(And finally, **evaluate**)

Active Learning: Overview



Active Learning: Key Terms

O : Oracle: The information source, which in most cases is a human expert.

T : The Total Dataset: All documents to be screened.

$T_{K,i}$: All labelled documents at iteration i of the AL process.

$T_{U,i}$: All unlabelled documents at iteration i of the AL process.

$T_{C,i}$: A subset of $T_{U,i}$ selected for labelling at iteration i of the AL process.

π : The policy: A function that selects $T_{C,i}$ from $T_{U,i}$.

B The batch size: The number of documents selected for labelling at each iteration.

Dataset pool sizes at each iteration

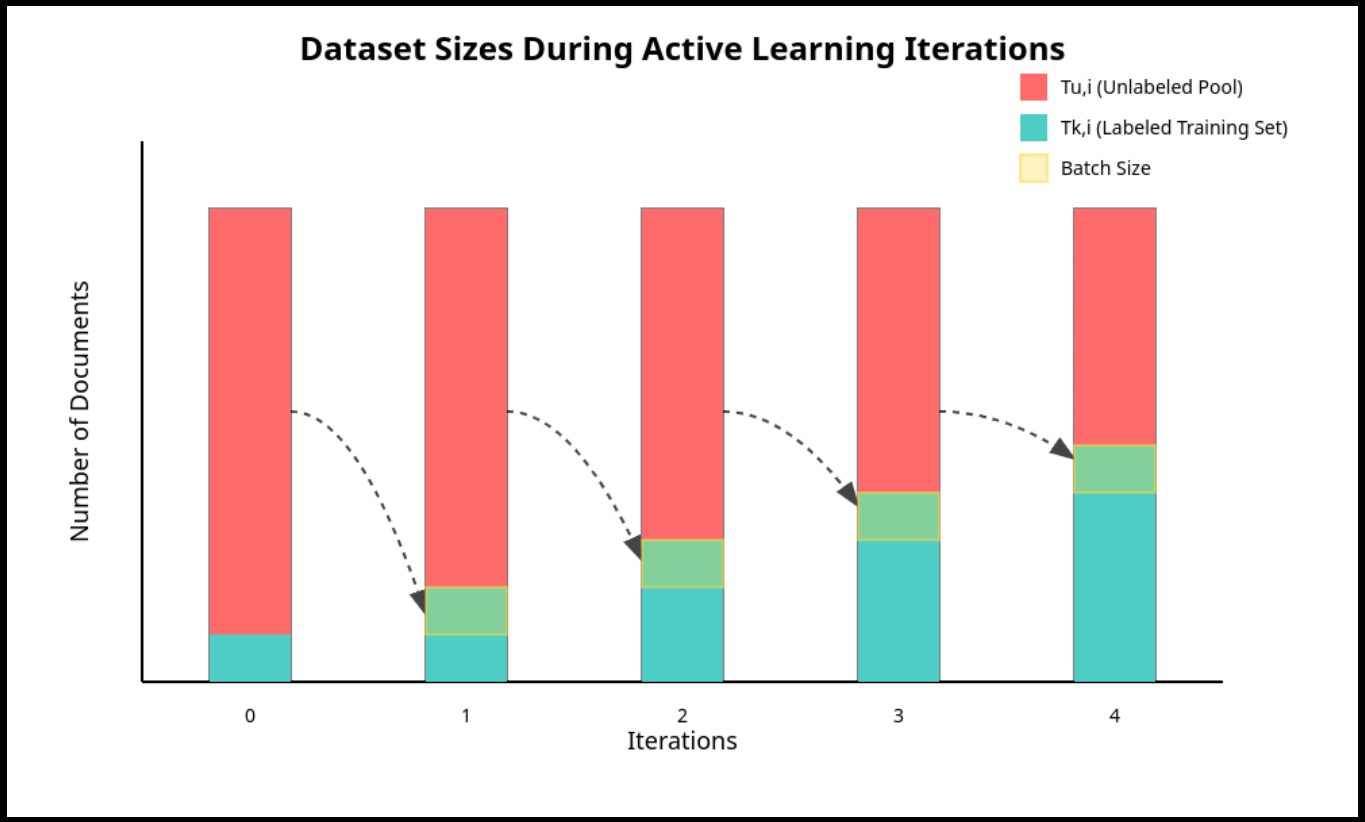


Figure: Changes in labelled and unlabelled pool sizes during active learning

Document Representations

- Many different approaches
- Each has its benefits and negatives
- Ranging from the simple (TF-IDF representation)
- To adapations (BM25 representation without query dependence)
- To more modern self-attention mechanisms

Sparse Document Representation: TF-IDF and BM25

Key idea

A document's importance is determined by its words[3].

TF-IDF Limitations:

- Linear term frequency: "cat cat cat cat" $\gg$ "cat cat"
- Length bias: Longer documents get higher TF scores

BM25 Improvements[4]:

- Term frequency saturation
- Length normalization ($b \approx 0.75$)
- Tunable scaling ($k_1 = 1.2\text{-}2.0$)
- Considers average document length

[3] SPARCK JONES 1972.

[4] Robertson and Walker 1994.

Dense Document Representation: Self Attention

Key idea

Learn the importance of word-word relationships[5]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- For each word w_i in document:
  - Query q_i: What information to look for
  - Key k_i: What information word contains
  - Value v_i: Actual content to be weighted
- Benefits:
  - Captures long-range dependencies
  - Position-aware representations
  - Parallel computation

[5] Vaswani et al. 2023.

Document Representation: Choosing Sparse vs Dense

Key idea

Transformer models leverage pre-trained domain knowledge

- **Benefits:**
  - Built-in domain knowledge from pre-training
- **Limitations:**
  - High computational cost and memory requirements
  - Limited interpretability
  - Requires large datasets for fine-tuning (which is an opposite need for the CAL process)

Document representation: Use in screening

We've represented the documents, lets use them!

Key idea

Classifier models rank documents to predict relevance

- Classification approaches:
  - Binary: Relevant vs. Not relevant
  - Categorical: Multiple levels of relevance
- Score/Ranking meanings vary by classifier:
  - Logistic Regression: Inclusion probability
  - SVM: Hyperplane distance
  - Random Forest: Aggregated tree votes

Policies: $\pi(T_{U,i})$

Key Definition

How documents are chosen from the ranked set is known as a policy (π).

- From this ranking, a subset of documents, with batch size B, is chosen for the oracle to label.
- π is a function that takes in the ranked set and outputs a subset of documents.
- A simple performant π could be to choose documents with the highest and lowest ranking scores.
- ... Relevance Sampling Policy.

Questions: $\pi(T_{U,i})$

Discussion Points

- **Why might relevance sampling be a good policy?**

Questions: $\pi(T_{U,i})$

Discussion Points

- **Why might relevance sampling be a good policy?**
- **Why might it be a bad policy?**

Questions: $\pi(T_{U,i})$

Discussion Points

- **Why might relevance sampling be a good policy?**
- **Why might it be a bad policy?**
- **Can you think of any other potential policies?**

Policies: $\pi(T_{U,i})$

- ↑ **Relevance Sampling**
  - Prioritizes highest and lowest ranked documents
  - Maximizes confidence in classification
- → **Uncertainty Sampling**
  - Focuses on boundary cases
  - Improves model precision
- **Random Sampling**
  - Provides unbiased document selection
  - Establishes baseline performance
- **Diversity Sampling**
  - Ensures varied document selection

Understanding Recall, Precision, and Recall@k

Core Metrics

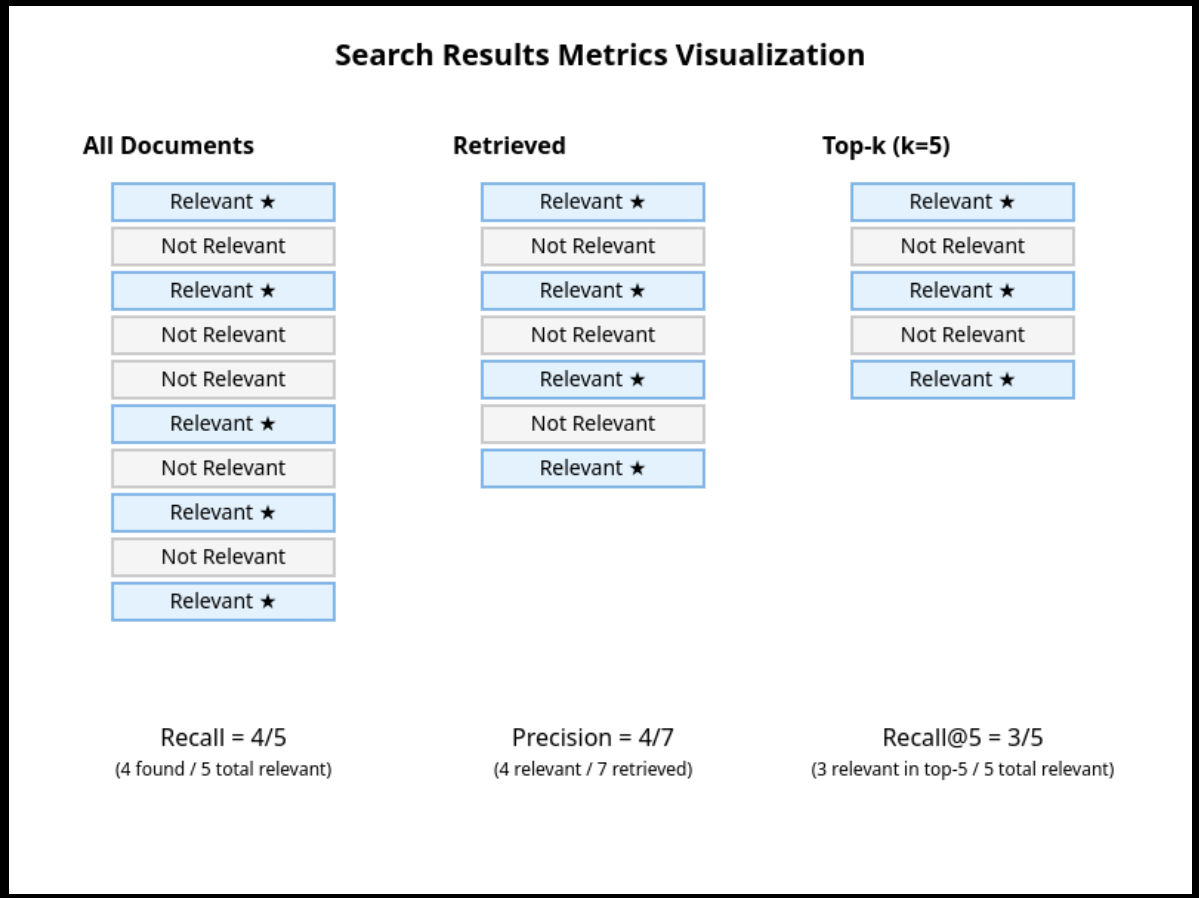
- **Recall**: Fraction of relevant items found
- **Precision**: Fraction of found items that are relevant
- **Recall@k**: Recall considering only top k ranked items

Why Recall@k?

- Evaluates ranking quality
- Reflects real-world usage (users often only check top results)
- Common in search and recommendation systems

$$\text{Recall@k} = \frac{\text{Relevant items in top k}}{\text{Total relevant items}}$$

Visualising Recall@k



Recap: Screening Prioritisation System

- We have all parts to create a screening prioritisation system.
  1. **Represent documents** in a way that machines can understand
  2. Form an initial training set
  3. Train a **model** on that training set to rank documents
  4. Use a **policy** to select a subset of unlabelled ranked documents
  5. Add the selected documents to the training set
  6. Retrain the model
  7. Repeat

  (And finally **evaluate**)

3. Sucessful implementations

Past, Present and Future

Let's explore some successful screening prioritisation systems

Approaches grouped based on the model type and document representation.

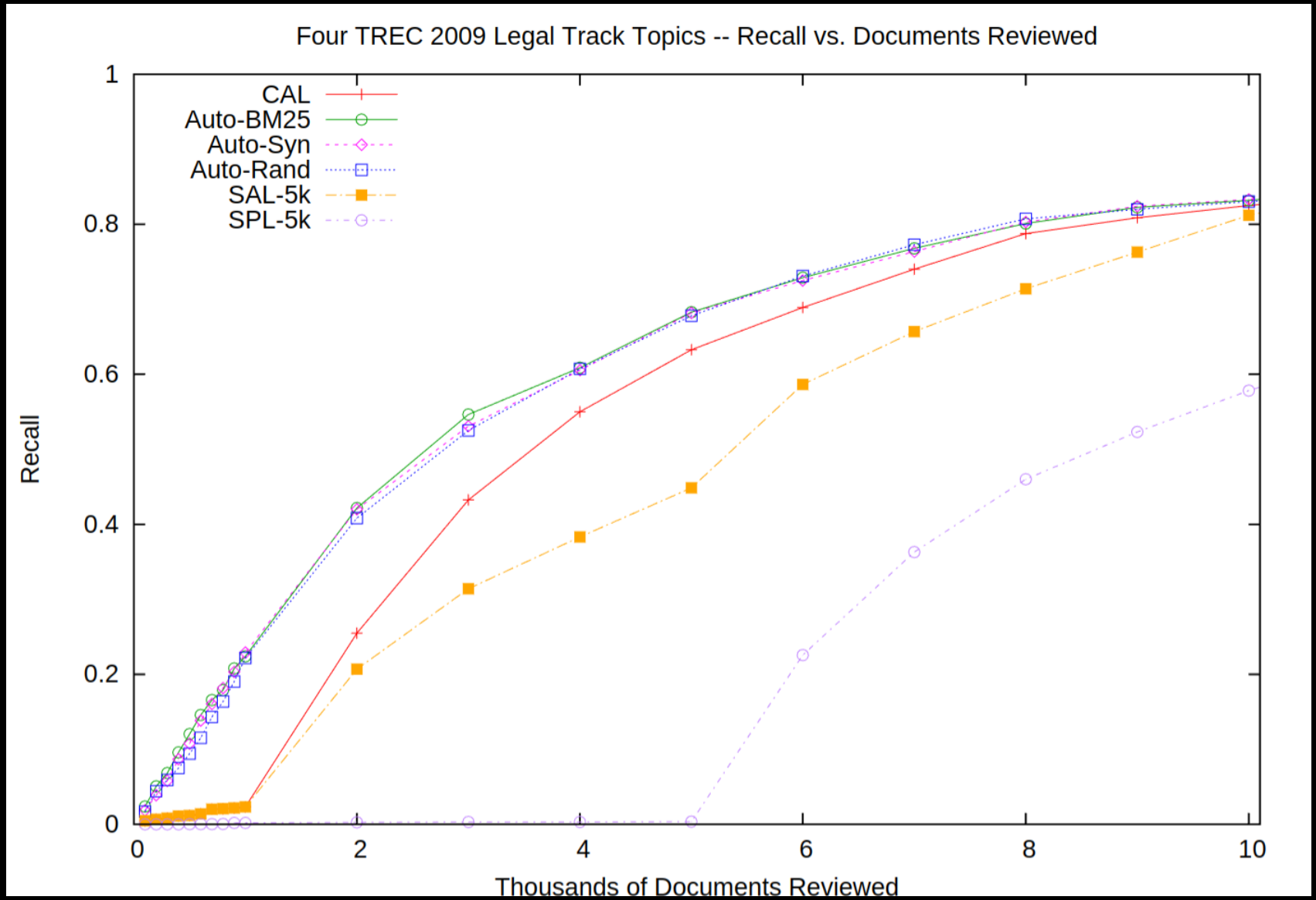
- **Feature-based Active Learning** (LR & BM25)
- **Encoder Active Learning** (Bert & Self Attention)
- **Decoder Active Learning** (Prompting & Self-Attention)

Feature-based CAL: AutoTAR[6]

1. Find relevant "seed" document via search or create
2. Add seed to training set as "relevant"
3. Initialize batch size $B = 1$
4. Temporarily add 100 random docs labelled "not relevant" forming $T_{K,1}$
5. Train SVM classifier
6. Remove temporary docs from $T_{K,1}$ and add back to $T_{U,1}$
7. Select B docs for review from $T_{U,1}$ using relevance π
8. O labels selected docs
9. Add to training set
10. Increase B by $\lceil B/10 \rceil$
11. Repeat 4-10 until termination criterion met

[6] Cormack and Grossman 2015.

Feature-based CAL: AutoTAR[7]



[7] Cormack and Grossman 2015.

AutoTAR: Key Points

- O must label increasingly large amounts of documents

Batch Size (B) Progression Over 30 Epochs

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11$$

$$\rightarrow 13 \rightarrow 15 \rightarrow 17 \rightarrow 19 \rightarrow 21 \rightarrow 24 \rightarrow 27 \rightarrow 30 \rightarrow 33 \rightarrow 37$$

$$\rightarrow 41 \rightarrow 46 \rightarrow 51 \rightarrow 57 \rightarrow 63 \rightarrow 70 \rightarrow 77 \rightarrow 85 \rightarrow 94 \rightarrow 104$$

- Classifier (SVM) not a true confidence measure
- Relevance π does not consider model uncertainty
- Model doesn't consider external knowledge
- Model is thrown away each iteration
- **Termination criterion:** When do you stop?

Encoder CAL[8]

Use BMI as a filter to reduce BERT's workload

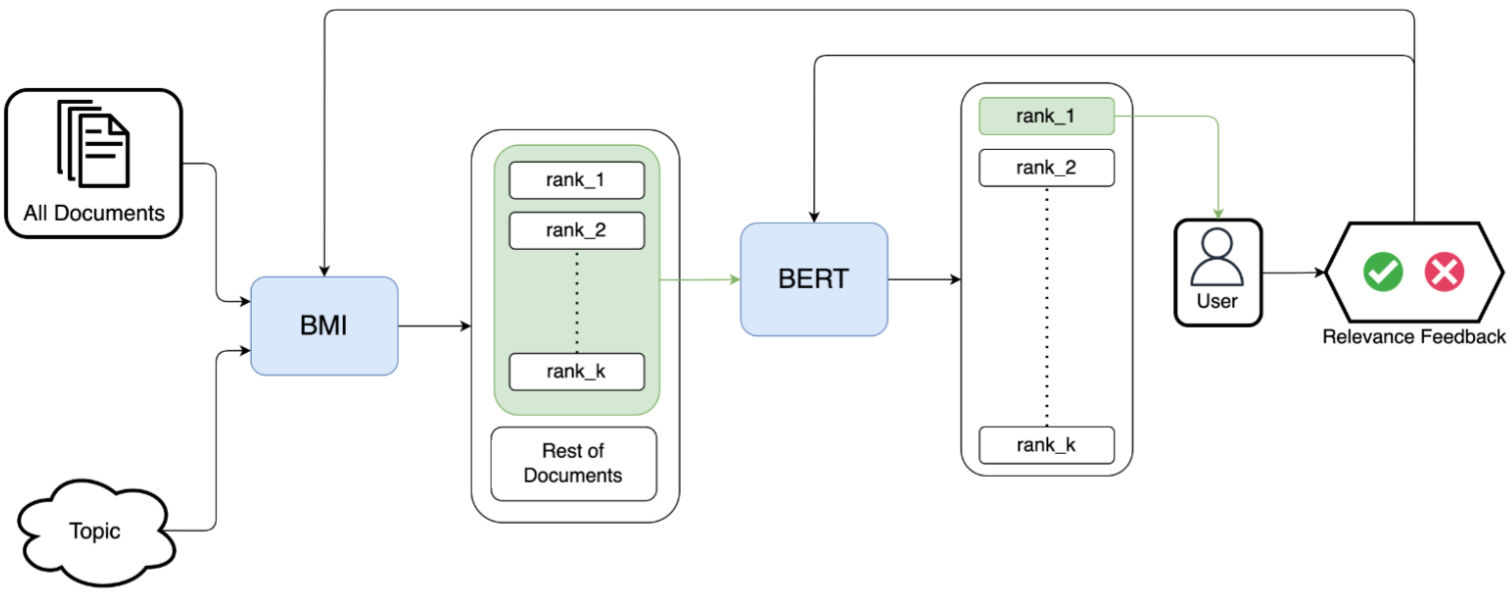


Figure 1: The CALBERT Model

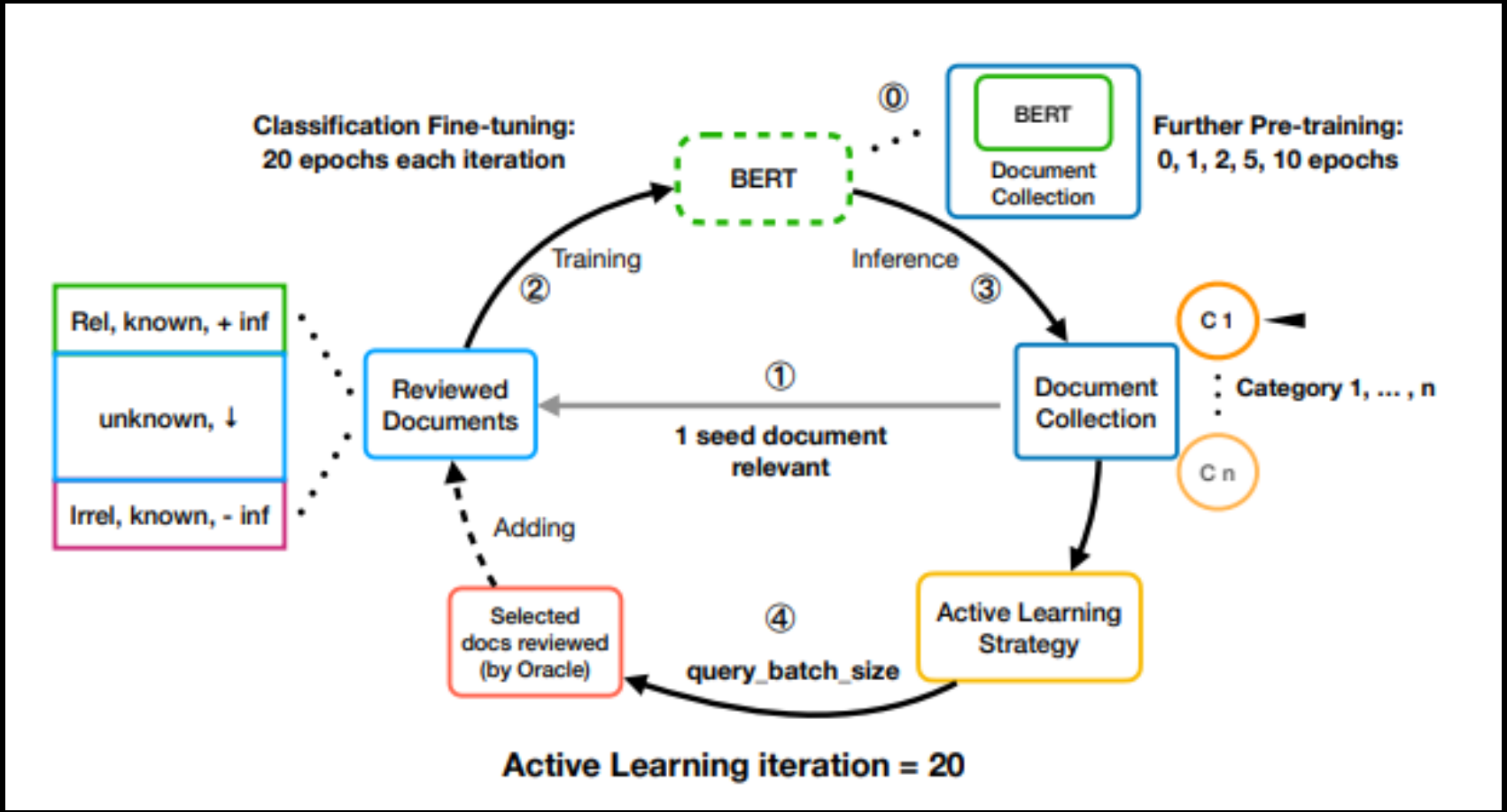
[8] Sadri and Cormack 2022.

Encoder CAL[9]

Model	k	epochs	training	P@100	R@100
BMI	-	-	-	**81.22**	**34.98**
monoBERT-large	10	0	-	72.60	28.60
monoBERT-large	100	0	-	61.07	19.15
monoT5-large	10	0	-	**80.26**	**31.80**
monoT5-large	100	0	-	59.73	21.08
monoBERT-large	10	0	-	78.86	32.59
monoBERT-large	100	0	-	64.76	19.77
monoT5-large	10	0	-	78.51	33.02
monoT5-large	100	0	-	65.49	19.99
CALBERT-large	10	5	scratch	**80.23**	**34.68**
CALBERT-large	100	5	scratch	76.94	31.92
CALBERT-large	100	5	incrementally	72.10	28.66
CALRoBERTa-large	10	5	incrementally	79.76	34.43

[9] Sadri and Cormack 2022.

Encoder CAL[10]



[10] Mao, Koopman, and Zuccon 2024.

Encoder CAL[11]

Collection	Further Pre-training Epoch	R-Precision (↑) Relevance	Uncertainty
CLEF 2017 train	baseline	**0.734**	**0.603**
	0	*0.612	0.598
	1	*0.674	0.679
	2	0.697	0.669
	5	0.711	0.670
	10	**0.722**	**0.680**
	BioLinkBERT-ep0	***0.838**	***0.761**
CLEF 2017 test	baseline	**0.782**	**0.657**
	0	0.727	0.722
	1	0.756	0.738
	2	0.746	0.748
	5	**0.776**	0.728
	10	0.776	**0.762**
	BioLinkBERT-ep0	**0.812**	***0.794**
CLEF 2018 test	baseline	**0.754**	**0.644**
	0	0.694	0.695
	1	0.725	0.725
	2	0.720	0.722
	5	0.729	0.729
	10	**0.747**	**0.750**
	BioLinkBERT-ep0	**0.793**	***0.780**

[11] Mao, Koopman, and Zuccon 2024.

Encoder CAL: Key Points

- **Classifier:** A pre-trained Encoder (BERT, BiolinkBERT, etc.)
- Classifier reinitialised each iteration
- Many hyperparameters (batch size, learning rate, etc.)
- Typically, only a few layers of the encoder are fine-tuned
- No great improvement on AutoTAR outside of specific pre-trained models/hyperparameters
- Loss of interpretability and computational efficiency

Decoder CAL?

- Use a decoder-based model, such as LLama2, GPT 4, etc.
- Prompting template to screen documents
- Not truely active learning - yet!
- What makes a good prompting template?
  - Actively researched topic

Decoder CAL?[12]

Model	Prompt
Alpaca	Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. ### Instruction: Answer 'yes' or 'no' to Judge if the following retrieved study should be included by the systematic review? ### Input: Review: *review_title* Study: *candidate_document* ### Response:
All Other Models	Answer 'yes' or 'no' to Judge if the following retrieved study should be included by the systematic review? Review: *review_title* Study: *candidate_document* The answer is '

[12] Wang et al. 2024.

Decoder CAL?[13]

	Model	P	R		Model	P	R
CLEF-2017	BioBERT	0.06	0.95*	CLEF-2019-Int	BioBERT	0.10	0.98*
	LlaMa-7b	0.04*	0.92*		LlaMa-7b	0.05*	0.86
	LlaMa2-7b	0.07	0.50*		LlaMa2-7b	0.08	0.30*
	LlaMa2-13b	0.04*	1.00*		LlaMa2-13b	0.05	1.00*
	Falcon-7b-ins	0.05*	0.92*		Falcon-7b-ins	0.05	0.91
	Alpaca-7b-ins	0.04*	0.92*		Alpaca-7b-ins	0.05	0.87
	LlaMa2-7b-ins	0.08	0.87		LlaMa2-7b-ins	0.08	0.90
	LlaMa2-13b-ins	**0.19***	0.41*		LlaMa2-13b-ins	**0.17***	0.45*
	Guanaco-7b-ins	0.04*	**1.00***		Guanaco-7b-ins	0.05	**1.00***
CLEF-2018	BioBERT	0.06	0.97*	CLEF-2019-dta	BioBERT	0.07	0.99
	LlaMa-7b	0.05*	0.92*		LlaMa-7b	0.07	0.93
	LlaMa2-7b	0.07	0.49*		LlaMa2-7b	0.08	0.48*
	LlaMa2-13b	0.05*	1.00*		LlaMa2-13b	0.07	1.00
	Falcon-7b-ins	0.05*	0.92		Falcon-7b-ins	0.07	0.95
	Alpaca-7b-ins	0.05*	0.91		Alpaca-7b-ins	0.07	0.91
	LlaMa2-7b-ins	0.09	0.88		LlaMa2-7b-ins	0.09	0.92
	LlaMa2-13b-ins	**0.26***	0.36*		LlaMa2-13b-ins	**0.19**	0.49*
	Guanaco-7b-ins	0.05*	**1.00***		Guanaco-7b-ins	0.07	**1.00**

[13] Wang et al. 2024.

Are Transformer CAL value for money?

If you have to pre-train a model from scratch, then probably not!

> BioLinkBERT of the -large size (340M params) from scratch, following the same procedure as -base, except that we use a peak learning rate of 4e-4 and warm up steps of 20%. Training took 21 days on eight A100 GPUs with fp16.

[14]

If you have to use APIs, probably also no!

> The predicted cost will be USD$4,000 and USD$80,000 if we use GPT-3.5-turbo and GPT-4, respectively. In our experiments, all employed models were configured to have a

[15]

[14] Yasunaga, Leskovec, and Liang 2022.

[15] Mao, Koopman, and Zuccon 2024.

Why isn't CAL standard for SRs?

- **Doctrine:** Long-established processes to screen documents for SRs
- **Reproducibility:** Models can be stochastic
- **Computational Cost:** Hard for small teams to access resources

Will CAL ever outperform human SRs?

- Should always have human-in-the-loop
- SR screening priorisation approaches are compared to existing ones performed by humans
- Assumption that humans always provide the correct label
- That is **FALSE**
- 10.76% error rate by humans in TAAS for false positive / negative inclusion[16]

[16] Wang et al. 2020.

Summary: Screening Prioritisation

- **Screening Challenge:** Reviewing large document collections can be done better
- **Active Learning Solution:**
  - Document representation (Sparse vs Dense)
  - Model training for document ranking
  - Policy selection (π) for efficient screening
  - Iterative improvement through oracle feedback
- **Implementation Approaches:**
  - Feature-based CAL (AutoTAR): Simple but effective
  - Encoder CAL: Leverages pre-trained knowledge
  - Decoder CAL: Emerging prompting-based methods
- **Current Challenges:**
  - Balancing cost vs. performance
  - Integration with established SR processes