Using relevance feedback and text similarity to reduce review effort in eDiscovery

Supervisors:
Dr. P.W.H vd Putten, Prof. Dr. Ir. J.C. Scholtes
L. Jean-Louis PhD & Z. Gerolemou MSc

Timo Kats, Universiteit Leiden.

Universiteit Leiden
The Netherlands

eDiscovery context 1/2

- Prior to lawsuits and investigations there's a ***discovery*** process.
- Produce ***evidence*** to the other party.
- ***eDiscovery*** refers to this process, but with the inclusion of electronically stored information (e.g. emails, PDFs).
- A set of documents is first collected and thereafter ***manually reviewed***.

eDiscovery context 2/2

- The ***manual reviewing*** of the collected evidence is the biggest cost driver in eDiscovery.
- High ***review effort*** (i.e. the amount of documents that have to be reviewed), is the main driver of these costs.

The Loneliness of the Long-Distance Document Reviewer: E-discovery and Cognitive Ergonomics [ADB09]

"To fulfill a similar transaction, the firm [Crowell & Moring] employed 125 contract lawyers for three months. They reviewed 30 million pages and produced 12 million relevant pages."

Problems in document review

- There are methods that reduce review effort.
- An ***active learning based approach*** with a human in the loop (also referred to as "Technology Assisted Review")
- Although effective at reducing review effort, a possibility of ***false negatives*** without the awareness of the user.

Tarpits: The sticky consequences of poorly implementing technology assisted review [Dow20]

"90% recall of relevant documents misses that the 10% of false negatives may be not just relevant, but crucial, even to the point of being more worthwhile than the other 90% altogether"

Research goal and questions

Research goal

- Use relevance feedback to reduce review effort.
  - ***Relevance feedback*** refers to the iterative process of improving the ***relevance ranking*** based on user feedback.
  - The ***relevance ranking*** will be based on the ***textual similarity*** between a queried document and the remaining documents.

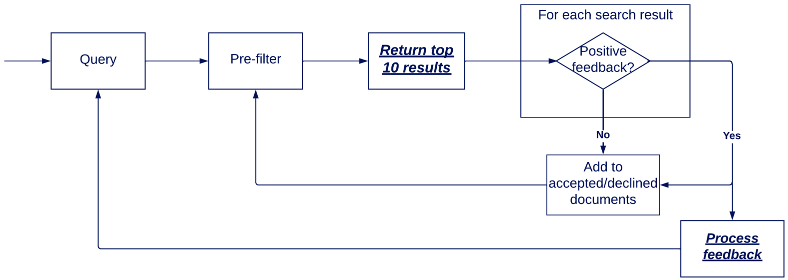
Research questions

1. Can text similarity be used for relevance feedback?
2. To what extent can relevance feedback help to reduce review effort?

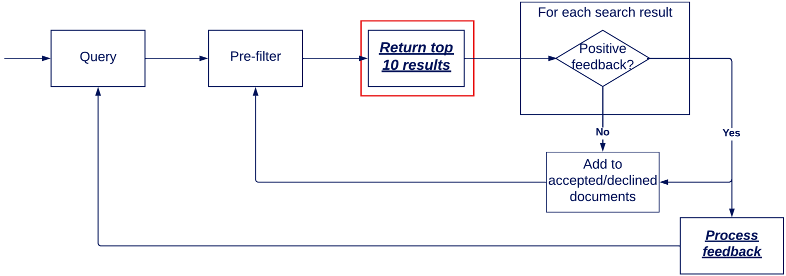
Data

- RCV 1 v2 dataset (annotated Reuters news articles)
- Test set (topics fairly unrelated to each other)
  - Strategy/Plans
  - Regulation/Policy
  - War/Civil War
  - Sports
  - Elections
- "Ambiguous" set (same parent topics)
  - EC Corporate policy
  - EC Internal market
  - Forex markets
  - Energy markets
- Random sample of 300 articles per topic.

Method


Query
Pre-filter
Return top 10 results
For each search result
Positive feedback?
No
Yes
Add to accepted/declined documents
Process feedback

Method



Query
Pre-filter
Return top 10 results
For each search result
Positive feedback?
No
Yes
Add to accepted/declined documents
Process feedback

Text similarity - Components

Pre-processing

- Removing numbers, special characters and text to lowercase.

Creating embeddings

- BERT based embeddings

Ranking

- Cosine distance between the embeddings

Implementation

- Solr's Dense Vector Search (DVS)

Text similarity - Embeddings

- ***BERT*** creates dense word embeddings that capture a form of ***semantic meaning*** through context.
- ***SBERT*** creates paragraph level embeddings through ***mean pooling*** these word embeddings (no document level).

```
Timo is a student > [-1.8472e-01, -3.1975e-01,  2.0524e-01,  ...]
```

SBERT [RG19]

"This reduces the effort for finding the most similar pair in a collection of 10,000 sentences from ***65 hours*** with BERT / RoBERTa to about ***5 seconds*** with SBERT…"

Text similarity - Ranking

- Text similarity can be computed through the ***cosine distance*** between the embeddings.
- On the paragraph level, the same document can be returned multiple times.

Paragraph based document rankings

Return the following 6 paragraphs: $\{d_1, d_2, d_2, d_3, d_3, d_3\}$ (where d_i refers to a paragraph from document i).

The first based ranking would be: $\{1, 2, 3\}$ whereas the count based ranking would be: $\{3, 2, 1\}$

Results - DVS configurations

Recall-Precision graph shows that DVS using the first based ranking outperforms DVS based on the count based ranking.

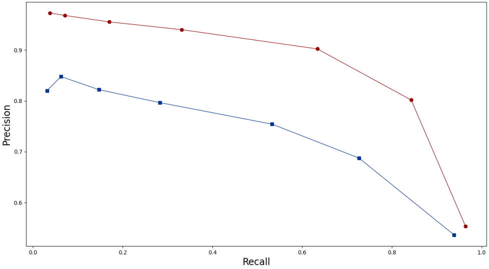
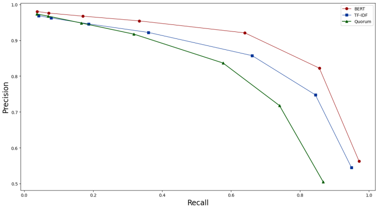


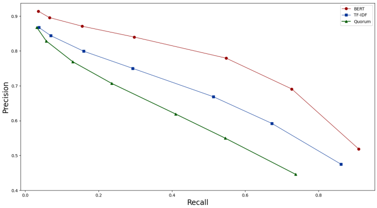
Figure: *DVS configurations*

Results - Baselines

Recall-Precision graph show that this DVS configuration outperforms the Quorum (occurence only) and TF-IDF based approaches.



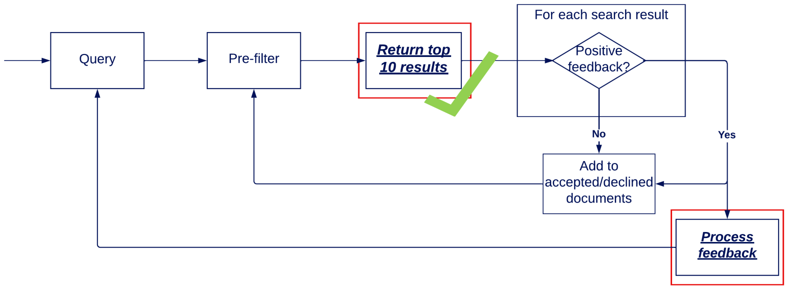
(a) *Test set*



(b) *Ambiguous set*

Figure: *Different configurations for DVS*

Method


Query
Pre-filter
Return top 10 results
For each search result
Positive feedback?
No
Yes
Add to accepted/declined documents
Process feedback

Relevance feedback - Components

Pseudo-relevance feedback

- ***Improve*** the queried embedding based on the ***selected*** search results (which re-ranks the results for the next ranking).
- We ***assume*** that articles that contain the queried topic get ***positive feedback*** from the “user”.

Feedback strategies

- Commonly used strategies/baselines:
  - No feedback (show next 10 results)
  - Keyword expansion
  - Rocchio
- Average/Sum the SBERT embeddings

Relevance feedback - Baselines

***Keyword expansion, each iteration*:**

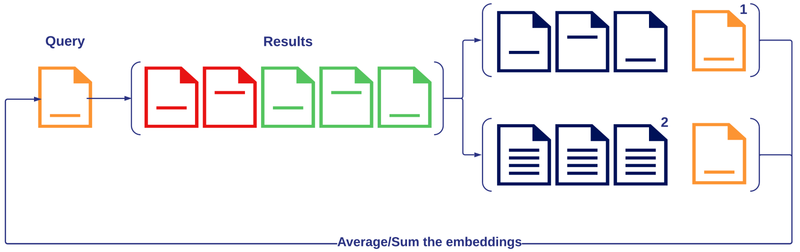
1. Get words from ***selected*** texts
2. Sort words by their *IDF* value (uniqueness)
3. Append the top 10 words to ***keyword filter***
4. Pre-filter results with: `keyword1 OR keyword2 OR ...`

Rocchio, each iteration:

1. Get average embedding of selected texts (α).
2. Average this (weighted) with queried embedding (β).
3. For this weighted average, we use $\boldsymbol{\alpha = 0.5, \beta = 0.5}$

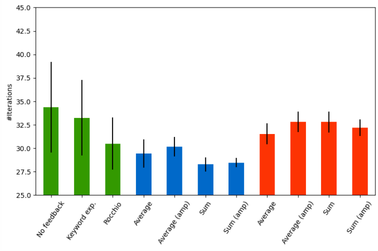
Relevance feedback - Strategies

Based on averaging/summing the SBERT embeddings. Variations based on (1) cumulative feedback and (2) feedback amplification.

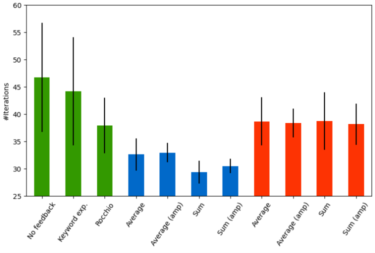


Relevance feedback - Review effort

- Cumulatively summing the embeddings reduces review effort the most (measured in iterations needed to achieve 80% recall).
- Adding sibling paragraphs reduces the standard deviation.



(a) *Test set*



(b) *Ambiguous set*

Figure: *Results of the relevance feedback strategies*

Relevance feedback – Latency

As for efficiency, summing/averaging the embeddings add little to no latency. Amplifying feedback does.

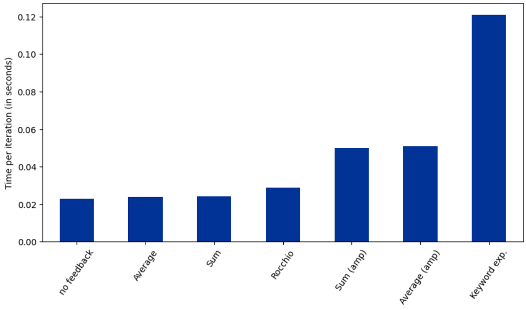


Figure: *Average iteration times for different feedback strategies*

Findings

- Compared to our minimal baseline (of no feedback), review effort is reduced between 17.85% (on the test set) and 59.04% (on the ambiguous set).
- Compared to an SVM based approach, this method is very fast.
- This method doesn't produce false negatives without the awareness of the user.

Limitations and future work

Limitations:

- Data size and data homogeneity.

Future work suggestions:

- Heterogeneous data (emails, contracts, direct messages, etc.)
- Chunking (compute text similarities in parallel)
- Generative AI
  - Help formulating queries
  - Summarizing results
  - A lot of developments that can be ***complementary*** to our method.

In conclusion

We found that...

- Our method reduced review effort between 17.85% and 59.04%.
- Very little latency.

However...

- Data homogeneity and size unrepresentative of eDiscovery

Regardless...

- Documents are only re-ranked, so no false negatives without the awareness of the user.
- Method applicable in real-world eDiscovery scenarios.

Thank you

Any questions?

```cpp
#include <iostream>
#include <string>

int main()
{
    std::string supervisors[4] = {"Jan", "Peter", "Zoe", "Ludovic"};

    for(int i = 0; i <= supervisors->length(); i++)
    {
        std::cout << "thank you " << supervisors[i] << std::endl;
    }
    return 0;
}
```

References

[ADB09] Simon Attfield, Stephen De Gabrielle, and Ann Blandford. "The loneliness of the long-distance document reviewer: e-Discovery and cognitive ergonomics". In: *DESI III workshop at ICAIL, Barcelona*. Citeseer. 2009.

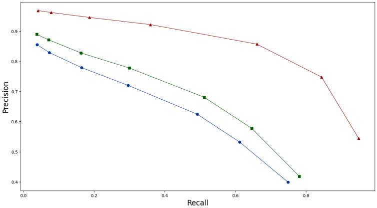
[RG19] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[Dow20] David Dowling. "Tarpits: The Sticky Consequences of Poorly Implementing Technology-Assisted Review". In: *Berkeley Tech. LJ* 35 (2020), p. 171.

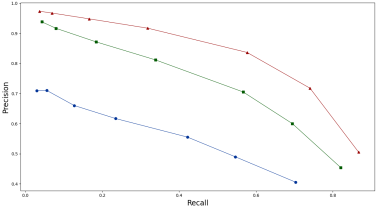
Additional information

TF-IDF/Quorum - paragraph level

For both methods the document level outperforms both paragraph based rankings.



(a) *TF-IDF*



(b) *Quorum search*

Figure: *Quorum and TF-IDF on the document/paragraph level*

Implementing this method

- DVS can be implemented out-of-the-box when using search engines like ***Solr***[1].
- SBERT embeddings can be created using the ***sentence-transformers***[2] Python library.
- The relevance feedback is implemented using ***Numpy***[3].

[1] https://solr.apache.org/
[2] https://www.sbert.net/
[3] https://numpy.org/

Error analyisis

Manually sampled and reviewed 50FPs and 50FNs per method and segmented into buckets. For the identified method (DVS):

False positives

- 10 due to “non topic similarities”
- 33 due to “partial topic overlap”
- 7 due to “bad paragraph segmentation”

False negatives

- 25 due to “dissimilar content”
- 9 due to “bad paragraph segmentation”
- 16 paragraphs are “too specific to match”

DVS optimization

DVS doesn't use chunking. Instead, it uses HNSW, which finds closest neighbors through (greedily) traversing through an hierarchical NSW network graph.

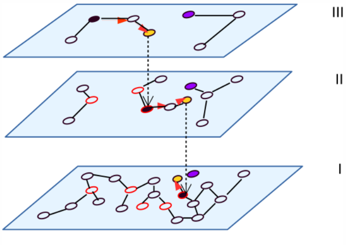


Figure: *Illustration of the HNSW algorithm*