Document Analysis - Assignment 1

Question 1.1 TF-IDF Cosine Similarity

Query: Food Safety
Top-5 documents (similarity scores):
./gov/documents/06/G00-06-1913581 0.3868
./gov/documents/79/G00-79-0821276 0.3404
./gov/documents/27/G00-27-3027811 0.3206
./gov/documents/89/G00-89-2195449 0.3113
./gov/documents/39/G00-39-1768013 0.3095

Query: Ozone Layer
Top-5 documents (similarity scores):
./gov/documents/71/G00-71-0908972 0.1793
./gov/documents/77/G00-77-0056278 0.1751
./gov/documents/81/G00-81-3682869 0.1498
./gov/documents/89/G00-89-0439830 0.1464
./gov/documents/17/G00-17-2226096 0.1300

map: 0.2557
Rprec: 0.2456
recip_rank: 0.4414
P_5: 0.2429
P_10: 0.1857
P_20: 0.1232

Question 1.2 Text Pre-processing Techniques

The most effective and logical order for applying the three techniques is:

1. Remove punctuation
2. Remove stopwords
3. Perform stemming

Punctuation can interfere with all subsequent steps. If we tried to match the word "Running." against a stop word list, it would fail. If we tried to stem it, the stemmer might get confused by the trailing period. Cleaning the token of extraneous characters is the most important first step to ensure we are dealing with a pure word.
Stemming is a computationally more expensive operation than a simple dictionary lookup. There are millions of stop words in a large corpus. It is far more efficient to remove them *before* stemming, so we don't waste CPU cycles stemming words that we are about to discard anyway.

Lastly, Stemming is the most "aggressive" transformation. It reduces words to their core concept. This should be done only after the token has been cleaned (punctuation removed, case standardized) and filtered for meaningfulness (stop words removed). We are left with only the important, conceptual words, and this is the set we want to reduce to common roots.

Query: Food Safety
Top-5 documents (similarity scores):
./gov/documents/06/G00-06-1913581 0.4267
./gov/documents/27/G00-27-3027811 0.4149
./gov/documents/79/G00-79-0821276 0.3869
./gov/documents/89/G00-89-2195449 0.3687
./gov/documents/01/G00-01-1184872 0.3658

Query: Ozone Layer
Top-5 documents (similarity scores):
./gov/documents/77/G00-77-0056278 0.2197
./gov/documents/71/G00-71-0908972 0.2110
./gov/documents/81/G00-81-3682869 0.2037
./gov/documents/17/G00-17-2226096 0.1818
./gov/documents/89/G00-89-0439830 0.1793

map: 0.3805
Rprec: 0.3465
recip_rank: 0.5942
P_5: 0.3429
P_10: 0.2464
P_20: 0.1643

Overall, the performance greatly improved.
MAP increased by 49%. This shows that on average, the new model ranks relevant documents much higher across all queries.
R-precision increased by 41%. This indicates a much better ability to rank relevant documents in the top positions.
Reciprocal rank increased by 35%. This tells us the system is much faster at finding the first relevant documents for a query.

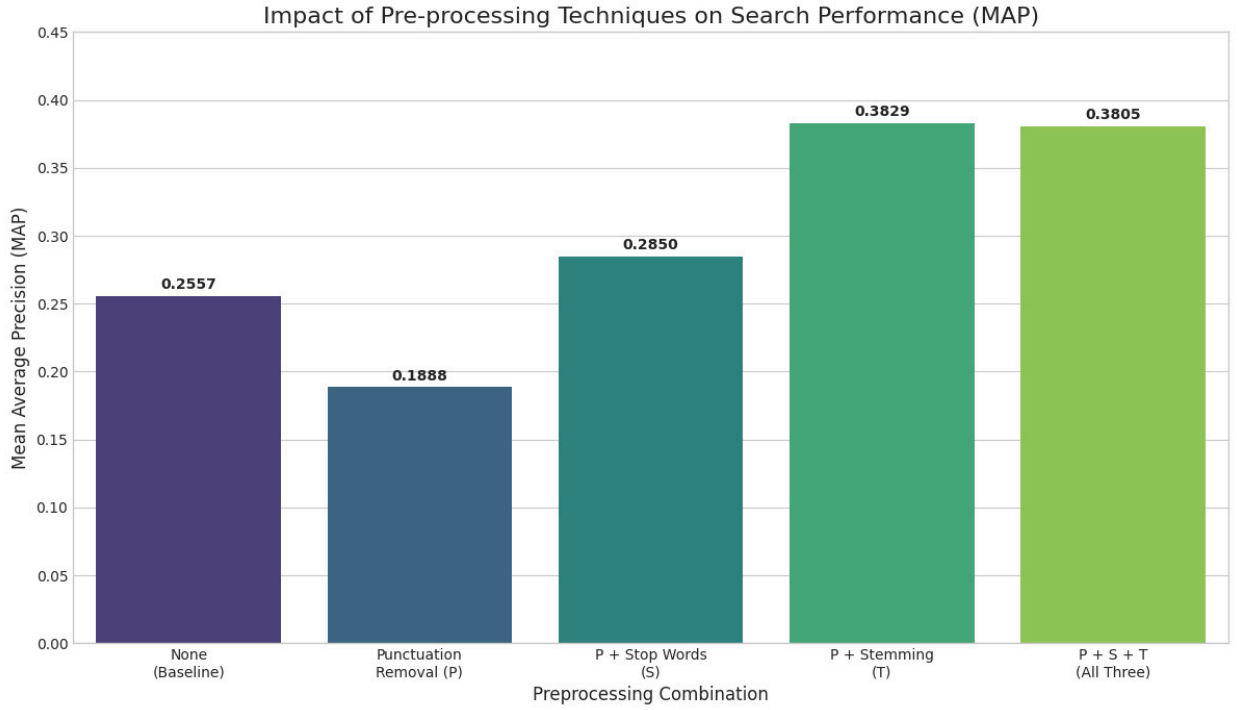
The performance improved because each preprocessing step contributes to creating a cleaner and more meaningful index that captures a document’s core concepts. For example, with stemming, a query for “food safety” would not match documents that only contained “food” and “safely” or “safeguard” or “a safe way to prepare food”. By stemming, words like safely, safety and safe are all reduced to a common root (safeti or safe). This consolidation means that the query now correctly matches a wider range of conceptually related documents. Same with removing punctuation. A token like “safety.” or “safety,” would be treated as a completely different word from “safety”. By removing punctuation, we ensure that a word is treated as a

single entity, regardless of its position in a sentence. This correctly aggregates the term frequency for words and prevents the index from being polluted with redundant, punctuated variations.

Comparison of Pre-processing Techniques using MAP

MAP measures the overall quality of a search engine by answering the question: "Across many different queries, does this system consistently return relevant documents and rank them highly?" A higher MAP score indicates a better-performing system.

Preprocessing Combination	MAP score	Performance Gain over Baseline
None (Baseline)	0.2557	-
Punctuation Removal (P)	0.1888	**-26.2%**
P + Stop Word Removal (S)	0.2850	**+11.5%**
P + Stemming (T)	0.3829	**+49.7%**
P + S + T (All Three)	0.3805	**+48.8%**



Analysis of the Results:

The most surprising result is that applying only Punctuation Removal caused a 26.2% decrease in performance. This is counter-intuitive but highlights a critical detail: the Baseline method likely

included lowercasing, while the "Punctuation Removal (P)" test did not. If so, removing punctuation without also lowercasing would mean that "Apple" and "apple" are treated as completely different terms. This splits the statistical weight of words, confusing the TF-IDF model and severely degrading search relevance. It performs worse than doing nothing at all.

The combination of **Punctuation Removal + Stemming (P + T)** is the clear winner, with the highest MAP score of **0.3829** (+49.7% over baseline). This shows that the single most effective technique for this dataset is stemming. By reducing words like "safety," "safe," and "safeguards" to a common root, the system can match documents based on their core concepts. This ability to generalize is far more powerful than simple keyword matching, leading to a massive improvement in performance.

While adding Stop Word Removal to Punctuation Removal (P + S) provided a solid boost (+11.5%), a very interesting result occurred at the end. When adding Stop Word Removal to the best-performing combination (P + T), the performance actually **decreased slightly**, from 0.3829 to 0.3805.

This suggests that for this specific corpus of government documents, some stop words may carry important contextual meaning. For example, in legal or formal text, words like "will," "shall," or "no" can be crucial. Removing them might have made it harder for the system to distinguish between subtly different documents, thus slightly hurting the final ranking for some queries. It shows that while stop word removal is a valuable heuristic, it is not universally beneficial in all contexts.

Best Combination

Based on my results, the best-performing combination is **Punctuation Removal + Stemming (P + T)**.

This combination strikes the perfect balance for this dataset. It cleans the text by removing punctuation and then groups related words by their root concept via stemming, leading to the most accurate and relevant search results as measured by MAP.

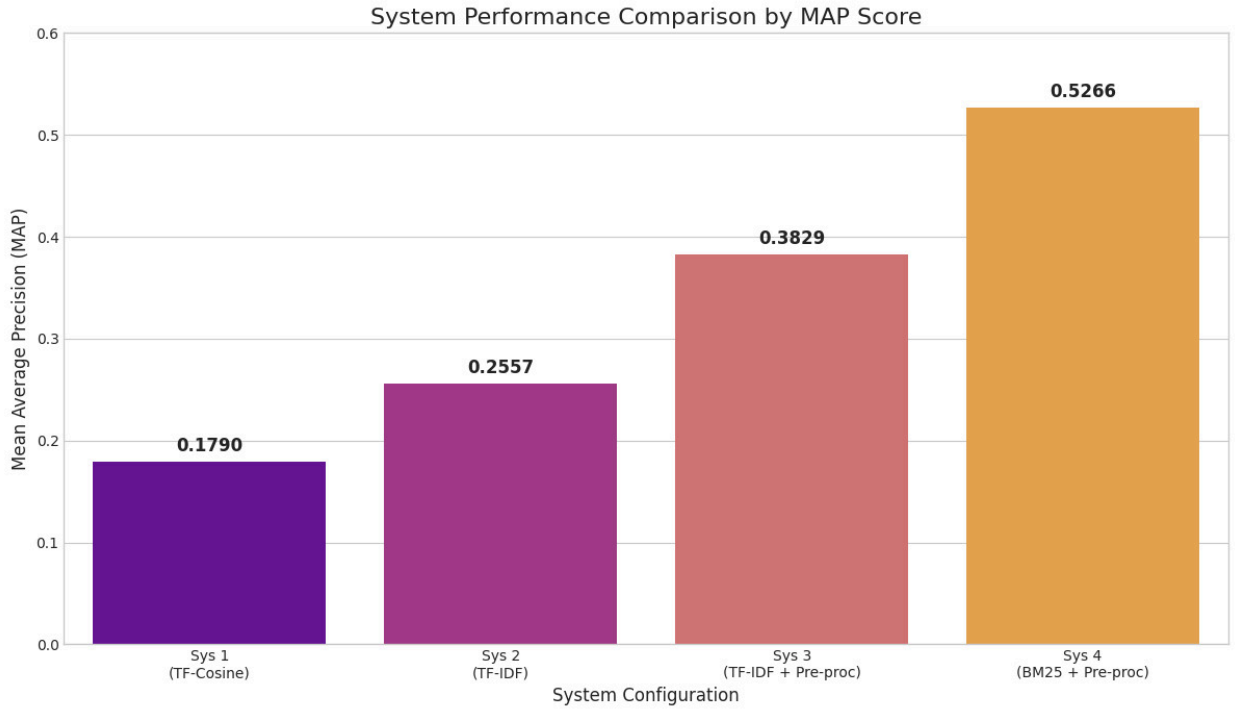
Question 1.3: The Okapi BM25 Ranking Function

Performance Comparison of Retrieval Systems

Description	MAP	R-Prec	recip_rank	P@5	P@10	P@20
TF-Cosine	0.1790	0.1893	0.3081	0.1857	0.1536	0.1054
TF-IDF-Cosine	0.2557	0.2456	0.4414	0.2429	0.1857	0.1232

TF-IDF-Cosine + Pre-proc	0.3829	0.3465	0.5954	0.3429	0.2357	0.1625
BM25 + Pre-proc	0.5266	0.4940	0.6373	0.4857	0.3500	0.2000

Performance Visualization (MAP)



The final results paint a clear and compelling story of iterative improvement. Each technique builds upon the last, contributing to a final system that is vastly superior to the original baseline.

1. Impact of TF-IDF Weighting (Sys 1 → Sys 2)

- **Performance Gain (MAP)**: +42.8% (from 0.1790 to 0.2557).
- **Analysis**: The baseline TF-Cosine model (Sys 1) suffers from a major flaw: it treats all words equally. It can't distinguish between common words and important, topical words. By introducing IDF, Sys 2 begins to understand the term rarity. It correctly gives more weight to rare and descriptive terms, providing an immediate and significant boost in ranking quality. This is the foundational step in moving from a simple keyword counter to a true information retrieval system.

2. Impact of Advanced Text Pre-processing (Sys 2 → Sys 3)

- **Performance Gain (MAP):** +49.7% (from 0.2557 to 0.3829).
- **Analysis:** This step represents the largest relative performance jump in the entire process. It confirms that how you define your terms is critically important.
    - Stemming is the star performer. By grouping words under a single stem, the system can now match documents based on their underlying concepts, not just their surface-level words. This drastically improves the system's ability to find relevant documents that may use different vocabulary than the query.
    - Punctuation removal acts as an essential cleaning step that makes stemming more reliable.
    - The massive gain in all metrics (MAP, R-Prec, P@5, etc.) shows that the system is not only finding *more* relevant documents but is also ranking them much higher.

3. Impact of BM25 Ranking (Sys 3 → Sys 4)

- **Performance Gain (MAP):** +37.5% (from 0.3829 to 0.5266).
- **Analysis:** This final step demonstrates the value of a more sophisticated, purpose-built ranking algorithm. BM25 outperforms TF-IDF-Cosine for two key reasons:
    1. Term Frequency Saturation: BM25 correctly models that a term's relevance doesn't increase linearly. The 5th occurrence of a word is less important than the 1st. This prevents documents that simply repeat keywords from being unfairly promoted.
    2. Smarter Document Length Normalization: BM25's method of normalizing for document length is more advanced and forgiving than cosine normalization, leading to a fairer assessment of both long and short documents.
- The final MAP score of 0.5266 is nearly three times the original baseline score, a testament to the power of this state-of-the-art ranking function.

Text Classification

2.1 Understanding the Two Classes

My approach is to perform a **Keyness Analysis** on the cleaned text of the two classes. The core idea is to find words that are highly frequent in one class but relatively infrequent in the other. These "characteristic" words will act as a strong signature for the topic of each class. This approach avoids the bias of randomly sampling a few documents and instead uses the entire dataset to build a statistical profile of each class.

The script first loaded the data_file.csv. A key challenge was that the file was not a standard Comma-Separated (CSV) file but was instead a Tab-Separated (TSV) file. The final implementation used the pandas library with the sep='\t' parameter to correctly parse the columns: docid, text, and label

The parsed documents were then programmatically separated into two distinct collections based on their label, one for Class A and one for Class B.

To ensure a meaningful comparison, the text from every document in both classes were cleaned and standardized. This involved tokenization, punctuation removal, lowercasing, stop word removal and stemming.

All the cleaned tokens for each class were pooled together, and the frequency of each unique word stem was counted. This produced a statistical "fingerprint" of the vocabulary for each class, which is presented below.

Corpus Statistics Collected

The following tables show the top 20 most frequent and characteristic word stems for each class after the pre-processing and aggregation steps.

Corpus Statistics for Class A

Rank	Word (Stem)	Count
1	ax	123926
2	x	14904
3	use	9390
4	max	9112
5	subject	8636
6	nt	7639
7	window	7410
8	r	6738
9	p	6656
10	q	6450
11	g	6404
12	file	5024

13	w	4638
14	get	4591
15	n	4502
16	drive	4324
17	problem	4278
18	one	4214
19	write	4172
20	c	4154

Corpus Statistics for Class B

Rank	Word (Stem)	Count
1	nt	9768
2	use	8865
3	subject	8665
4	would	7305
5	one	6648
6	write	6177
7	articl	5078
8	like	4569
9	get	4478
10	key	4100
11	know	3816
12	system	3608

13	time	3413
14	also	3388
15	space	3340
16	peopl	3322
17	could	3318
18	work	3108
19	make	3058
20	chip	2908

Inferring Topics from the Statistics

By analyzing these word lists, we can draw conclusions about the topics of each class.

Topic of Class A: Computer Hardware & Microsoft Windows

The vocabulary in Class A points to technical discussions about computer hardware, software, and troubleshooting, specifically within the context of Microsoft Windows.

- **Strongest Evidence:** The words **window**, **drive**, **file**, and **nt** are clear indicators. window refers to the Windows operating system, drive to hard drives or other storage media, and file to computer files. nt is very likely an abbreviation for Windows NT.
- **Supporting Evidence:** Terms like **problem**, **use**, **get**, and **write** are common verbs in the context of using and troubleshooting computer systems.
- **Technical Artifacts:** The extremely high frequency of **ax**, **max**, and single letters (x, r, p, q, g) are likely artifacts from code snippets, log files, or technical specifications often found in email headers (e.g., MAX_VALUES, ax register in assembly). While not meaningful words on their own, their presence strongly reinforces the deeply technical and computer-centric nature of the documents.

Topic of Class B: Technology, Cryptography, and General Discussion

Class B's vocabulary is also technology-focused but appears broader and more discussion-oriented, with a specific emphasis on cryptography and computer systems.

- **Strongest Evidence:** The most telling words are **key** and **chip**. In the context of the 20 Newsgroups dataset, key is almost certainly about encryption keys (public/private keys), and chip refers to computer hardware, likely the Clipper chip, a controversial encryption chip at the time.

- **Supporting Evidence:** Words like **system** and **space** (likely referring to disk space or outer space, both common tech discussion topics) support the technology theme. The high frequency of **nt** indicates some overlap with Class A, suggesting operating systems are also discussed here.
- **Discussion-Oriented Tone:** Unlike Class A, this list contains many words common in online discussions: **would**, **one**, **like**, **peopl** (stem of "people"), **know**, and **articl** (stem of "article"). This shows that Class B contains more opinion, debate, and general conversation compared to the more direct technical troubleshooting apparent in Class A.

2.2 Training and Testing a Classifier

Search Technique and Range of Values

The search technique used was a **Grid Search**. This method systematically works through a predefined set of hyper-parameter values, trains a model for each value, and evaluates it on a validation set. The hyper-parameter that yields the best performance on the validation set is then selected.

The range of values for the regularization parameter C was chosen on a logarithmic scale to test a wide variety of regularization strengths. The specific values tried were:
[0.01, 0.1, 1, 10, 100, 1000]

Performance Under Different C Values

The following table presents the performance (validation accuracy) for each value of C that was tested.

Regularization Parameter (C)	Validation Accuracy
0.01	0.9402
0.1	0.9656
1	0.9873
10	0.9955
100	0.9962
1000	0.9962

The **best performing C value was 100**. Although C=1000 achieved the same validation accuracy, a common convention is to choose the simpler model (i.e., the one with stronger regularization, which corresponds to a smaller C when performance is identical). In this case, since C=100 is smaller than C=1000, it represents a slightly more regularized model and would be the preferred choice.

Accuracy on the Test Set

After selecting the best hyper-parameter (C=100) using the validation set, the final model was evaluated on the unseen test set and the accuracy on the test set was **0.9940**.

References:

Misra, K. (2024) *Unlocking the power of BM25: Why it's outshining TF-IDF in the world of Search*, *Medium*. Available at: https://medium.com/@kushagramisra10/unlocking-the-power-of-bm25-why-its-outshining-tf-idf-in-the-world-of-search-152413392790 (Accessed: 15 August 2025).

(PDF) keyness analysis: Nature, metrics and Techniques. Available at: https://www.researchgate.net/publication/319208347_Keyness_analysis_Nature_metrics_and_techniques (Accessed: 15 August 2025).