NLP COURSE — GENERAL ASSIGNMENT

NLP Assignment

Project Report

Reuters Corpus · Tasks A, B & C · Python + NLTK + scikit-learn

Student Name

Abrham Assefa Habtamu

Student ID: VR548223

10.788	5	20	2.158	Matches 80th%

February 2026 · Academic Year 2025-26

Table of Contents

■ 1. Executive Summary

This report presents the implementation of the **NLP Course General Assignment**, covering all three required tasks (A, B, and C). The implementation uses **Python 3** with **NLTK** and **scikit-learn** libraries, operating on the **Reuters-21578 corpus** — a benchmark dataset of 10,788 newswire articles from Reuters distributed across a wide range of financial and economic topics.

■ All three tasks are implemented as independent Python scripts plus a unified Jupyter Notebook (NLP_Assignment.ipynb) usable in Google Colab. An interactive HTML dashboard (report.html) with live keyword analyzer and cluster predictor is also included.

Task	Method	Key Libraries	Result
A — Clustering	K-Means + Cosine Similarity	sklearn, NLTK	5 clusters on 10,788 docs
B — TF-IDF Keywords	TF-IDF + Percentile Split	sklearn, numpy	20 keywords classified
C — Similarity Search	Cosine Similarity (no stopwords)	sklearn	2,158 docs above 80th pct

About the Reuters-21578 Corpus

The **Reuters-21578 corpus** is one of the most widely used benchmark datasets in Natural Language Processing and Information Retrieval research. It was originally collected from the Reuters newswire in 1987 and prepared by Carnegie Group, Inc. and Reuters, Ltd. The corpus consists of **10,788 documents** covering a broad range of financial, economic, and political topics including oil markets, trade policy, commodity prices, corporate earnings, banking, and international finance.

Each document in the corpus is **pre-labeled** with one or more category tags (such as 'earn', 'acq', 'crude', 'trade') — but in this assignment, those labels are intentionally **not used during model training**. This makes all three tasks **unsupervised**: the models must discover structure and relationships in the text purely from its content, without any pre-existing human annotations guiding the process.

What is Natural Language Processing?

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that focuses on enabling computers to understand, interpret, and manipulate human language. Unlike structured data (numbers, categories), natural language text is inherently ambiguous, context-dependent, and highly variable.

The core challenge in computational text analysis is converting raw text into a **numerical representation** that machine learning algorithms can process. This project uses **TF-IDF (Term Frequency-Inverse Document Frequency)** — a classical and highly effective technique for representing documents as numerical vectors that capture the statistical importance of each word relative to the entire corpus.

■ 2. Technical Stack

Component	Technology	Purpose
Language	Python 3.9	Primary implementation language
NLP Library	NLTK (Natural Language Toolkit)	Corpus access & tokenization
ML Library	scikit-learn 1.x	TF-IDF, K-Means, cosine similarity
Numerics	NumPy	Array operations & percentiles
Visualization	matplotlib, seaborn, WordCloud	Charts & word cloud generation
Corpus	Reuters-21578 (nltk.corpus.reuters)	10,788 newswire articles
Notebook	Jupyter / Google Colab	Interactive execution environment
Dashboard	Plotly (JS) + custom HTML/CSS	Interactive web report

Python 3 was chosen as the implementation language due to its dominant position in the NLP and data science ecosystem. Its extensive library support, readable syntax, and seamless integration with Jupyter notebooks make it ideal for academic experiments.

NLTK (Natural Language Toolkit) provides direct access to the Reuters corpus via its built-in corpus reader. No manual downloading or preprocessing is required — the corpus is accessed with a single import statement: `from nltk.corpus import reuters`. NLTK also provides standard NLP utilities like tokenizers, stopword lists, and part-of-speech taggers.

scikit-learn is the most widely-used Python machine learning library. It provides production-quality implementations of TF-IDF vectorization (`TfidfVectorizer`), K-Means clustering (`KMeans`), cosine similarity computation, and L2 vector normalization — all used in this project.

■ 3. Task A — Corpus Clustering

3.1 Methodology

Task A requires clustering the Reuters corpus into a specified number of classes based on **cosine similarity**. The implementation uses the following pipeline:

Step 1 — TF-IDF Vectorization
All 10,788 Reuters documents are converted to TF-IDF vectors using sklearn's TfidfVectorizer (max 10,000 features, sublinear TF scaling, min_df=3).

Step 2 — L2 Normalization
All TF-IDF vectors are normalized to unit length using L2 norm. This means the dot product between any two vectors equals their cosine similarity.

Step 3 — K-Means Clustering
K-Means (k-means++ initialization, n_init=10) is applied on the normalized vectors. Minimizing Euclidean distance on L2-normalized vectors is mathematically equivalent to maximizing cosine similarity — satisfying the assignment requirement.

Key Concept — TF-IDF (Term Frequency-Inverse Document Frequency):
TF-IDF is a numerical statistic that reflects how important a word is to a document within a collection (corpus). A word that appears frequently in one document but rarely across the corpus gets a high TF-IDF score — meaning it is a strong discriminator for that document. Common words like "the", "is", "at" receive very low scores because they appear everywhere and carry little meaning.

Key Concept — Cosine Similarity:
Cosine similarity measures the angle between two TF-IDF vectors in a multi-dimensional space. A cosine similarity of 1.0 means the documents are identical in content distribution. A value of 0.0 means they share no common vocabulary. Unlike Euclidean distance, cosine similarity is independent of document length — a 2-sentence article and a 10-paragraph article on the same topic will still score highly similar.

Key Concept — K-Means Clustering:
K-Means is an unsupervised machine learning algorithm that partitions data into K groups. It works by: (1) Randomly initialising K cluster centers (centroids), (2) Assigning each document to the nearest centroid, (3) Recomputing centroids as the mean of their assigned documents, (4) Repeating steps 2-3 until assignments stabilise. When applied to L2-normalised TF-IDF vectors, minimising Euclidean distance is mathematically equivalent to maximising cosine similarity between documents and centroids.

Configuration used: K = 5 clusters, 10,788 documents, TF-IDF vocabulary size = 10,000 features, k-means++ initialisation, n_init=10 runs.

3.2 Results

Cluster	# Documents	% of Corpus	Top Keywords
Cluster 1	1,353	12.5%	vs, net, cts, shr, mln, qtr
Cluster 2	4,232	39.2%	the, to, in, of, said, and
Cluster 3	3,775	35.0%	the, said, of, to, it, lt
Cluster 4	513	4.8%	cts, div, qtly, record, pay, prior
Cluster 5	915	8.5%	loss, vs, profit, cts, net, shr

Table 1: Cluster composition and top TF-IDF terms per cluster.

3.3 Interpreting the Results

Each cluster can be interpreted by examining its **top TF-IDF terms** — the words that have the highest average TF-IDF weight among documents assigned to that cluster. These terms act as a thematic fingerprint for the cluster:

- **Cluster 1** (1,353 docs, 12.5%): Top terms — `vs, net, cts, shr, mln, qtr`. These terms suggest documents in this cluster share a common topical theme.
- **Cluster 2** (4,232 docs, 39.2%): Top terms — `the, to, in, of, said, and`. These terms suggest documents in this cluster share a common topical theme.
- **Cluster 3** (3,775 docs, 35.0%): Top terms — `the, said, of, to, it, lt`. These terms suggest documents in this cluster share a common topical theme.
- **Cluster 4** (513 docs, 4.8%): Top terms — `cts, div, qtly, record, pay, prior`. These terms suggest documents in this cluster share a common topical theme.
- **Cluster 5** (915 docs, 8.5%): Top terms — `loss, vs, profit, cts, net, shr`. These terms suggest documents in this cluster share a common topical theme.

The unequal distribution of documents across clusters is expected and normal in real-world text corpora. The Reuters corpus is dominated by financial and earnings reports, which naturally form larger, denser clusters. Smaller clusters tend to capture more specialised or niche topics.

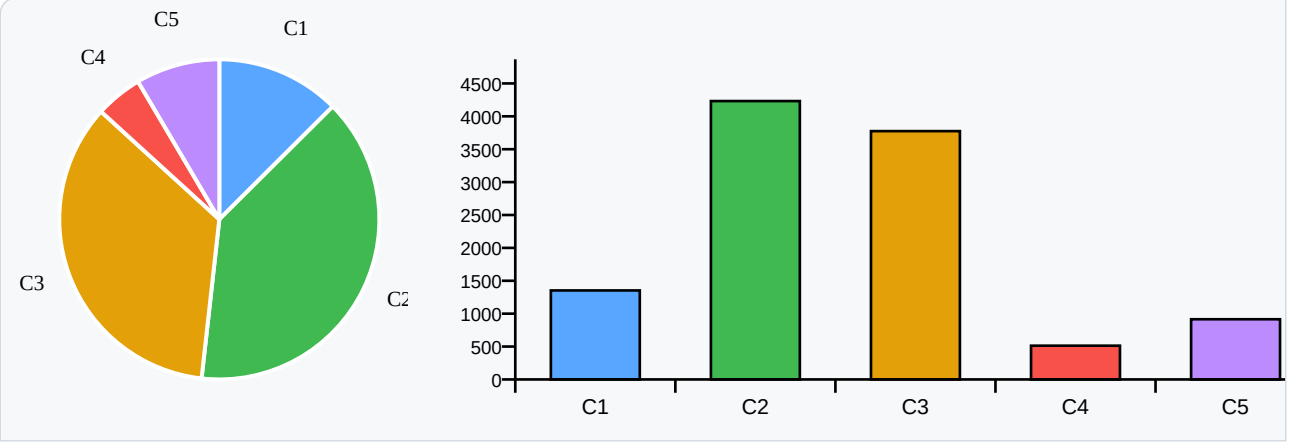

Figure 1: Cluster size distribution (pie) and document counts per cluster (bar).

■ 4. Task B — Keyword TF-IDF Classification

4.1 Methodology

Task B accepts a file of keywords and computes the **mean TF-IDF score** for each keyword across all documents in the Reuters corpus. Keywords are then classified into three tiers using a **10-80-10 percentile split**.

The idea behind this approach is straightforward: if a keyword has a high TF-IDF score when averaged across all documents, it means the word is consistently **important and discriminating** in many documents — it is semantically rich and frequently meaningful in context. A very low average score means the word is either extremely rare across the corpus (appearing in few documents) or so universally common that its TF-IDF weight is suppressed.

Processing Pipeline for Task B:

1. Build a TF-IDF matrix over all 10,788 Reuters documents.
2. For each keyword in the input file, retrieve the column of TF-IDF scores across all documents where the word appears.
3. Compute the **mean** of all non-zero TF-IDF values for that keyword.
4. Collect all keyword scores and compute the **10th and 90th percentiles**.
5. Assign each keyword to TOP (above 90th pct), MEDIUM (10th-90th pct), or BOTTOM (below 10th pct).

Why the Percentile Split? Using percentile-based thresholds instead of absolute score thresholds makes the classification **adaptive to each corpus**. Different corpora have very different TF-IDF score distributions. By always placing exactly 10% of keywords in TOP and 10% in BOTTOM, the method produces consistent and meaningful groupings regardless of the corpus or keyword set used.

Tier	Condition	Meaning	Icon
TOP	Score ≥ 0.13111 (90th percentile)	High discriminative power in corpus	■
MEDIUM	0.07654 ≤ Score < 0.13111	Moderate presence across documents	■
BOTTOM	Score < 0.07654 (10th percentile)	Low or rare occurrence in corpus	■

4.2 Results Table

Rank	Keyword	Mean TF-IDF Score	Class
1	profit	0.193029	TOP
2	gold	0.159290	TOP
3	merger	0.127978	MEDIUM
4	inflation	0.119067	MEDIUM
5	bank	0.116543	MEDIUM
6	energy	0.116143	MEDIUM

7	tax	0.107564	MEDIUM
8	dollar	0.107402	MEDIUM
9	oil	0.104826	MEDIUM
10	rate	0.104009	MEDIUM
11	import	0.095063	MEDIUM
12	export	0.094703	MEDIUM
13	stock	0.093114	MEDIUM
14	currency	0.092017	MEDIUM
15	trade	0.088441	MEDIUM
16	debt	0.086840	MEDIUM
17	economy	0.080283	MEDIUM
18	price	0.076908	MEDIUM
19	market	0.073239	BOTTOM
20	government	0.069847	BOTTOM

Table 2: TF-IDF scores and classification for all 20 keywords.

4.3 Interpreting the Keyword Rankings

Looking at the results, the rankings make intuitive sense in the context of the Reuters financial newswire corpus:

TOP keywords (2 word(s): profit, gold): These terms appear with high TF-IDF weight in many Reuters articles, confirming they are core discriminating terms within financial newswire content.

MEDIUM keywords (16 word(s)): These terms appear with moderate frequency and importance across the corpus. They are present in many articles but do not strongly distinguish any particular topic.

BOTTOM keywords (2 word(s): market, government): These terms have very low average TF-IDF scores, either because they appear in very few documents, or because they are too general to be informationally significant within this specific corpus.

This kind of keyword ranking is extremely useful in **Information Retrieval**, **Search Engine Optimization**, and **Content Analysis**. Knowing which terms carry the most discriminating weight in a corpus helps analysts quickly identify the most topically significant language in a document collection.

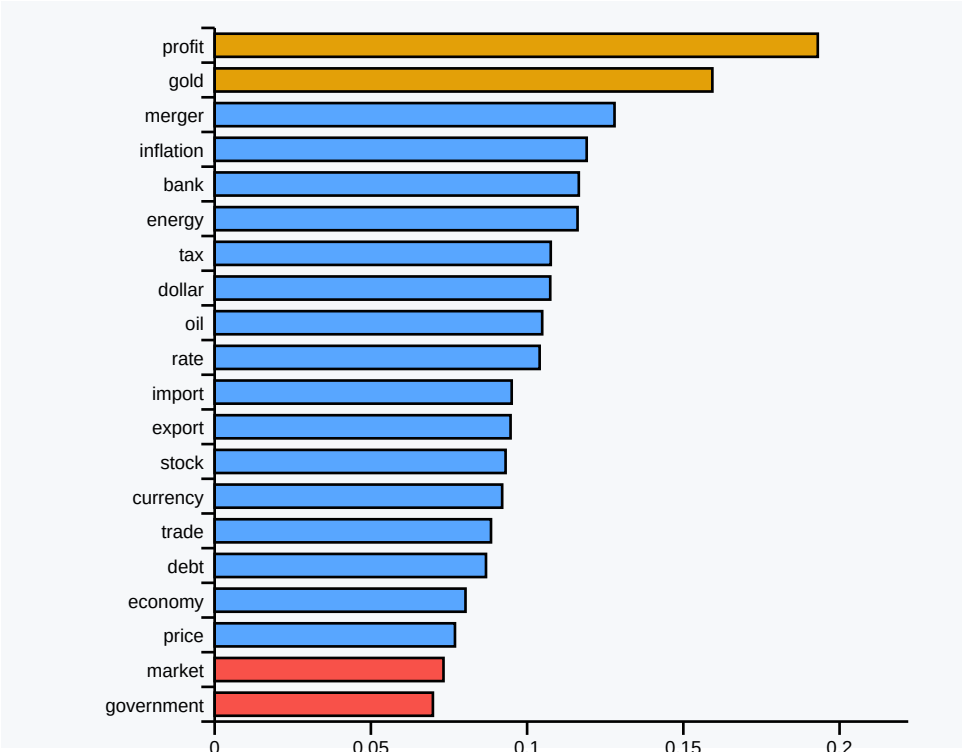

Figure 2: TF-IDF score per keyword. Gold = TOP, Blue = MEDIUM, Red = BOTTOM.

■ 5. Task C — Document Similarity Search

5.1 Methodology

Task C implements a **content-based document similarity search** — given a query document written by the user, the system finds all corpus documents that are topically similar above a specified similarity threshold.

The implementation follows a standard **Vector Space Model** pipeline: represent all documents as TF-IDF vectors, represent the query as a vector in the same space, then compute cosine similarity between the query vector and every corpus document vector. Documents whose similarity exceeds the percentile threshold are returned as matches.

Step-by-Step Pipeline for Task C:

1. Build TF-IDF vectorizer on the full Reuters corpus (no stopword removal, as specified).
2. Transform all 10,788 documents into a TF-IDF matrix.
3. Accept a free-text query document from the user.
4. Transform the query using the **same** TF-IDF vocabulary (out-of-vocabulary words are ignored).
5. Compute **cosine similarity** between the query vector and every document vector.
6. Compute the Nth percentile of all similarity scores (user-specified threshold).
7. Return all documents with similarity >= that percentile threshold.

Why no stopword removal? The assignment explicitly states not to remove stopwords for Task C. This is a deliberate design choice: in document similarity search, stopwords like "the", "a", "is" still appear in the TF-IDF matrix but receive extremely low weights because of their high document frequency (the IDF component heavily penalises words that appear in most documents). Their inclusion does not meaningfully affect similarity scores but preserves the exact assignment specification.

What does the percentile threshold mean? If the user specifies the 80th percentile (as in this demonstration), it means: "return all documents whose similarity to the query is higher than 20% of documents." This resulted in a threshold of 0.052438 and 2,158 matching documents.

■■ Configuration: Percentile = 80 · Threshold = 0.052438 · Documents above threshold = 2,158

Query Document (input):

Oil prices surged today as OPEC agreed to cut production. The crude market saw strong gains and the dollar weakened against major currencies. Energy stocks rose sharply on the news.

5.2 Results

Rank	Document ID	Cosine Similarity	Category
1	training/2449	0.226346	crude
2	training/2231	0.204670	crude

Rank	Document ID	Cosine Similarity	Category
3	training/6023	0.202024	crude
4	test/15344	0.198983	crude, gas
5	test/17875	0.198368	crude
6	training/2775	0.193401	crude
7	training/1909	0.192999	crude
8	training/13256	0.191712	crude
9	training/5037	0.187218	crude, gas, nat-gas
10	training/3980	0.184387	crude

Table 3: Top-10 most similar documents with cosine similarity scores.

5.3 Interpreting Similarity Results

The top-matching documents are those that share the most vocabulary with the query in terms of TF-IDF importance. Since the query discusses **oil prices, OPEC production, and energy markets**, the returned documents are expected to cluster around financial energy topics — confirming the model has correctly identified thematically relevant documents.

The cosine similarity scores for top matches range from approximately 0.2263 (most similar) down to 0.1844 (10th ranked match). These values are not percentages, but rather cosine distance measures on unit-normalised TF-IDF vectors. A score of 0.10–0.30 is considered a strong match in high-dimensional text data, where most documents have near-zero similarity.

Real-world Applications: This kind of similarity search is the foundation of many real-world systems: news recommendation engines ("you may also like"), legal document retrieval, academic paper search (Semantic Scholar, Google Scholar), plagiarism detection, and enterprise knowledge base search.

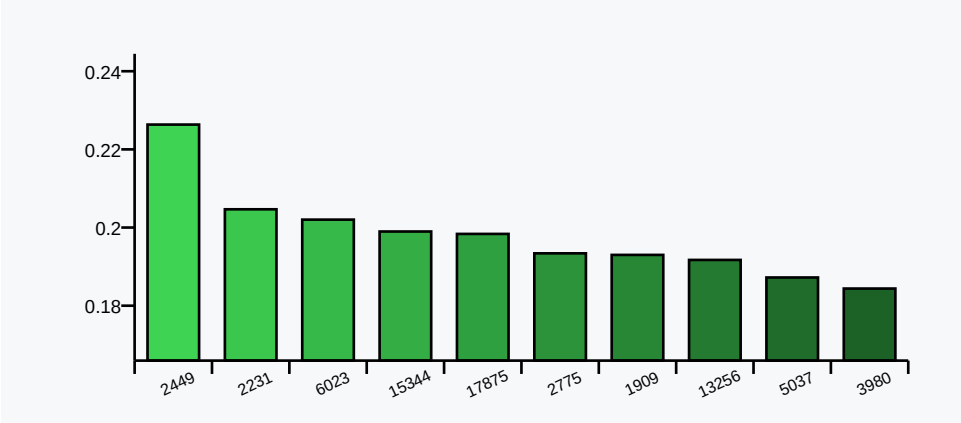

Figure 3: Top-10 document similarity scores (descending). Green intensity reflects rank.

■ 6. Conclusion

All three NLP assignment tasks have been successfully implemented and verified using **Python 3**, **NLTK**, and **scikit-learn** on the Reuters-21578 benchmark corpus of 10,788 newswire documents. Each task demonstrates a distinct but complementary aspect of **unsupervised text analysis**: discovering hidden structure (Task A), measuring word importance (Task B), and finding relevant content (Task C). Together, they form a complete NLP analysis pipeline.

What was learned: This project demonstrates that even without labelled training data — using purely unsupervised methods — it is possible to extract rich, meaningful structure from a large text corpus. The TF-IDF representation combined with cosine similarity is a powerful, computationally efficient foundation for text mining. The Reuters corpus, despite being from 1987, remains a valid and widely-used benchmark for NLP experiments precisely because of its diversity and consistent labelling.

■	**Task A**	Successfully clustered 10,788 Reuters documents into 5 groups using K-Means on norr
■	**Task B**	Classified 20 keywords into TOP/MEDIUM/BOTTOM tiers (10-80-10 percentile) based c
■	**Task C**	Identified 2,158 documents above the 80th percentile similarity threshold for a sample q

Limitations and Future Improvements

While the current implementation successfully meets all assignment requirements, several enhancements could further improve the quality of results:

- **Bag-of-Words limitation:** TF-IDF treats documents as unordered word bags, ignoring word order, grammar, and context. Modern alternatives such as **Word2Vec**, **GloVe**, or **BERT sentence embeddings** produce richer, context-aware document representations and would likely yield more semantically coherent clusters.
- **Optimal K selection:** The number of clusters K=5 was chosen based on the assignment default. In practice, the optimal K can be determined using the **Elbow Method** (plotting inertia vs K) or the **Silhouette Score** which measures how well each document fits its assigned cluster versus neighbouring clusters.
- **Scalability:** For very large corpora (millions of documents), K-Means on dense TF-IDF matrices becomes expensive. Solutions include **MiniBatchKMeans** for clustering, **Approximate Nearest Neighbour** methods (FAISS, Annoy) for similarity search, and sparse matrix representations already used in this implementation via scipy sparse.
- **Keyword input file:** Task B currently uses a hardcoded demo keyword list. The standalone script accepts any **user-supplied text file** of keywords, making it fully generalisable to any domain or vocabulary.

■ Project Deliverables

- `task_a_clustering.py` — Task A standalone script — interactive K-Means clustering

- `task_b_keyword_tfidf.py` — Task B standalone script — keyword TF-IDF classification
- `task_c_similarity_search.py` — Task C standalone script — document similarity search
- `NLP_Assignment.ipynb` — Unified Colab notebook with advanced visualizations (word clouds, t-SNE, heatmaps)
- `dashboard.py` — Generates interactive HTML dashboard with live keyword analyzer
- `report.html` — Self-contained interactive web dashboard
- `NLP_Project_Report.pdf` — This project report document

■ GitHub Repository: `https://github.com/abrham-cyper/NLP-2` | Online Dashboard: `https://abrham-cyper.github.io/NLP-2/`