Report on Classifier Evaluation for PDF-to-Text Processing

Pablo de Vicente Abad 3/05/2025

Overview

This report presents an evaluation of various classifiers within the PDF-to-text classification pipeline. The analysis explores performance across two main axes. We will evaluate the accuracies of the different set of data points in this report

- **Vector Space Models (VSMs):**
  Three VSM techniques were tested to represent document text: **Word2Vec**, **GloVe**, and **FastText**. Initial experiments also included fine-tuned versions of these models; however, they were ultimately discarded due to a lack of performance improvement. For further details, refer to the *VSM Evaluation Report*.

- **Classifier Models:**
  Using each VSM representation, we tested three classification algorithms: **Logistic Regression**, **Support Vector Machines (SVM)**, and **Random Forest**. The goal was to

assess how different classifiers perform when applied to varying vector representations of textual data.

Dataset Composition

This report tests a first demo version composed of 5 classes and 50 documents per class.The classes were chosen at random

- **cable-ties-zip-ties:** 39 documents
- **coaxial-cables-rf:** 45 documents
- **Controller Accessories:** 50 documents
- **Batteries (Non-Rechargeable):** 49 documents
- **printers-label-makers:** 45 documents

This proved to be insufficient for reasons discussed later, so a second demo database of 5 classes and 150 documents per class was also created. Some consideration was taken into the selection of the set of classes, mainly choosing those classes with more documents already downloaded locally, so as not to have to go back and re-download some data.

- **Speakers:** 150 documents
- **Microphones:** 171 documents
- **Controller Accessories:** 150 documents
- **Batteries (Non-Rechargeable):** 152 documents
- **Alarms:** 150 documents

Results

In this section, we present the outcomes of the various experiments conducted, detailing the methodology and reasoning behind each setup.

Experiment 1: Baseline

As an initial baseline, we evaluated several vector encoding strategies in combination with different classifier types. The goal of this experiment was to establish a reference point for subsequent comparisons. This preliminary evaluation focused on a subset of the dataset consisting of five document classes, each containing approximately 50 documents.

Note: Due to computational constraints, the table-to-text generation workflow was not included in any of the experiments presented in this report.

Note2: The fasttext vector space model takes significantly longer to train (5-10 seconds vs 20 minutes when processing 700 files)

5 classes +-50 docs/class	Logistic Regression	SVM	Random Forest
Word2Vec	0.7955	0.7045	0.9545
Word2Vec - finetuned	0.7500	0.7955	0.8636
Glove	0.7561	0.7561	0.8537
Fasttext	0.3171 *	0.1951 *	0.9268

* look at page 72 of notebook for possible explanation : The lack of hyperparameter tuning makes the models underperform for complexity reasons

Experiment 2: Data Analysis

To gain a more accurate understanding of class performance, we expanded the experimental setup. This involved conducting a comprehensive hyperparameter search using cross-validation to optimize each classifier. Additionally, a more detailed evaluation report was generated, including per-class metrics and improved interpretability of the results.

Class Distribution and Mean Cosine Similarity

The initial evaluation revealed the following class distributions. Fairly regular with the limited amount of data we are currently working with.

- **Training Set:** {0: 42, 1: 32, 2: 34, 3: 38, 4: 26}
- **Test Set:** {0: 7, 1: 6, 2: 11, 3: 11, 4: 9}

For the vector space model (VSM) configurations, the mean cosine similarity in the training set varied according to the vector encoding used. Specifically:

- **word2vec-5-50.pkl:** Mean Cosine Similarity = 0.7533
- **glove-5-50.pkl:** Mean Cosine Similarity = 0.9608
- **fasttext-5-50.pkl** : Mean Cosine Similarity in Training Set: 0.9010

These metrics reflect the average similarity among the document vectors. Although all documents pertain to similar categories, the difference in cosine similarity values highlights the impact of the vector codification method on the VSM representations. I am unsure what the best results would be.

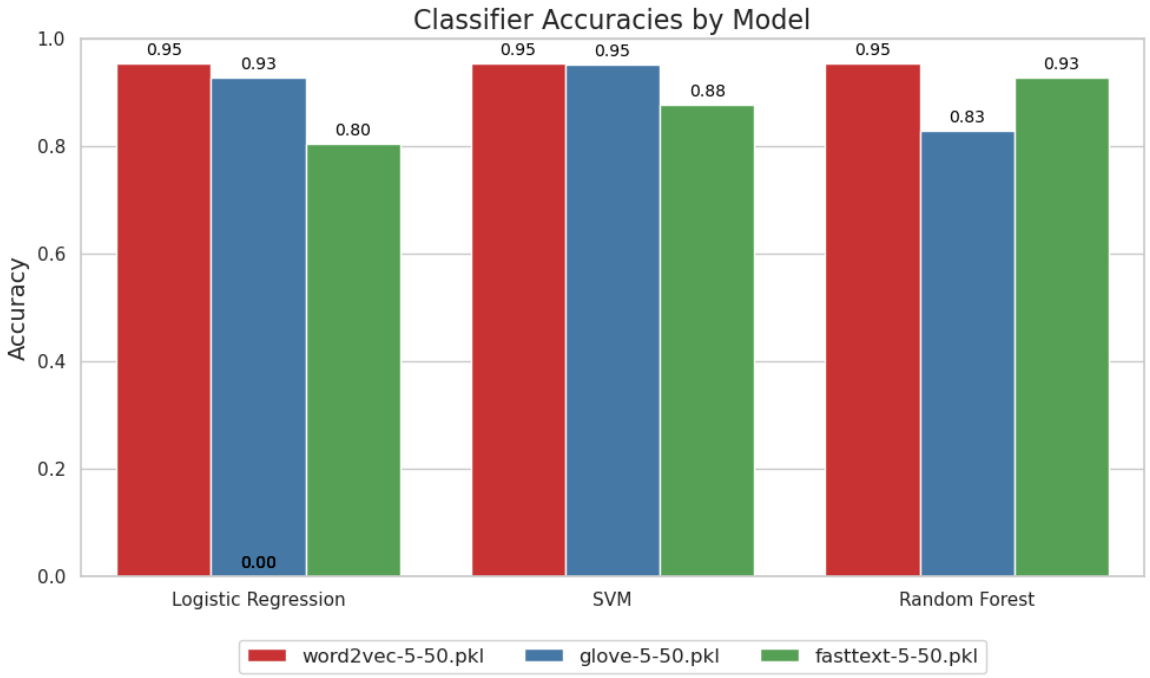
Hyperparameter Tuning and Classifier Optimization

To fine-tune the classifier models, cross-validation combined with a hyperparameter search (using GridSearchCV) was performed. The hyperparameter grids used were as follows:

```
"Logistic Regression":
```

```
            'C': [0.1, 1, 10],
            'max_iter': [500, 1000]
    "SVM":
            'C': [0.1, 1, 10],
            'kernel': ['linear', 'rbf']
    "Random Forest":
            'n_estimators': [50, 100, 200],
            'max_depth': [None, 10, 20]
```

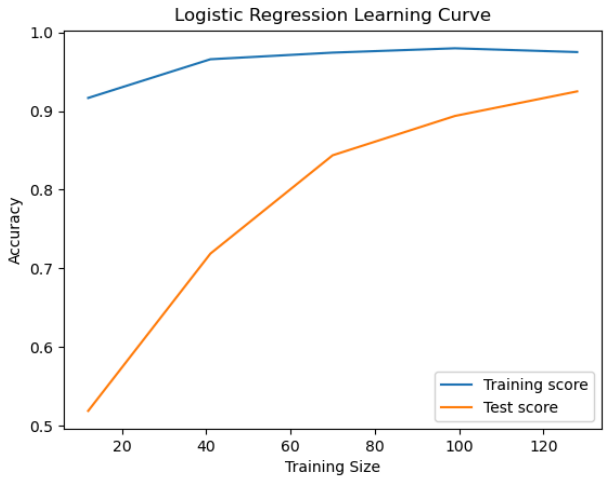
Different combinations of VSM configurations and classifier models were evaluated, and GridSearchCV helped identify the optimal parameters for each classifier. The training and test distributions were relatively balanced, with no significant disparities observed between the classes.



Analysis of Model Performance

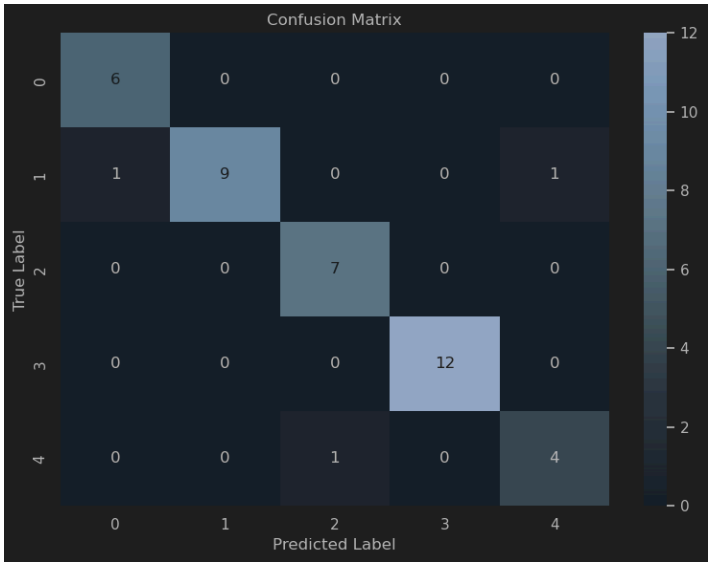
As anticipated, the mean cosine similarity varied between the different VSM models. Since cosine similarity measures how close the vector representations are, this variability is a key indicator of the model's behavior.

Learning curves were generated; however, these curves confirmed what was already expected without providing further actionable insights. They did, nonetheless, serve to validate the overall model trends.



When evaluating performance on a per-class basis, the **Random Forest classifier's report** was particularly striking. The confusion matrix clearly shows why the values for precision and recall are so high. It appears that the impressive results are not a byproduct of overfitting but may instead be due to the relatively small sample size used for testing and validation.

	precision	recall	f1-score	support
Batteries-non-rechargable-primary	0.86	1	0.92	6
cable-ties-zip-ties	1	0.82	0.9	11
coaxial-cables-rf	0.88	1	0.93	7
microphones	1	1	1	12
printers-label-makers	0.8	0.8	0.8	5
accuracy			0.93	41
macro avg	0.91	0.92	0.91	41
weighted avg	0.93	0.93	0.93	41



Next Steps: Expanded Evaluation Dataset

To address concerns about the limited amount of testing and validation data, a new dataset was created comprising 150 documents across 5 classes. The performance on this more robust dataset will be evaluated shortly, allowing us to better assess the classifiers' generalization capabilities.

Experiment 3: Expanding the database

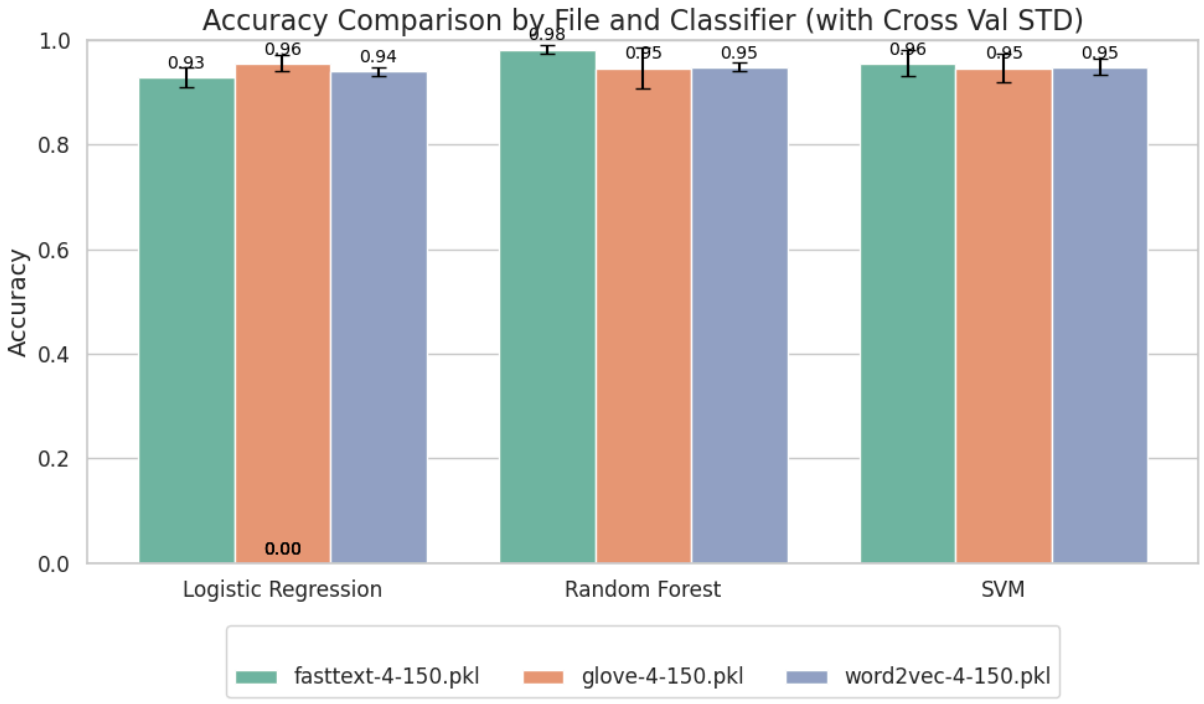
Unfortunately there was a codification error when transcribing the pdfs from Alarms. The pdfs where corrupted somehow in origin (when downloading most probably) and thus, from 150 there where only 38 left for processing.

I will divide this into two experiments then:

1st : 4 classes with around 150 documents each
2nd : 4 classes with around 150 documents each + 4 classes with around 40 documents each.

This is to see how the models perform when different input data sizes are passed. Maybe we can balance the classes somehow

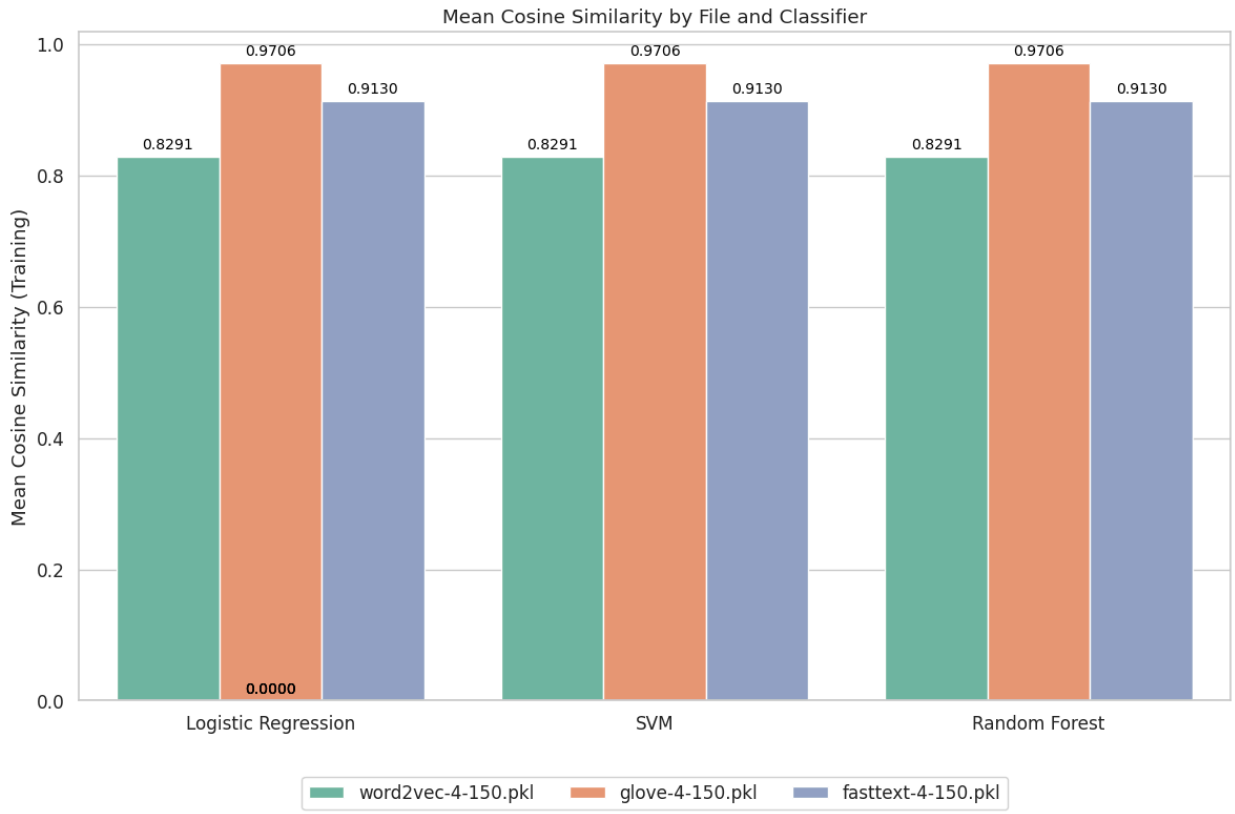
First:
We see good performance from the classifiers, having likewise results.



Model: word2vec-4-150.pkl - Logistic Regression (similar for every model). We can see much more testing data, which is what we wanted.

- **Training Set:** {0: 118, 1: 116, 2: 108, 3: 123}
- **Test Set:** {0: 33, 1: 29, 2: 28, 3: 27}

We do see a stark difference in the mean cosine similarity in between the encodings by word2vec and the other two models. I theorize this is the best approach, as it maintains more differences between different vectors whilst keeping the accuracies on par

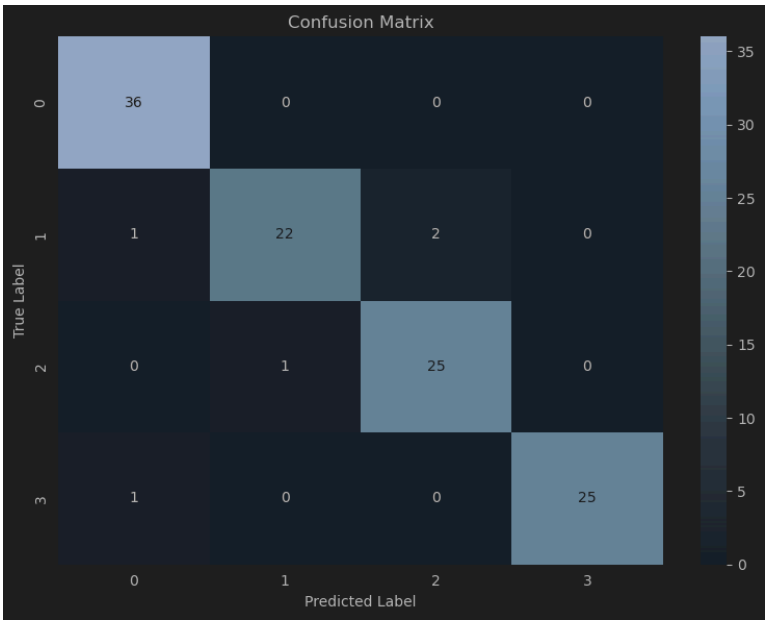


** i know this graph is not correct. fe de erratas

This is the confusion matrix and evaluation metrics divided by class. We see that the models perform really well for this low amount of classes, additionally, adding more data has certainly helped the tests.

Model: fasttext-4-150.pkl - SVM

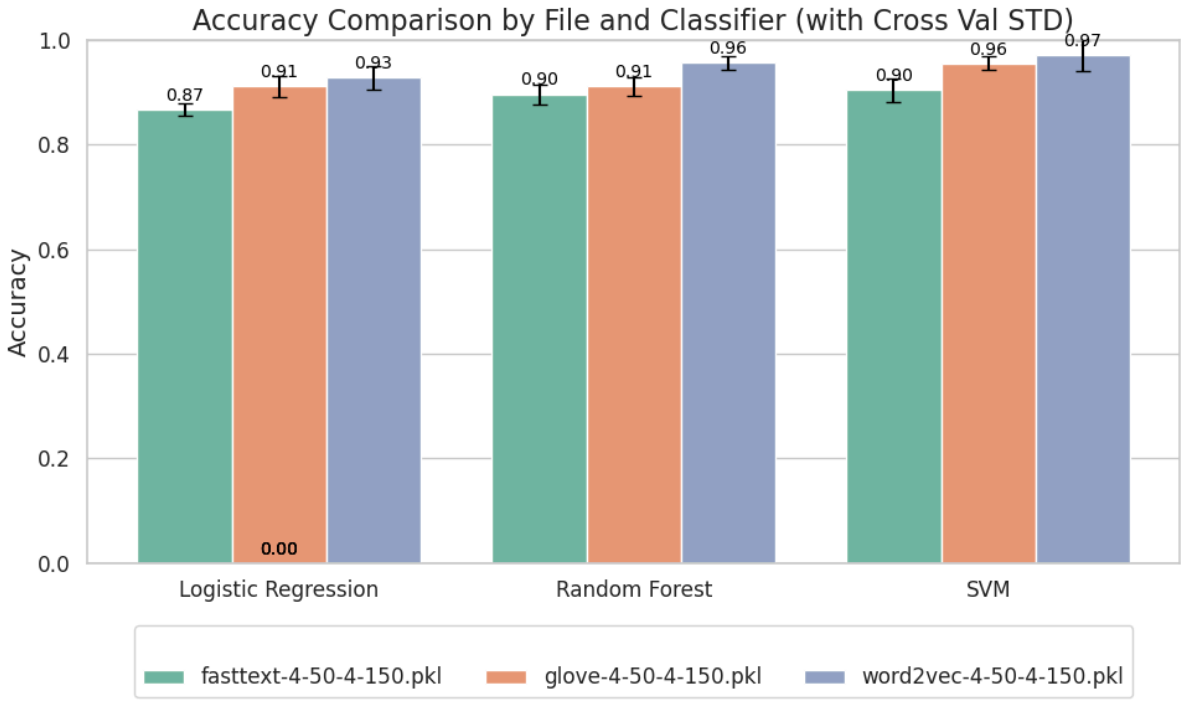
	precision	recall	f1-score	support
Batteries-non-rechargable-primary	0.95	1	0.97	36
Controller-accesories	0.96	0.88	0.92	25
microphones	0.93	0.96	0.94	26
speakers	1	0.96	0.98	26
accuracy			0.96	113
macro avg	0.96	0.95	0.95	113
weighted avg	0.96	0.96	0.96	113



Experiment 4: Class imbalance

In this experiment, we are going to build upon the previous one and expand the demo data. The idea is to evaluate document class imbalance. I will join together both datasets into one, and see their performance

- **cable-ties-zip-ties:** 39 documents
- **coaxial-cables-rf:** 45 documents
- **printers-label-makers:** 45 documents
- **Speakers:** 150 documents
- **Microphones:** 171 documents
- **Controller Accessories:** 150 documents
- **Batteries (Non-Rechargeable):** 152 documents



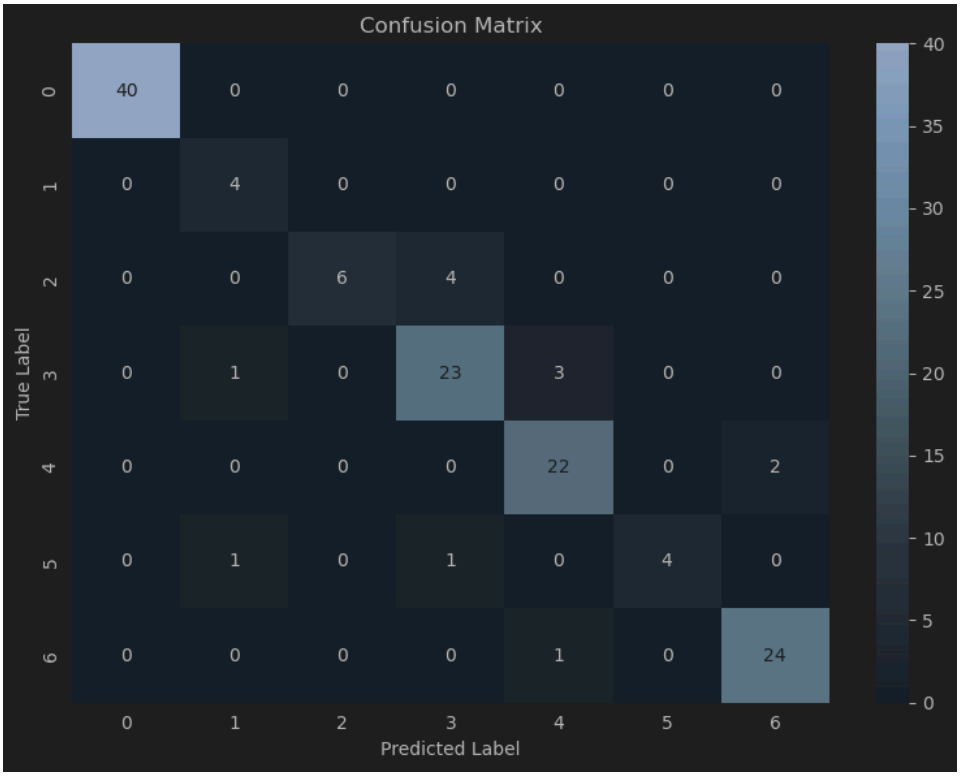
Accuracies drop, as expected

Class distribution in training set: {0: 109, 1: 28, 2: 35, 3: 118, 4: 97, 5: 29, 6: 125}

Class distribution in test set: {0: 40, **1: 4, 2: 10**, 3: 27, 4: 24, **5: 6**, 6: 25}

2025-04-12 11:42:26,594 - INFO - SVM Classification Report:

	precision	recall	f1-score	support
Batteries-non-rechargable-primary	1	1	1	40
Cable-ties-zip-ties	0.67	1	0.8	4
Coaxial-cables-rf	1	0.6	0.75	10
Controller-accesories	0.82	0.85	0.84	27
Microphones	0.85	0.92	0.88	24
Printers-label-makers	1	0.67	0.8	6
Speakers	0.92	0.96	0.94	25
accuracy			0.9	136
macro avg	0.89	0.86	0.86	136
weighted avg	0.91	0.9	0.9	136



Classifier Selection and Final Choice

In the initial phase of experimentation, we explored three widely used classification models: **Logistic Regression**, **Support Vector Machines (SVM)**, and **Random Forests**. Each was selected for its distinct strengths. Logistic Regression was chosen for its simplicity, interpretability, and scalability, serving as a reliable baseline. SVMs, known for their performance in high-dimensional spaces such as those generated by text data, were evaluated for their ability to handle complex boundaries. Random Forests were included due to their robustness, capacity to model non-linear relationships, and natural handling of multi-class scenarios.

Pros and Cons Overview:

- **Logistic Regression**
  Pros: Fast training, scalable to large datasets, interpretable coefficients, works well with linearly separable classes.
  Cons: Limited in capturing complex patterns or non-linear relationships.

- **Support Vector Machines (SVM)**
  Pros: Excellent for high-dimensional data, effective with clear margins of separation, robust to overfitting in lower-dimensional space.
  Cons: Computationally expensive for large datasets, less interpretable, sensitive to

parameter tuning.

- **Random Forests**
  Pros: Handles non-linear relationships well, resistant to overfitting, provides feature importance, works well with imbalanced data.
  Cons: Higher memory usage, slower to train and predict with large numbers of trees, less interpretable than simpler models.

Despite achieving comparable performance across all three models on the current dataset of 600 documents and 5 classes, **Logistic Regression** was ultimately selected as the preferred model moving forward. This decision was guided by its **computational efficiency, scalability, and ease of integration**—especially relevant as the dataset is expected to grow significantly to at least 15,000 documents and 30 classes. Given these scaling requirements, Logistic Regression provides a balanced trade-off between performance and practicality for future development.

Next steps

1. **Define Evaluation Metrics and Results**: Establish how the system's performance will be measured — for instance, through retrieval accuracy, relevance of results, or precision/recall based on labeled queries.
2. **Embedding Choice**: For consistency and interpretability, we will proceed with **Word2Vec** as the embedding model, given its solid performance and ease of analysis in our current setup.
3. **Handling Dataset Size:** The current workflow is scalable
4. How should the workflow be presented, modularity in current approach.
5. Writting

Literature review for Query VSM

Study of Word2Vec, FastText, and Glove Embeddings

Full title : **Comparative Study of Word2Vec, FastText, and Glove Embeddings for Synonym Identification in Bugis Language**

The results show that Word2Vec has the highest accuracy of 0.9412, followed by Glove with 0.9353 and FastText with 0.9262. Additional experiments with cosine similarity revealed that Glove performed best for predicting synonyms across all word occurrence frequency categories, while FastText and Word2Vec had inferior results. These findings indicate that although high training accuracy is achieved, it does not always imply that the model will excel in predicting synonyms.

https://ieeexplore.ieee.org/document/10942212

Bag of Tricks for Efficient Text Classification

Unlike unsupervised trained word vectors from word2vec, our word features can be averaged together to form good sentence representations. In several tasks, fastText obtains performance on par with recently proposed methods inspired by deep learning, while being much faster

https://arxiv.org/pdf/1607.01759

Word and Sentence similarity

This paper presented an approach to calculate the semantic similarity between two words, sentences or paragraphs. The algorithm initially disambiguates both the sentences and tags them in their parts of speeches. The disambiguation approach ensures the right meaning of the word for comparison. The similarity between words is calculated based on a previously established edge-based approach.

not sure how applicable this is though
https://arxiv.org/pdf/1802.05667

Sentence Embedding Index

Building a sentence embedding index with fastText and BM25. Standard text token searches are getting caught up with synonyms and non-exact matches. Word embeddings are a great way to find similar results that don't match exactly.

https://medium.com/data-science/building-a-sentence-embedding-index-with-fasttext-and-bm25-f07e7148d240

Fast and scalable neural embedding models for biomedical sentence classification

benchmark for sequential sentence classification
https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2496-4

Writing

1. Are we writing a paper in the end?
2. Is the paper joining both research and thesis? or separate topics

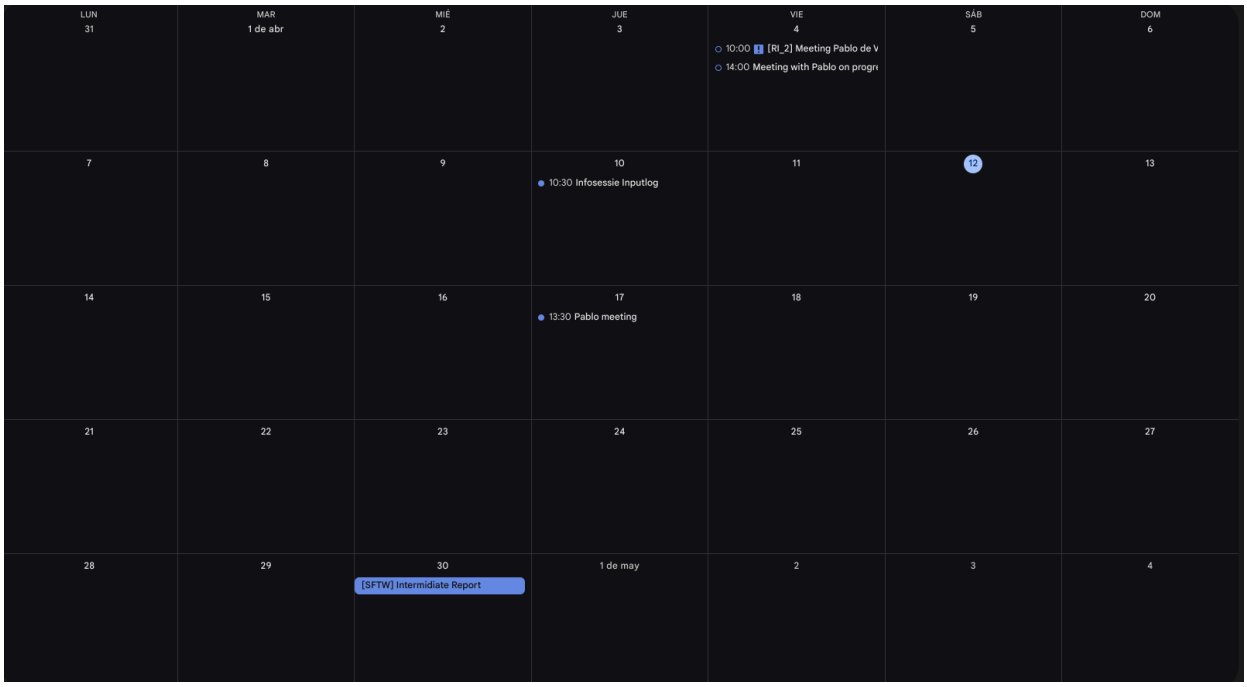
Timeline

From now until **May 17**, the focus is on finalizing the Query-to-VSM pipeline — implementing the search logic, testing it with sample queries, and preparing a brief demo for review.

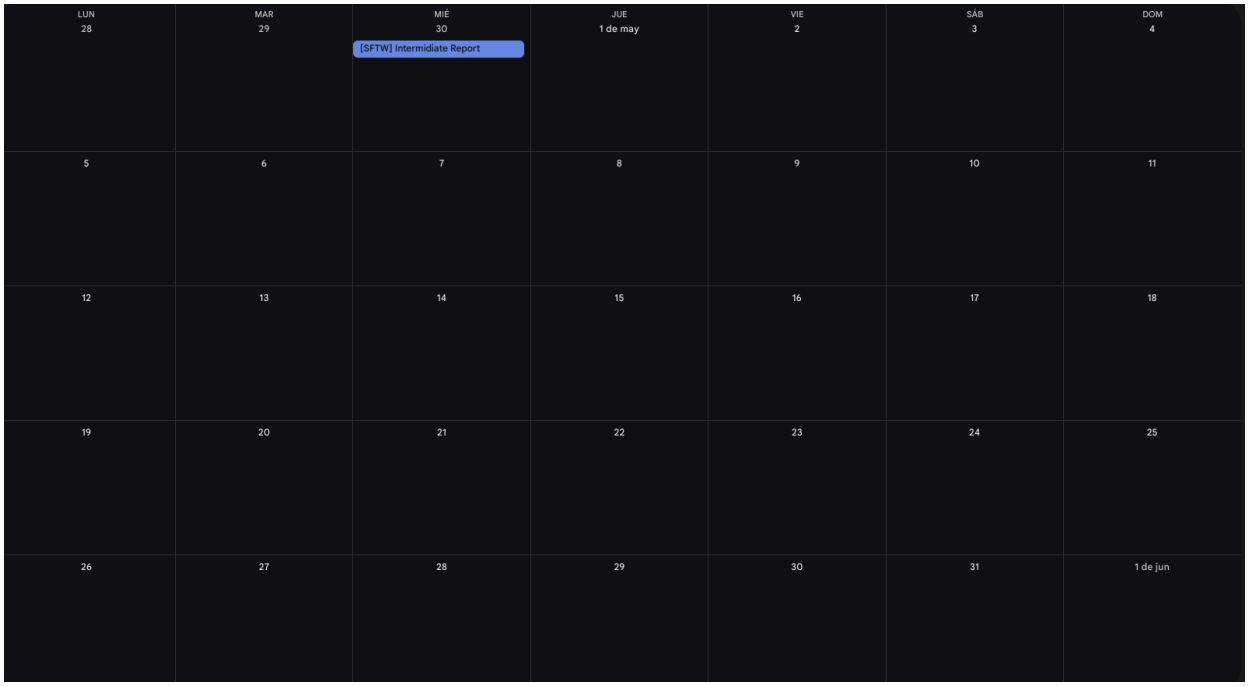
On **May 17**, there will be a meeting to present progress, share preliminary results, and gather feedback.

From **May 18 to June 16**, the goal is to complete the writing phase: drafting the report, refining the content, and applying feedback in preparation for final submission.

April 2025



May 2025


LUN 28
MAR 29
MIÉ 30
[SFTW] Intermidiate Report
JUE 1 de may
VIE 2
SÁB 3
DOM 4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
1 de jun

June 2025


LUN 26
MAR 27
MIÉ 28
JUE 29
VIE 30
SÁB 31
DOM 1 de jun
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
1 de jul
2
3
4
5
6