Spam/Ham Classification: TF-IDF/CountVectorizer vs. Word2Vec/FastText

Introduction

The user is working on a spam/ham classification problem with a dataset of 5,000 training examples. The goal is to distinguish between spam (unwanted messages) and ham (legitimate messages). The key question is whether to use traditional methods like **TF-IDF/CountVectorizer** or embedding-based approaches like **Word2Vec/FastText**. Below is a detailed analysis of the pros and cons of each approach in the context of this problem.

Why TF-IDF/CountVectorizer Are Better for This Task

1. Small Dataset Advantage

- **Word2Vec/FastText**: These methods require large corpora (e.g., millions of sentences) to learn meaningful word embeddings. With only 5,000 samples, the embeddings may not capture reliable semantic patterns.
- **TF-IDF/CountVectorizer**: These methods work robustly even with small datasets because they rely on simple word-frequency statistics, making them more suitable for this scenario.

2. Spam Detection Relies on Keywords

- Spam messages often contain specific "trigger words" (e.g., "free," "win," "click now," "urgent").
- **TF-IDF** directly highlights these important words by weighting their frequency, making it ideal for keyword-driven tasks like spam detection.

3. Speed and Simplicity

- **TF-IDF + Logistic Regression/SVM**: Trains in seconds and requires minimal computational power.
- **Word2Vec/FastText**: Requires extra steps (e.g., averaging word vectors for sentences) and more complex models (e.g., neural networks), which risk overfitting on small datasets.

4. Interpretability

- **TF-IDF**: Allows you to inspect which words are most predictive (e.g., "$$$" or "lottery" for spam), which is useful for debugging and understanding the model.
- **Embeddings (Word2Vec/FastText)**: Produce abstract vectors that are harder to interpret.

5. Out-of-Vocabulary (OOV) Handling

- While **FastText** handles OOV words via subwords, spam detection rarely benefits from subword semantics.
- Misspelled spam words (e.g., "pr0n" or "vi@gra") can still be detected via **TF-IDF's** exact token matching.

When to Consider FastText

While **TF-IDF/CountVectorizer** is recommended for this task, **FastText** might be a viable alternative in the following scenarios:

1. **Noisy Text**: If your spam data contains many rare or misspelled words (e.g., "w4tch fr33 m0vies"), FastText's subword embeddings can generalize better.
2. **Deep Learning Experimentation**: If you want to experiment with a deep learning approach (e.g., simple neural networks) for slightly better accuracy, ensure you have enough data to avoid overfitting.
3. **Multilingual Support**: If you need to handle multiple languages, FastText supports subwords for morphologically rich languages.

Recommendation

For a dataset of 5,000 samples, **TF-IDF/CountVectorizer** with a simple model like **Logistic Regression** or **SVM** is the preferred approach. It is efficient, works well with smaller datasets, and provides interpretability. However, if computational resources allow, experimenting with both approaches can provide additional insights. Prioritize traditional methods for this scenario due to their simplicity and effectiveness.