Modified DFS-based term weighting scheme for text classification

Abhishek Kumar (181CO201), Sasidhar Swarangi (181CO245)

Objectives

- Implement Modified DFS-based term frequency (TF-MDFS) weighting scheme.
- Compare and visualize the performance of TF-MDFS and other state-of-the-art term weighting schemes.
- Apply TF-MDFS to more sophisticated text representations like N-grams, Skip-grams and Continuous Bag of Words.

Vector Space Model

- ML Models cannot operate directly on the textual data of the documents.
- Using term-weighting schemes, the documents are transformed into vectors which can then be fed to models.
- Examples: Term frequency, Term frequency-inverse document frequency weights
- **Designing appropriate term-weighting schemes is crucial.**

```
corpus = [
  [
    'this', 'is', 'the',
    'first', 'document'
  ],
  [
    'this', 'is', 'the',
    'second', 'second',
    'document'
  ],
  [
    'and', 'the', 'third',
    'one'
  ],
  [
    'is', 'this', 'the',
    'first', 'document'
  ]
]

# Corresponding Term Frequency Vector
[
  [0, 1, 1, 1, 0, 0, 1, 0, 1],
  [0, 1, 0, 1, 0, 2, 1, 0, 1],
  [1, 0, 0, 0, 1, 0, 1, 1, 0],
  [0, 1, 1, 1, 0, 0, 1, 0, 1]
]
```

Distinguishing Feature Selector

An ideal feature selection method should assign high scores to distinctive terms while assigning lower scores to irrelevant features. In particular, if a term:

- Occurs frequently in a single class and does not occur in other classes, it is distinctive.
- Occurs frequently in all classes, it is irrelevant.
- Occurs rarely in a single class and does not occur in other classes, it is irrelevant.
- Occurs in some of the classes, it is relatively distinctive.

TF-DFS was not effective as it assigns scores between 0.5 to 1.0 and does not demonstrate specificity adequately.

$$DFS(t_i) = \sum_{j=1}^{q} \frac{P(c_j|t_i)}{P(\bar{t}_i|c_j) + P(t_i|\bar{c}_j) + 1},$$

Modified DFS-based Term Frequency

- A distinguishing feature must predict both true-positives and true-negatives.
- Modified DFS is decomposed into q class-specific scores and is calculated from both “positive” and “negative” occurrences.
- Modified-DFS based term frequency outperformed all existing term weighting schemes in accuracy.

$$tf-mdfs(t,d) = mdfs(t) \times tf(t,d)$$

$$mdfs(t_i) = \sum_{j=1}^{q} w_{ij} mdfs(t_i, c_j)$$

$$w_{ij} = log(1 + \frac{df(t_i, c_j)}{max(1, df(\bar{t_i}, c))} \times \frac{df(\bar{t_i}, \bar{c_j})}{max(1, df(t_i, \bar{c_j}))})$$

$$mdfs(t_i, c_j) = \frac{P(c_j|t_i)P(\bar{c_j}|\bar{t_i})}{1 + P(\bar{c_j}|t_i) + P(c_j|\bar{t_i})}$$

Computing TF-MDFS

Algorithm 1 TF–MDFS Learning (D)

Input: D-a training document set

Output: All term weights

1: **for** each term t_i $(i = 1, 2, \cdots, m)$ and each class c_j $(j = 1, 2, \cdots, q)$ **do**

2: Calculate $MDFS_{cs}(t_i, c_j)$ by Eq. (3)

3: Calculate w_{ij} by Eq. (5)

4: **end for**

5: **for** each term t_i $(i = 1, 2, \cdots, m)$ **do**

6: Calculate $MDFS(t_i)$ by Eq. (4)

7: Calculate $W_{\text{TF-MDFS}}(t_i)$ by Eq. (6)

8: **end for**

9: **return** $W_{\text{TF-MDFS}}(t_i)$ $(i = 1, 2, \cdots, m)$

Pros and Cons of TF-MDFS

Pros:

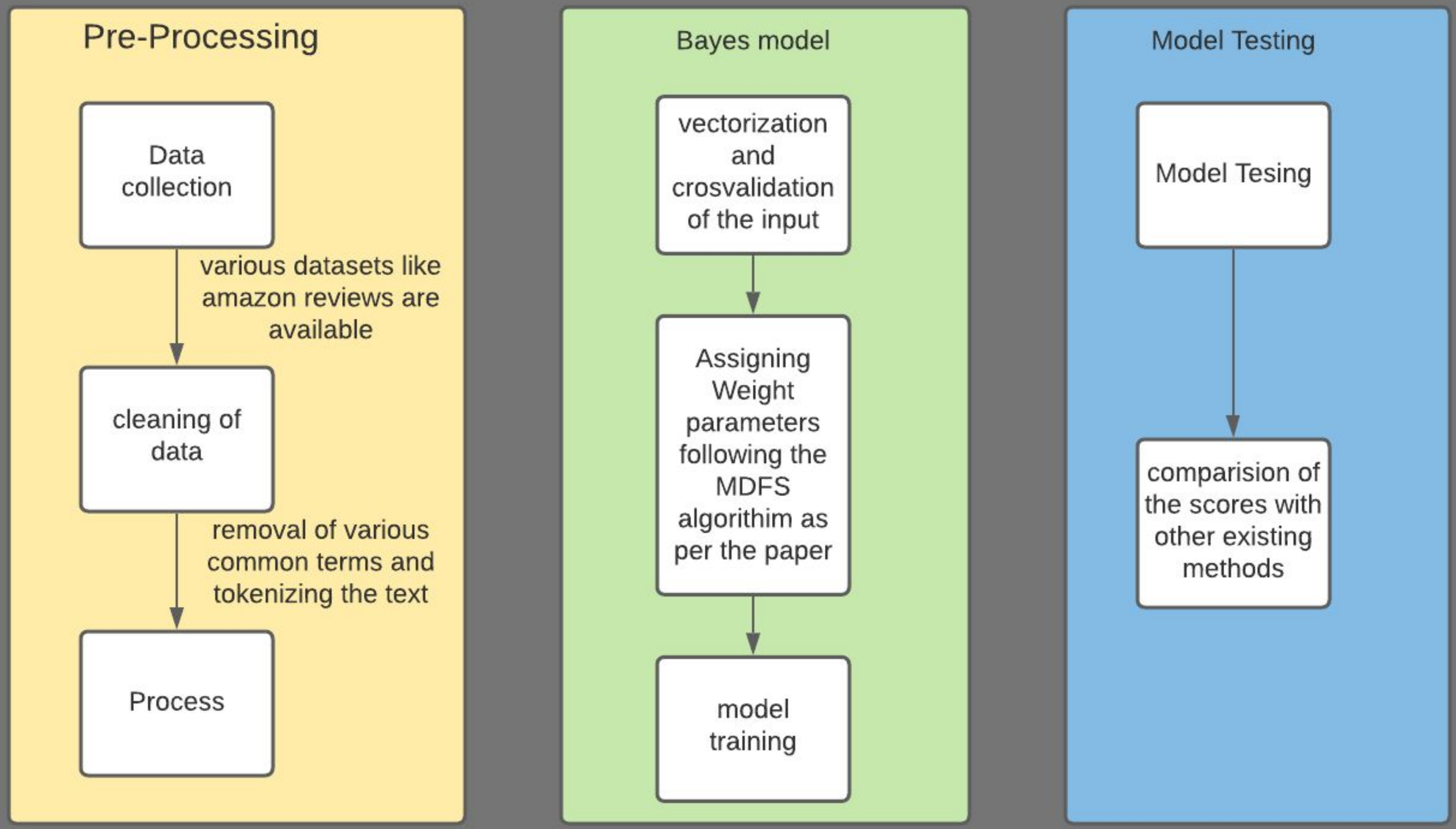
- Utilizes the class labels as well to compute weights.
- Outperforms state-of-the-art term weighting schemes.

Cons:

- Computing TF-MDFS takes O(N * M) time where N is the number of classes and M is the number of documents, which can be infeasible for datasets with large number of classes.

Implementation (Open in Colab)

1) Load the datasets.
2) Preprocess the corpus - tokenization, word stemming, lemmatization and drop the stopwords.
3) Calculate weights (TF, TF-IDF and TF-MDFS) for the Vector Space Model.
4) Train Naive Bayes, Support Vector Machine and Logistic Regression on each term-weighting scheme.
5) Compute accuracy for each model, scheme and dataset.


Pre-Processing
Data collection
various datasets like amazon reviews are available
cleaning of data
removal of various common terms and tokenizing the text
Process
Bayes model
vectorization and crosvalidation of the input
Assigning Weight parameters following the MDFS algorithim as per the paper
model training
Model Testing
Model Tesing
comparision of the scores with other existing methods

Results

Dataset	TF (%)	TF-IDF (%)	TF-MDFS (%)	Improvement over TF (%)
Amazon Review Dataset	81.783	82.483	82.783	1.000
Polarity v2.0 Movie Review Dataset	79.833	81.500	81.667	1.833
RSS Feed Topics Dataset	64.681	64.965	65.362	0.681

Therefore, we find that TF-MDFS outperforms the existing term weighting schemes.

Our Contribution

- Authors believed that applying TF-MDFS to more sophisticated text representations will make the improvement stronger.
- We followed through, applying TF-MDFS to bigram representation of same datasets.
- We also wanted to apply it to Word2Vec but couldn’t implement.

Accuracies for Bigram

Dataset	TF (%)	TF-IDF (%)	TF-MDFS (%)	Improvement over TF (%)
Amazon Review Dataset	83.500	84.600	84.933	1.433
Polarity v2.0 Movie Review Dataset	81.667	82.167	82.833	1.167
RSS Feed Topics Dataset	65.730	63.574	67.972	2.241

Using TF-MDFS on more sophisticated text representations improves the advantage even further!