UNIVERSITY of WISCONSIN UWMILWAUKEE | LUBAR COLLEGE OF BUSINESS

Final Project

Title: Sentiment Analysis of Financial Data

Objective: To analyze financial data and determine sentiment trends using natural language processing and machine learning techniques for sentiment classification.

The project will include data collection, preprocessing, feature engineering, model selection, training, evaluation, and deployment steps.

1. Introduction

Sentiment analysis plays a critical role in understanding the emotions and opinions expressed in text data. In the financial domain, sentiment analysis can provide valuable insights into market trends and investor behavior. This project aims to develop a sentiment analysis system for financial data using machine learning techniques.

2. Methodology

The project will follow these steps:

a. Data Collection: Obtain financial data in CSV format.

b. Preprocessing: Clean the collected text data and perform tokenization, stemming, and lemmatization.

c. Feature Engineering: Convert the preprocessed text into numerical features using techniques like Bag-of-Words, TF-IDF, or word embeddings.

d. Model Selection: Choose a suitable machine learning or deep learning model for sentiment classification.

e. Training: Train the selected model on the dataset, optimizing hyperparameters for performance.

f. Evaluation: Assess the model's performance on an unseen test set using metrics like accuracy, precision, recall, F1 score, confusion matrix, and AUC-ROC curve.

g. Deployment: Deploy the trained model as a RESTful API or integrate it directly into an application or service.

3. Possible Resources
a. Financial Sentiment Analysis | Kaggle https://www.kaggle.com/datasets/sbhatti/financial-sentiment-analysis?select=data.csv
b. Basic to advanced models for Sentiment Analysis | Kaggle
https://www.kaggle.com/code/sohailshaik272/basic-to-advanced-models-for-sentiment-analysis

4. Sample R Code

The sample R code provided demonstrates how to build a sentiment analysis system using a financial CSV dataset with tidytext and randomForest packages for preprocessing and model creation, respectively.

5. Conclusion

Upon completion, this project will result in a sentiment analysis system capable of analyzing financial data and determining sentiment trends. By following the outlined methodology and utilizing the provided resources and sample code, students can gain hands-on experience in developing a sentiment analysis system using machine learning techniques.

==

7 phases:

To divide the sentiment analysis project into a phased approach, you can follow these phases:

Phase 1: Data Collection

Collect data from a relevant source (e.g., financial CSV or social media data)

Create a dataset suitable for sentiment analysis

Phase 2: Data Preprocessing

Clean the text data by removing special characters, emojis, URLs, and stop words

Convert the text to lowercase for consistency

Perform tokenization, stemming or lemmatization

Phase 3: Feature Engineering and Dataset Splitting

Convert preprocessed text into numerical features using techniques like Bag-of-Words, TF-IDF, or word embeddings

Split the dataset into training, validation, and test sets

Phase 4: Model Selection and Training

Choose a suitable machine learning or deep learning model for sentiment classification

Train the selected model on the training set, tuning hyperparameters to optimize performance

Regularly evaluate the model on the validation set to prevent overfitting and ensure generalization

Phase 5: Model Evaluation and Fine-tuning

Test the trained model on an unseen test set to assess its performance

Use evaluation metrics like accuracy, precision, recall, F1 score, confusion matrix, and AUC-ROC curve

Fine-tune the model based on the evaluation results by adjusting its parameters or trying different algorithms

Phase 6: Model Deployment and Integration

Deploy the trained model as a RESTful API or integrate it directly into an application or service

Set up a data pipeline for real-time sentiment predictions

Phase 7: Results Visualization and Interpretation

Visualize and analyze the sentiment trends and patterns

Interpret the results to understand the factors influencing sentiment and improve the model

===

An example:

• R code :

R code for a sentiment analysis system using a financial CSV dataset. This example uses the tidytext and randomForest packages to preprocess the data and create a Random Forest model for sentiment classification.

```
# Load necessary libraries

library(tidyverse)

library(tidytext)

library(textdata)

library(tm)

library(randomForest)


# Data Collection

# Load CSV file (replace 'path/to/your/csv' with the actual file path)

data <- read.csv('path/to/your/csv')


# Preprocessing

data_clean <- data %>%
  unnest_tokens(word, your_text_column) %>% # Replace 'your_text_column' with the actual column
name
  mutate(word = tolower(word)) %>%
  anti_join(stop_words) %>%
  filter(!grepl('[0-9]', word)) %>%
  filter(!grepl('[[:punct:]]', word))

# Feature Engineering - Term Frequency-Inverse Document Frequency (TF-IDF)
corpus <- VCorpus(VectorSource(data_clean$word))
tdm <- TermDocumentMatrix(corpus)
matrix <- as.matrix(tdm)
tfidf <- weightTfIdf(matrix)

# Model Selection - Random Forest
set.seed(42)
train_index <- sample(seq_len(nrow(tfidf)), size = floor(0.8 * nrow(tfidf)))
train_set <- tfidf[train_index, ]
test_set <- tfidf[-train_index, ]

# Training
rf_model <- randomForest(your_target_column ~ ., data = train_set, ntree = 100) # Replace
'your_target_column' with the actual target column name

# Evaluation
predictions <- predict(rf_model, test_set)
accuracy <- mean(predictions == test_set$your_target_column) # Replace 'your_target_column' with the
actual target column name
print(paste("Accuracy:", accuracy))

# Deployment
```

```r
# Save the trained model

saveRDS(rf_model, 'rf_model.rds')


# Load the model and make predictions

loaded_model <- readRDS('rf_model.rds')

new_predictions <- predict(loaded_model, new_data)
```

This example assumes you have a CSV file with a text column for sentiment analysis and a target column with sentiment labels (e.g., positive, negative, or neutral). Adjust the column names and file paths accordingly. Also, note that deployment in this example saves the model as an RDS file. You can deploy the model as a RESTful API using packages like plumber or integrate it directly into your application or service.