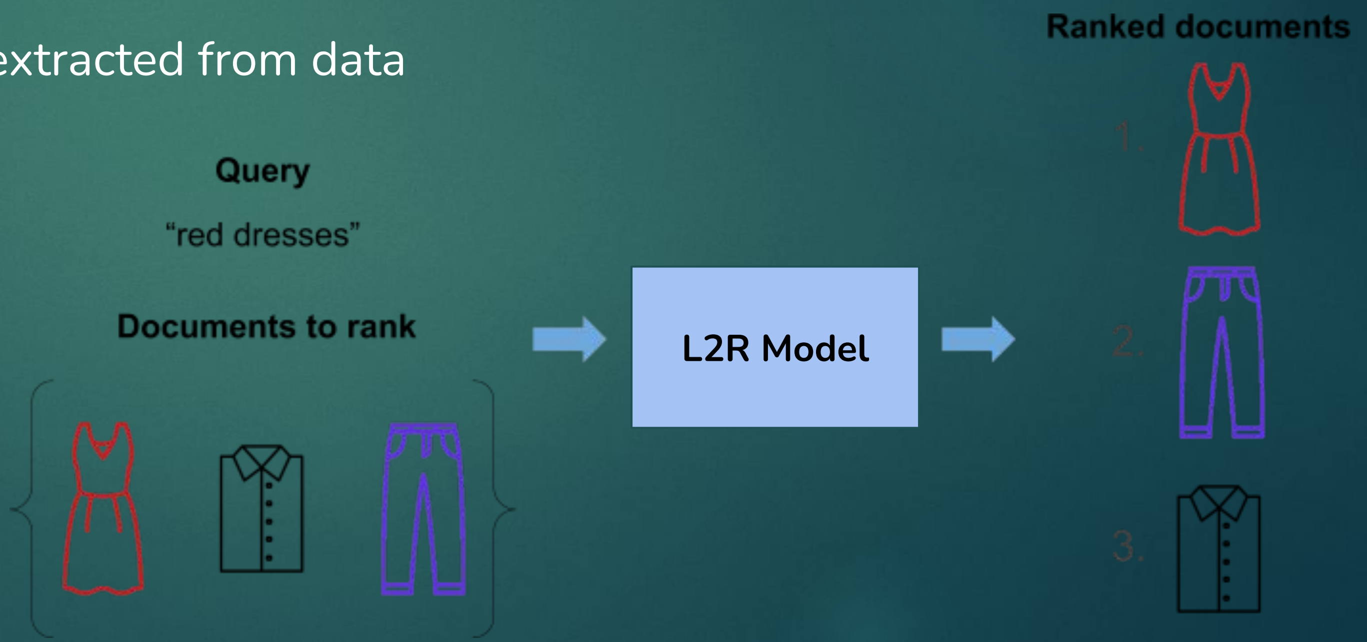
Ranking Source Code Files for Bug Prediction: An L2R Approach

CREATED BY: TASHAN GILLEM

Learning to Rank (L2R)

- A machine learning model that predicts the most relevant order of a set of items
- L2R models can assess and rank code files by their likelihood of containing bugs
- They are trained using features extracted from data



Introduction

- Early bug detection is key to saving time and ensuring product quality
- This project aims to use machine learning to predict and prioritize code files based on bug likelihood
- These techniques are applied to an open source repository due to its vast amount of commit history

Data Collection and Feature Engineering

Data Collection:

- Extracted approximately 1500 commits using GitHub API

Data Cleaning:

- Filtered out code files irrelevant to bugs (.mdf, .png, and Dockerfile, etc.)

Feature Engineering:

- Reflected commit activity such as commit timestamps, file change magnitude, and frequency of commits per file

```
def clean_data(df):
    df = handle_missing_data(df)
    df = normalize_text_data(df)
    df = filter_relevant_files(df)
    df = detect_and_handle_outliers(df)
    df = remove_duplicates(df)
    return df


1 usage
def engineer_features(df):
    df = create_bug_fix_feature(df)
    df = create_change_magnitude_feature(df)
    df = create_commit_frequency_feature(df)
    df = create_commit_time_feature(df)
    return df


1 usage
def preprocess_data(df):
    df = encode_categorical_data(df)
    df = encode_numerical_data(df)
    return df
```

Model Development and Implementation

Choosing a Model:

- L2R model is effective in ranking tasks and handling large datasets

Implementing with LightGBM:

- LightGBM is efficient in supporting L2R tasks

Training and Tuning the Model:

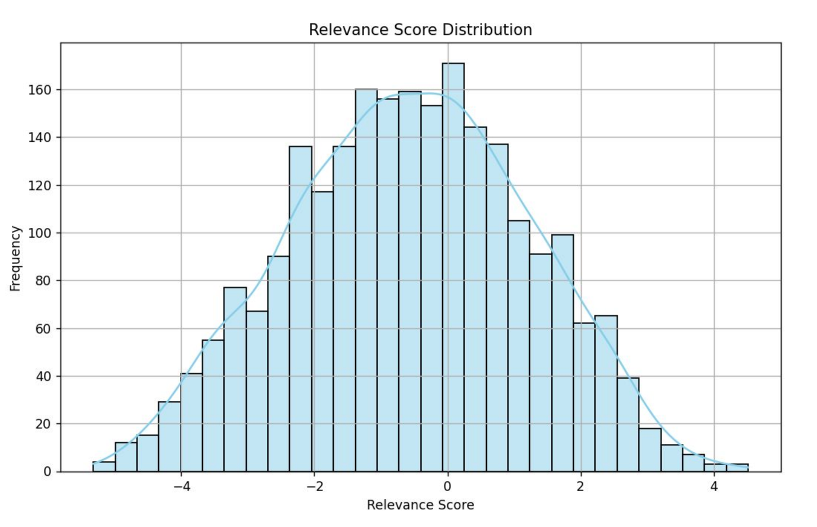
- Trained the model using the features derived from the dataset

```
def prepare_and_group_data(df):
    grouped = df.groupby('sha')
    X = df.drop(['additions', 'deletions', 'sha', 'message', 'author', 'date',  'is_bug_related'], axis=1)
    y = df['is_bug_related']    # Relevance label
    groups = grouped.size().to_list()
    return X, y, groups


1 usage
def create_lgb_dataset(X, y, groups):
    return lgb.Dataset(X, label=y, group=groups)


1 usage
def train(train_data, feature_names):
    params = {
        'objective': 'lambdarank',
        'metric': 'ndcg',
        'learning_rate': 0.1,
        'num_leaves': 31,
        'verbose': 1
    }
    gbm = lgb.train(params, train_data, num_boost_round=100, feature_name=feature_names)
    return gbm
```

Evaluation and Results

- The model achieved an almost perfect score of NDCG score 0.9688
- The relevance scores varied between -5.319 to 4.515

	file	relevance_score
0001	packages/trpc/server/routers/viewer/teams/_router.tsx	4.515340957046679
0002	apps/web/components/settings/DisableTwoFactorModal.tsx	4.324266617910534
0003	packages/trpc/server/routers/viewer/availability/schedule/_router.tsx	4.324266617910534
0004	apps/web/pages/[user].tsx	4.096295487528863
0005	apps/web/lib/withEmbedSsr.tsx	3.985356560178883
0006	packages/features/bookings/Booker/components/Header.tsx	3.981970460873396
0007	packages/app-store/googlecalendar/lib/CalendarService.ts	3.7377406247210065
0008	packages/trpc/server/routers/viewer/workflows/update.handler.ts	3.6974445489496723
0009	packages/core/getBusyTimes.ts	3.643701925724574
0010	apps/web/playwright/event-types.e2e.ts	3.643701925724574



Questions?