Dokumentacja Konserwacyjna – PhishGuardian

PhishGuardian to rozszerzenie przeglądarki stworzone w celu wykrywania i zarządzania podejrzanymi wiadomościami e-mail. Aby zapewnić jego prawidłowe działanie oraz umożliwić przyszłe aktualizacje i poprawki, przygotowano dokumentację konserwacyjną.

1. Struktura projektu:

Katalog backend:

- app.py: Główny plik aplikacji Flask. Odpowiada za obsługę zapytań HTTP oraz komunikację z serwerem e-mail.
- spam_classifier_model.pkl: Wstępnie wytrenowany model uczenia maszynowego do klasyfikacji wiadomości e-mail.
- vectorizer.pkl: Wstępnie wytrenowany wektoryzator do przekształcania treści wiadomości e-mail w format odpowiedni dla klasyfikatora.
- source.txt: Zawiera link, z którego pobrano zestawy danych.
- lingSpam.csv, enronSpamSubset.csv, completeSpamAssasin.csv: Zestawy danych używane do trenowania modelu (użyto modelu Random Forest).
- data_join.py: Skrypt, który łączy trzy zestawy danych w jeden plik CSV o nazwie joined_data.csv.
- joined_data.csv: Połączony zestaw danych wynikający z data_join.py.
- ML.ipynb: Notatnik Jupyter zawierający informacje o uczeniu maszynowym oraz szczegóły dotyczące wektoryzatora.
- requirements.txt: Plik zawierający listę wymaganych pakietów Pythona.

Katalog extension:

- popup.html: Główny plik HTML interfejsu rozszerzenia.
- popup.js: JavaScript obsługujący interakcje w popupie, takie jak logowanie, pobieranie wiadomości e-mail i obsługę odpowiedzi.
- background.js: JavaScript zarządzający zadaniami w tle rozszerzenia, takimi jak otwieranie popupa.
- styles.css: CSS zawierający style interfejsu rozszerzenia.
- manifest.json: Plik konfiguracyjny dla rozszerzenia Chrome.
- images/icon16.png, images/icon48.png, images/icon128.png: Ikony używane w rozszerzeniu.

2. Środowisko programistyczne:

Wymagania wstępne:

- Python 3.6+
- Flask
- scikit-learn
- Przeglądarka Chrome

Instalacja:

1. Sklonuj repozytorium:
   - git clone https://git.wmi.amu.edu.pl/s452649/PhishGuardian.git
   - cd PhishGuardian/backend

2. Zainstaluj wymagane pakiety Pythona:
   - pip install -r requirements.txt

3. Uruchom backend Flask:
   - python app.py

Konfiguracja rozszerzenia:

1. Otwórz Chrome i przejdź do chrome://extensions/.
2. Włącz "Tryb deweloperski" przełącznikiem w prawym górnym rogu.
3. Kliknij "Wczytaj rozszerzenie bez pakietu" i wybierz katalog extension w katalogu PhishGuardian.

3. Testowanie:

Testy manualne:

1. Upewnij się, że backend Flask działa poprawnie.
2. Otwórz rozszerzenie w Chrome i zaloguj się za pomocą danych logowania do Outlook.
3. Użyj przycisku "Fetch Emails", aby pobrać wiadomości e-mail.
4. Przetestuj funkcje "Classify Email", "Mark as Safe" i "Delete Email" na różnych wiadomościach.

Testy automatyczne:

1. Dodaj testy jednostkowe dla funkcji klasyfikacji wiadomości e-mail w app.py.
2. Skorzystaj z narzędzi takich jak pytest do uruchamiania testów.

4. Aktualizacje i utrzymanie:

Aktualizacja modelu:

1. Zaktualizuj zestawy danych (lingSpam.csv, enronSpamSubset.csv, completeSpamAssasin.csv) i użyj data_join.py do połączenia ich w joined_data.csv.
2. Przetrenuj model używając Jupyter Notebook (ML.ipynb), zaktualizuj pliki spam_classifier_model.pkl i vectorizer.pkl.

3. Przetestuj nowy model na zbiorze testowym, aby upewnić się, że jego wydajność jest satysfakcjonująca.

Aktualizacja pakietów:

1. Sprawdź aktualizacje pakietów używanych w projekcie (Flask, scikit-learn itd.).
2. Zaktualizuj plik requirements.txt i przeprowadź testy, aby upewnić się, że aplikacja działa poprawnie z nowymi wersjami pakietów.

Aktualizacja rozszerzenia:

1. Dodaj nowe funkcje lub popraw błędy w plikach popup.js, background.js i styles.css.
2. Przetestuj rozszerzenie w przeglądarce Chrome, aby upewnić się, że działa zgodnie z oczekiwaniami.

5. Rozwiązywanie problemów

Typowe problemy i rozwiązania:

1. Problem z logowaniem:
- Upewnij się, że dane logowania do Outlook są poprawne.
- Sprawdź, czy serwer IMAP jest dostępny.

2. Problem z pobieraniem wiadomości e-mail:
- Upewnij się, że backend Flask działa poprawnie.
- Sprawdź, czy połączenie z serwerem e-mail jest stabilne.

3. Problem z klasyfikacją wiadomości:
- Upewnij się, że model klasyfikacji (spam_classifier_model.pkl) i wektoryzator (vectorizer.pkl) są załadowane poprawnie.
- Przeprowadź testy jednostkowe dla funkcji klasyfikacji w app.py.