ΠΡΟΤΑΣΗ ΕΛΛΑΚ – ΑΝΑΛΥΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΚΑΙ ΟΔΗΓΙΕΣ

PYTHON MAKER

Έργο της ομάδας MindLab-AI Guardian για τον 6ο ΠΑΝΕΛΛΗΝΙΟ ΔΙΑΓΩΝΙΣΜΟ ΑΝΟΙΧΤΩΝ ΤΕΧΝΟΛΟΓΙΩΝ ΣΤΗΝ ΕΚΠΑΙΔΕΥΣΗ

Περιγραφή:

Η ομάδα μας, αποτελούμενη από μαθητές Γυμνασίου-Λυκείου, αναλαμβάνει το έργο "AI Guardian" για τον 6ο ΠΑΝΕΛΛΗΝΙΟ ΔΙΑΓΩΝΙΣΜΟ ΑΝΟΙΧΤΩΝ ΤΕΧΝΟΛΟΓΙΩΝ ΣΤΗΝ ΕΚΠΑΙΔΕΥΣΗ. Η ανάπτυξη του προγράμματος θα επικεντρωθεί στην αναγνώριση και κατηγοριοποίηση μηνυμάτων και e-mails ως spam ή phishing, χρησιμοποιώντας προγραμματισμό σε Python και τεχνικές τεχνητής νοημοσύνης.

Κεντρική ιδέα του έργου μας είναι η δημιουργία μιας εφαρμογής που θα επιτρέπει στον χρήστη να ελέγχει την ασφάλεια των μηνυμάτων που λαμβάνει. Αρχικά, ο χρήστης θα εισάγει ένα κείμενο ή θα αντιγράφει το e-mail που έλαβε, και το πρόγραμμα θα αναλύει το κείμενο χρησιμοποιώντας τον αλγόριθμο Naive Bayes classifier.

Η εφαρμογή παράλληλα, θα δίνει στον χρήστη τη δυνατότητα να πραγματοποιήσει αναζήτηση στο διαδίκτυο για σχετικές αναφορές και κριτικές από άλλους χρήστες που έχουν λάβει παρόμοια μηνύματα. Επιπλέον, η εφαρμογή μας μπορεί να επιλέξει να έχει πρόσβαση σε δεδομένα από το διαδίκτυο, για επιπλέον ενημέρωση και ανάλυση.

Ένα ακόμη χαρακτηριστικό που προσθέτουμε είναι η δυνατότητα της εφαρμογής να πραγματοποιεί live check των e-mails του χρήστη, εντοπίζοντας αν είναι spam ή όχι και παρέχοντας σχετικά προειδοποιητικά μηνύματα, όταν ο χρήστης το επιλέγει.

Τα λογισμικά που θα χρησιμοποιηθούν για την υλοποίηση του έργου είναι:

- Python για τον προγραμματισμό
- Naive Bayes classifier για την ταξινόμηση των δεδομένων
- Πρόσβαση σε δεδομένα από το διαδίκτυο για επιπλέον ενημέρωση
- Επικοινωνία με τον χρήστη μέσω γραφικού περιβάλλοντος

Σκοπός:

Σκοπός της εφαρμογής είναι να προσφέρει ένα ισχυρό εργαλείο για την προστασία του χρήστη από απειλές όπως το spam και το phishing, ενώ παράλληλα παρέχει επιπλέον λειτουργίες όπως live check και αναζήτηση αναφορών από άλλους χρήστες.

Στόχοι:

1) Η απόκτηση γνώσεων στο κομμάτι των διαδικτυακών απειλών και ηλεκτρονικού εγκλήματος spam/phishing

2) Εμβάθυνση σε έννοιες Τεχνητής Νοημοσύνης και Δικτύωσης σε Python

3) Ανάπτυξη δεξιοτήτων όπως η κρητική σκέψη, η συνεργασία, δημιουργικότητα κ.α.

Σχεδιασμός-Υλοποίηση:

Κατά τη δημιουργία της εφαρμογής AI Guardian ακολουθήσαμε την εξής διαδικασία:

1. **Εισαγωγή των Απαραίτητων Βιβλιοθηκών:**

Αρχικά, εξηγήσαμε τις βιβλιοθήκες που χρησιμοποιούνται, όπως η tkinter για το GUI, η sklearn για τον Naive Bayes classifier, η nltk για την επεξεργασία φυσικής γλώσσας και η requests για τις HTTP αιτήσεις.

```
import tkinter as tk
from tkinter import Text, Label, Button, messagebox
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
import nltk
import requests
```

2. **Καθαρισμός του Κειμένου:**

Δημιουργήσαμε τη συνάρτηση clean_text, όπου η λειτουργία της είναι να αφαιρεί τις «στοπ» λέξεις και τα σημεία στίξης από το κείμενο.

<u>Αναλυτικότερα:</u>

Αυτή η συνάρτηση χρησιμοποιεί το Natural Language Toolkit (NLTK) για να καθαρίσει το κείμενο από στόχους που δεν προσθέτουν νόημα, όπως στόπ λέξεις (stop words), σημεία στίξης και άλλα.

```
# Function to clean the text
def clean_text(text):
    stop_words = set(stopwords.words('english'))
    words = word_tokenize(text)
    cleaned_words = [word.lower() for word in words if word.isalpha() and word.lower() not in stop_words]
    return ' '.join(cleaned_words)
```

3. **Εκπαίδευση του Ταξινομητή (Naive Bayes Classifier):**

Δημιουργήσαμε και εκπαιδεύσαμε τον Naive Bayes Classifier χρησιμοποιώντας ένα παράδειγμα με δεδομένα εκπαίδευσης. Εξηγήσαμε και αναλύσαμε τη λειτουργία του, όπου μάθαμε ότι για να εξασφαλίσουμε την πιστότητα της πρόβλεψής μας , πρέπει να εκπαιδεύσουμε το μοντέλο μας με όσο το δυνατόν περισσότερες πληροφορίες μπορούμε, έτσι ώστε να εξασφαλίσουμε ακρίβεια στη μέθοδό μας.

<u>Αναλυτικότερα:</u>

Στο X , εισάγουμε τα δεδομένα και τα αντιστοιχούμε στη μεταβλητή y ως spam ή not spam. Έπειτα, καθαρίζουμε τα δεδομένα στο X για να τα επεξεργαστούμε.

```
# Sample data
X = ["Get a free iPhone now!", "Meeting at 10 AM tomorrow", "Hello!", "Give me your bank acount!",
y = ["spam", "not spam", "not spam", "spam", "spam", "spam", "not spam", "not spam"]

X = [clean_text(text) for text in X]
```

4. **Έλεγχος Spam (check_spam):**

- Στο σημείο αυτό, ο κώδικας παίρνει το κείμενο που εισήγαγε ο χρήστης.
- Κατόπιν, το κείμενο καθαρίζεται από περιττές λέξεις και σημεία στίξης.
- Έπειτα, χρησιμοποιείται ένας Naive Bayes Classifier για να προβλεφθεί εάν το κείμενο είναι spam ή όχι.
- Αν το κείμενο εντοπιστεί ως spam, εμφανίζεται ένα μήνυμα προειδοποίησης στον χρήστη.

<u>Αναλυτικότερα:</u>

Αυτή η συνάρτηση παίρνει το κείμενο που εισήγαγε ο χρήστης, το καθαρίζει και στη συνέχεια χρησιμοποιεί έναν Naive Bayes Classifier για να προβλέψει εάν είναι spam ή όχι. Εάν είναι spam, εμφανίζεται ένα μήνυμα προειδοποίησης.

```
# Function to check spam
def check_spam():
    input_text = text_entry.get("1.0",'end-1c')
    cleaned_text = clean_text(input_text)
    vectorized_text = vectorizer.transform([cleaned_text])
    prediction = classifier.predict(vectorized_text)[0]

    result_label.config(text=f"Result: {prediction}")

    if prediction == 'spam':
        messagebox.showwarning("Warning", "This message may be spam!")
    else:
        messagebox.showinfo("Info", "This message is not spam.")
```

5. **Αναζήτηση Κριτικών (check_reviews):**
   - Σε αυτό το βήμα, ο χρήστης έχει τη δυνατότητα να εισάγει το email του αποστολέα.
   - Η εφαρμογή, στη συνέχεια, χρησιμοποιεί το Yelp API για να αναζητήσει κριτικές που σχετίζονται με το email αποστολέα.
   - Εάν δεν εισαχθεί email, εμφανίζεται ένα μήνυμα προειδοποίησης.
   - Οι κριτικές που εντοπίζονται εμφανίζονται στον χρήστη για περαιτέρω εξέταση.

   <u>Αναλυτικότερα:</u>

   Αυτή η συνάρτηση χρησιμοποιεί το Yelp API για να αναζητήσει κριτικές σχετικά με το email αποστολέα. Εάν δεν υπάρχει email, εμφανίζεται ένα μήνυμα προειδοποίησης. Οι κριτικές που βρέθηκαν εμφανίζονται στον χρήστη (Το ορίζουμε εμείς, από τη μεταβλητή limit, στην συνάρτηση search_yelp_reviews()). Το query αναζήτησης στο Yelp, το δίνει ο χρήστης μέσω της γραφικής διεπαφής που δημιουργήθηκε.

```python
def search_yelp_reviews(api_key, query):
    headers = {
        "Authorization": f"Bearer {api_key}"
    }

    url = "https://api.yelp.com/v3/businesses/search"

    params = {
        "term": query,
        "limit": 5
    }

    response = requests.get(url, headers = headers, params = params)
    data = response.json()

    reviews = []

    for business in data.get("businesses", []):
        reviews.append(business.get("name", "N/A"))

    return reviews

def check_reviews():
    sender_email = review_entry.get()
    if not sender_email:
        messagebox.showwarning("Warning", "Please enter the sender's email for review!")
        return
    search_results = search_yelp_reviews(yelp_api_key, sender_email)

    messagebox.showinfo("Yelp Reviews", f"Yelp Reviews for {sender_email}: \n{','.join(search_results)}")
```

6. Λειτουργίες GUI:

Δημιουργήσαμε τα κουμπιά, τα κείμενα και τα στοιχεία εισόδου στο GUI, συνδέσαμε τις λειτουργίες των κουμπιών με τις αντίστοιχες συναρτήσεις και έτσι, δημιουργήσαμε το γραφικό περιβάλλον της εφαρμογής μας.

Αναλυτικότερα:

Ο κώδικας για τη δημιουργία του γραφικού περιβάλλοντος χρησιμοποιεί τη βιβλιοθήκη Tkinter για να δημιουργήσει ένα παράθυρο GUI. Περιλαμβάνει ετικέτες (Labels), κουτιά κειμένου (Text Entry), κουμπιά ενέργειας (Buttons) και μηνύματα προειδοποίησης (Message Boxes).

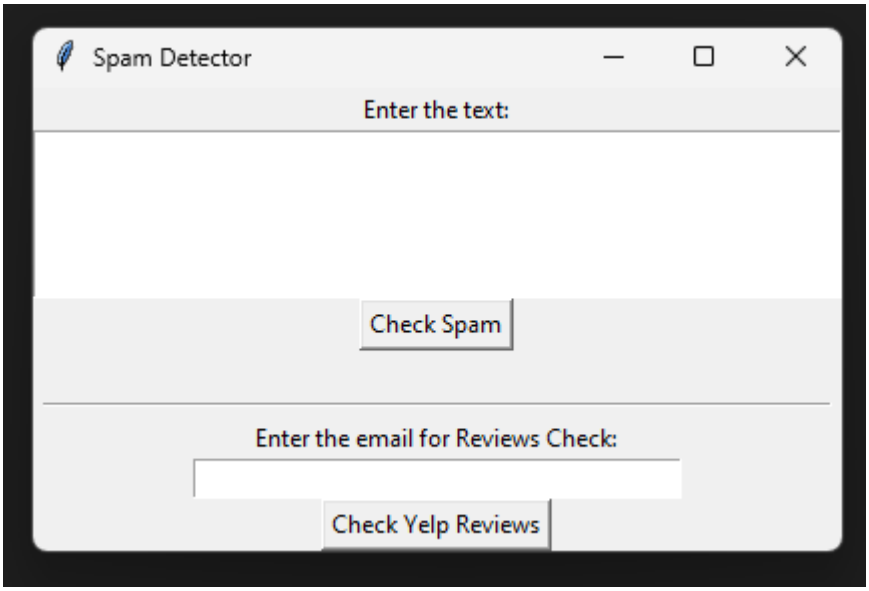
Αυτές οι λειτουργίες ενσωματώνονται σε ένα γραφικό περιβάλλον που επιτρέπει στον χρήστη να ελέγχει τα μηνύματά του για spam και phishing, καθώς και να αναζητά κριτικές σχετικά με το email αποστολέα.

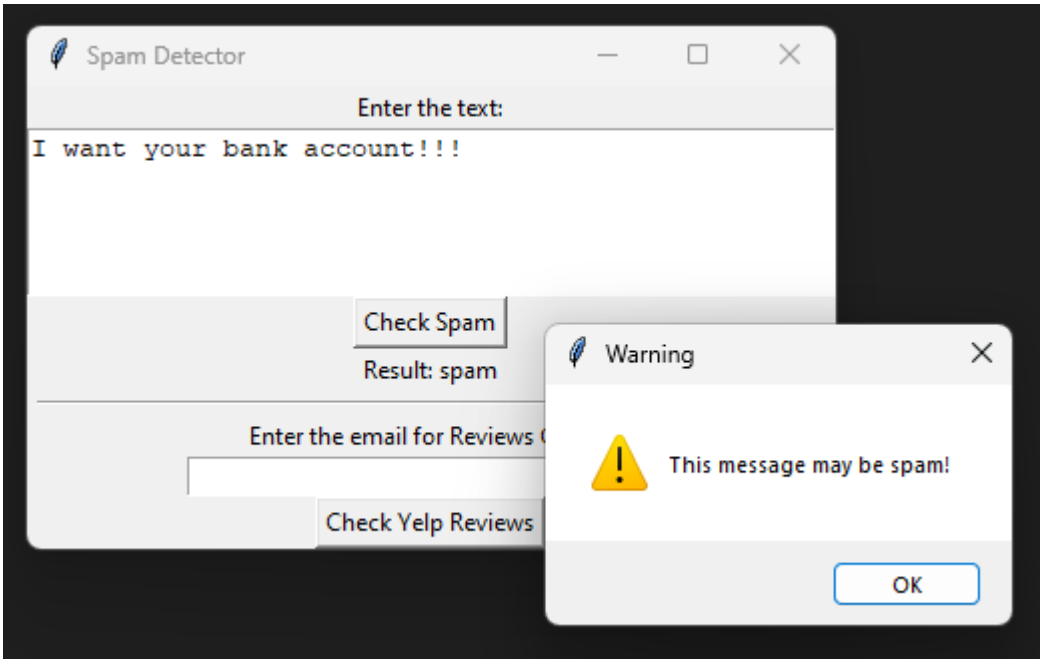
```python
# Create the GUI
app = tk.Tk()
app.title("Spam Detector")
```

```
# Text label
text_label = Label(app, text="Enter the text:")
text_label.pack()

# Text widget for text input
text_entry = Text(app, height=5, width=50)
text_entry.pack()

# Check button
check_button = Button(app, text="Check Spam", command=check_spam)
check_button.pack()

# Result label
result_label = Label(app, text="")
result_label.pack()

separator  =  tk.Frame(height = 2, bd = 1, relief = tk.SUNKEN)
separator.pack(fill = tk.X, padx = 5, pady = 5)


review_label = Label(app, text = "Enter the email for Reviews Check:")
review_label.pack()

review_entry = Text(app, height=1, width=30)
review_entry.pack()

review_button = Button(app, text="Check Yelp Reviews", command=check_reviews)
review_button.pack()


# Start the GUI
app.mainloop()
```

<u>Η γραφική διεπαφή της εφαρμογής:</u>

Έχει δύο πεδία συμπλήρωσης, ένα για να αντιγράψει ο χρήστης το μήνυμα που του ήρθε στο e-mail του και ένα, για να εισάγει ο χρήστης το e-mail του αποστολέα που θεωρεί ύποπτο, αφού δημιουργήσει λογαριασμό στο Yelp και αιτηθεί, το Yelp API key.





7. Επέκταση και Προσαρμογή:

Μετά την κατανόηση της βασικής λειτουργικότητας, ενθαρρύναμε τους μαθητές να επεκτείνουν τη λειτουργικότητα της εφαρμογής, προσθέτοντας νέες λειτουργίες ή βελτιώνοντας τις υπάρχουσες. Μια ιδέα θα μπορούσε να είναι η προσθήκη ενός μηχανισμού για την απόκρυψη των ευαίσθητων πληροφοριών που εισάγονται από τον χρήστη, όπως το email του, για περισσότερη ασφάλεια.