```python
import os
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report
from email_processor import load_emails_from_folder, preprocess_email
from joblib import dump, load  # Importez dump et load depuis joblib
```

```python
    else:  # Si la prédiction est que l'email ne contient pas un événement
        print(f"Email trouvé dans {filename}:")
        print(f"Sujet: {filename}")
        print(f"Corps: {email}")
        print(f"Prédiction actuelle: Ne contient pas d'événement (Probabilité: {probability:.2f})")
        print("Que souhaitez-vous corriger ?")
        print("1. L'IA a mal classifié cet email (contient un événement)")
        print("2. Aucune action nécessaire")

        user_choice = input("Votre choix (1/2/3) : ")
        if user_choice == "1":
            emails_avec_eve_test.append((filename, probability))
        elif user_choice == "2":
            emails_sans_eve_test.append((filename, probability))

print("Emails contenant des événements :")
for filename, probability in emails_avec_eve_test:
    print(f"Fichier: {filename} - Probabilité: {probability:.2f}")

print("\nEmails ne contenant pas d'événements :")
for filename, probability in emails_sans_eve_test:
    print(f"Fichier: {filename} - Probabilité: {probability:.2f}")

return pipeline  # Retourne le modèle entraîné
```

Le code possède la fonction train_and_classify_emails qui entraîne un modèle de machine learning pour classifier des emails comme contenant ou non des événements, en utilisant le français comme langue de traitement (via Spacy). Voici une description des fonctionnalités principales de ce code :

1. **Chargement du modèle Spacy** : Le modèle linguistique français fr_dep_news_trf est chargé via Spacy pour le prétraitement des emails.
2. **Vérification des répertoires d'entrainement** : Le code vérifie l'existence des dossiers contenant les emails avec événements (path_avec_eve) et sans événements (path_sans_eve). S'ils n'existent pas, ils sont créés.
3. **Chargement ou entraînement du modèle** :
    - Si un fichier modèle (par défaut "trained_model.joblib") existe déjà, le modèle est chargé via joblib.load.
    - Sinon, les emails des dossiers sont chargés, étiquetés (1 pour les emails avec événements, 0 pour ceux sans), et prétraités avec la fonction preprocess_email.

- Un pipeline est créé avec un TfidfVectorizer (pour transformer les emails en vecteurs de caractéristiques) et un classificateur SVC (Support Vector Machine) avec un noyau linéaire. Ce modèle est entraîné sur les emails prétraités et les étiquettes.
- Le modèle entraîné est sauvegardé avec joblib.dump.

4. **Prédiction et évaluation** :
    - Les emails du dossier de test (path_teste) sont chargés et prétraités. Pour chaque email, le modèle prédit la probabilité qu'il contienne un événement.
    - Si la prédiction est incorrecte, l'utilisateur peut manuellement corriger la classification via une entrée utilisateur (input) dans la console.
5. **Affichage des résultats** :
    - Après la classification, le programme affiche les emails classifiés comme contenant ou non des événements, avec les probabilités associées.

Fonctionnalités supplémentaires :

- Le modèle sauvegardé peut être réutilisé sans nécessiter de réentraînement, ce qui optimise le temps d'exécution pour les futures classifications.
- La correction manuelle permet d'améliorer la précision du modèle en ajustant les erreurs de classification faites par l'IA.